Testing diversity-enhancing migration policies for hybrid on-line evolution of robot controllers *

P. García-Sánchez[1], A. E. Eiben[2], E. Haasdijk[2], B. Weel[2] and J.J. Merelo[1]

[1] Dept. of Computer Architecture and Technology, University of Granada, Spain
[2] Dept. of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
pgarcia@atc.ugr.es

Abstract. We investigate on-line on-board evolution of robot controllers based on the so-called hybrid approach (island-based). Inherently to this approach each robot hosts a population (island) of evolving controllers and exchanges controllers with other robots at certain times. We compare different exchange (migration) policies in order to optimize this evolutionary system and compare the best hybrid setup with the encapsulated and distributed alternatives. We conclude that adding a difference-based migrant selection scheme increases the performance.

1 Introduction

Evolutionary robotics concerns itself with evolutionary algorithms to optimise robot controllers [7]. Traditionally, robot controllers evolve in an off-line fashion, through an evolutionary algorithm running on some computer searching through the space of controllers and only calling on the actual robots when a fitness evaluation is required. To distinguish various options regarding the evolutionary system Eiben et al. proposed a naming scheme based on *when*, *where* and *how* this evolution occurs [3]. The resulting taxonomy distinguishes between design time and run-time evolution (off-line vs. on-line) as well as between evolution inside or outside the robots themselves (on-board vs. off-board). In a system comprising of multiple robots, there are three options regarding the *'how'*:

Encapsulated: A population of genotypes encoding controllers evolves inside each robot independently, without communication with other robots.

Distributed: Each robot carries a single genotype and reproduction requires the exchange of genotypes with other robots. The evolving population is formed by the combined genotypes of all the robots.

Hybrid: Each robot has its own locally evolving population and there is exchange of genotypes between robots. In terms of parallel evolutionary algorithms, this can be seen as an island-model evolutionary algorithm with migration.

* This work was supported in part by Spanish Projects EvOrq (TIC-3903), CEI BioTIC GENIL (CEB09-0010), MICINN CEI Program (PYR-2010-13) and FPU research grant AP2009-2942 and the European Union FET Proactive Initiative: Pervasive Adaptation funding the SYMBRION project under grant agreement 216342. The authors wish to thank Selmar Smit and Luis Pineda for their help and fruitful discussions.

In this paper we investigate aspects of the hybrid approach: we test the effects of the *migration policy* (migration of the best, random, or most different individual), the *admission policy* (always accept the migrant, or accept only after re-evaluation) and the *island topology* (ring vs. fully connected). Furthermore, we look into these effects for different numbers of robots (4, 16 or 36).

Specifically, our research questions are:

- Using the hybrid approach (island model), which is the best combination of migration policy, admission policy, and island topology?
- Is this combination better than the encapsulated and distributed alternatives?

The rest of the work is structured as follows: after the state of the art, we present the developed algorithms and experimental setting. Then, the results of the experiments are shown (Section 4), followed by conclusions and suggestions for future work.

2 State of the art

Migration among otherwise reproductively isolated populations has been proven to leverage the inherent parallelism in evolutionary algorithms, not only by obtaining speed-ups, but also by increasing the quality of results, since the reproduction restrictions inherent in the division of the population into islands is a good mechanism to preserve population diversity, as shown in, for instance, [2].

To improve population diversity in an island model evolutionary algorithm, the MultiKulti algorithm [1] takes the genotypic differences of individuals when selecting migrants into account. It is based in the idea that the inflow of migrants that differ from the rest of an island's population increases diversity and thus improves performance. An island requests a migrant from one of its neighbours by sending a genotype that represents the population. This can either be the the best individual (based in the assumption that when a population tends to converge after a few generations, the best is a fair representation of the whole population) or a *consensus sequence* (the most frequent allele in each position of the genotype using binary genomes). In answer, an island selects the most different genotype in either its whole population or the top individuals (the elite). In their experiments, the islands were connected in a ring topology, with migration taking place asynchronously. Results of the experiments performed in [1] show that MultiKulti policies outperform classic migration policies (send the best or random individuals from the population), especially with a low number of individuals but larger number of islands. It is shown to be better to send the consensus as a representation and that sending the most different of a well-chosen elite (those with the best fitness) is better than sending the most different overall.

On-line evolutionary robotics has been studied in works like [8], where genetic programming was used to evolve a robot in real time, and [11], where several robots evolve at the same time, exchanging parts of their genotypes when within

communication range. [5] compares an encapsulated and a distributed version; the latter is implemented as a variant of EVAG [6], where each robot has one active solution (genotype) a cache of genotypes that are active in neighbouring robots. Parents are selected through a binary tournament in each robot's cache. If the new solution (candidate) is better than the active, it replaces the active solution. The work compares this algorithm with a panmictic version, where parents are selected (again using binary tournament) from the combined active solutions of all robots.

One of the peculiarities of evolutionary robotics, particularly on-line, is that the fitness evaluations are very noisy [4]. The conclusions in [1], however, are based on experiments with noiseless fitness functions, so we cannot take these conclusions for granted in on-line evolutionary robotics and we have to test the MultiKulti algorithm in our particular setting.

3 Algorithms and Experimental Setup

We carried out our experiments with e-puck like robots simulated in the RoboRobo simulator[3]. The robot is controlled by an artificial neural net with 9 inputs (corresponding to the robot's distance sensors and a bias node), 2 outputs (wheel speeds). Genetically, this was represented as a vector coding the network's 18 weights. All algorithms were evaluated using the *Fast Forward* task and next fitness function:

$$f = \sum_{t=0}^{\tau} (v_t \cdot (1 - v_r)) \tag{1}$$

where v_t and v_r are the translational and the rotational speed, respectively. v_t is normalised between -1 (full speed reverse) and 1 (full speed forward), v_r between 0 (movement in a straight line) and 1 (maximum rotation). Whenever a robot touches an obstacle, $v_t = 0$, so the fitness increment during collisions is 0. There is more information about this function in [5]. This fitness is noisy: a controller configuration can produce different fitness values depending on the robot's position in the arena when evaluation starts. The robots are placed in a small maze-like arena (Fig. 2). To ensure a fair comparison across different numbers of robots, each robot is placed in a separate instance of the arena to avoid physical interaction between robots. Robots can communicate across arenas instances.

In our experiments, we compare three algorithms:

Encapsulated evolutionary algorithm The encapsulated algorithm we use is the $\mu + 1$ on-line algorithm presented in [4]. Here, each robot runs a stand-alone evolutionary algorithm with a local population of μ individuals. In each cycle, one new solution (controller) is created and evaluated. This solution replaces the worst individual of the population if it has higher fitness. To combat the effects of noisy evaluations, an existing solution can be re-evaluated, instead of generating and testing a new one, depending on the re-evaluation rate ρ.

[3] `http://www.lri.fr/~bredeche/roborobo/`

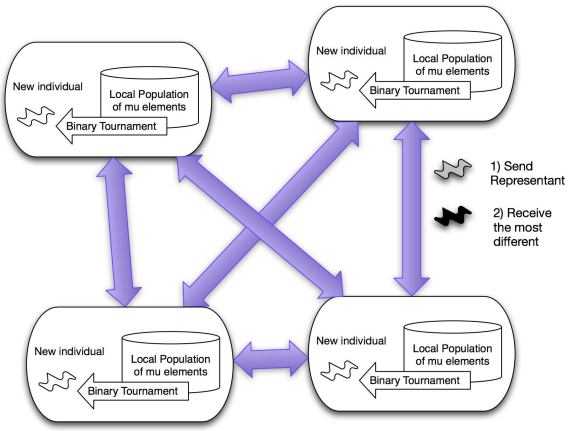


Fig. 1: Migration mechanism: each robot has a local population and in each migration cycle request a different type individual from others robots' populations. If MultiKulti is used, then a message is sent (gray genotype) to receive the most different (black genotype).

Distributed evolutionary algorithm As a benchmark distributed algorithm we use the panmictic algorithm presented in [5]. Here, a single controller is present in each robot. New controllers are created using the controllers of two robots as parents. In each iteration, a robot randomly selects two others to create a new chromosome by crossover and mutation. If the new chromosome is better, it replaces the actual one.

Hybrid evolutionary algorithm This algorithm is an adaptation of the μ+ 1 on-line algorithm that includes a migration mechanism to exchange genotypes among robots (every robot is an island) as shown in Fig. 1. We test two migrant acceptance mechanisms: a migrant can be added to the local population either regardless of its fitness (to give it a chance to be selected) or only if it is better than the worst.[4]

Each experiment lasts for 50,000 evaluation steps. In on-line evolution, the robots train on the job: this means that the *robot's* performance is not (only) determined by the best individual it stores at any one time, but by the joint performance of all the candidate controllers it considers over a period. Therefore, we assess the algorithms' performance using the average of the last 10% evaluations over all robots.

As stated in [9], an algorithm's parameters should be tuned to obtain (approximately) the best possible parameter settings and so ensure a fair comparison between the best possible instances of the algorithms. We used Bonesa [10] to tune the parameters for the algorithms we investigate in the following configurations:

– Number of robots: executions with 4, 16 and 36 robots have been performed.

[4] Source code of the presented algorithms is available in `http://atc.ugr.es/~pgarcia`, under a GNU/GPL license.

– Migrant selection: select the *Best*, *random* or *most different* (MultiKulti) individual as a migrant.
– Admission policy: when a new migrant arrives, it is evaluated and accepted only if is better than the worst (*no-replacement*) or accepted regardless, always replacing the worst of the population (*replacement*).
– Topology: migration can move between neighbours and the islands are arranged in a *ring* or in a *random* topology, which is rewired after every evaluation.

We conducted Bonesa runs for each possible combination of these configurations to tune the settings for canonical parameters (e.g., mutation step size, crossover rate) and the following more specific parameters:

Along the canonical GA parameters (like mutation or crossover rate) the MultiKulti parameters to study are the next:

– Migration rate: likelihood of migration occurring per evaluation cycle.
– Best rate: probability of representing the population with the best individual or with a consensus sequence (average of genes). This parameter applies only for MultiKulti instances.
– Elite percentage: the size of the elite group to select the migrant from (if 1, receive the most different of all the population). This parameter applies only for MultiKulti instances.

Population size μ was fixed to 10 individuals to isolate the interactions between the other parameters. Figure 3 lists all tuned parameters and their ranges. For the final analysis, we ran 50 iterations of each configuration with the parameters set to those reported as optimal by Bonesa.

4 Results and Analysis

4.1 Comparing Migration Configurations

The first question we asked ourselves was "which is the best combination of migration policy, admission policy, and island topology?" To answer this question, we analyse the results as reported by Bonesa for each of the configurations

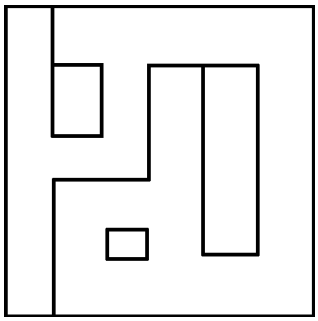


Fig. 2: Arena used in the experiments.

Parameter Name	*Range*
Evaluation steps	300-600
Mutation step size	0.1-10
μ	10
Re-evaluation rate	0-1
Crossover rate	0-1
mutation rate	0-1
migration rate	0-1
elite percentage	0-1
best Rate	0-1

Fig. 3: Parameters to tune.

we considered. Table 1 shows the best parameters obtained for all configurations with 4, 16 and 36 robots. We discuss the results in the following four paragraphs, each discussing the results for one combination of admission policy and island topology.

Replacement Admission Policy and Panmictic Topology In all cases, the re-evaluation, crossover, mutation and migration rates are very high. Also, EliteSize is almost 1 everywhere: the migrant is selected from almost the whole population. It also turns out that is better to send a consensus sequence rather than the best individual as a representative of the population (bestRate has low values). There is no clear trend for migration rate.

Replacement Admission Policy and Ring Topology Changing the island topology to a ring arrangement, three settings change materially: as can be see in Table 1 for MultiKulti with 4 and 16 robots, the migration rate is much lower, but for 36 robots it remains very high. Also, but only for 4 robots, BestRate is higher (send the best individual as representative, not the consensus sequence).

No-replacement Admission Policy and Panmictic Topology When changing the replacement policy a remarkable decrease can be seen in the migration rate and, more importantly, the re-evaluation rate across the board. For 4 robots, EliteSize is much lower than in all three other combinations of admission policy and topology.

No-replacement Admission Policy and Ring Topology Apart from lower migration rates for most of the policies and a drop in EliteSize for 16 robots, Bonesa reports similar values for this combination of admission policy and topology and the previous one. For 4 robots, EliteSize again has a high value.

Comparing Performance Figures 4a, 4b and 4c show box plots summarising 50 repeats of each configuration, grouped by number of robots.

Although in terms of performance levels there is no clear trend it is clear that the admission policy does have an appreciable impact: choosing the no-replacement admission policy always leads to a marked decrease in performance variation, with an increase of minimum performance. So we can conclude that evaluating an immigrant and only admitting it if it outperforms the worst individual in the population leads to more consistent performance with fewer very poor results.

Combined with the no-replacement admission policy, MultiKulti is either the best or at a par with the best migrant selection scheme, especially as the number of robots increases.

Finally, the ring topology shows a slight, but not always significant, drop in performance. This may be explained by the fact that in a ring topology, good solutions spread over the islands at a much slower rate than in a randomly connected topology.

Replacement admission policy and panmictic topology									
	4 ROBOTS			16 ROBOTS			36 ROBOTS		
	MK	RANDOM	BEST	MK	RANDOM	BEST	MK	RANDOM	BEST
evolutionSteps	345	310	312	310	306	425	538	561	584
stepSize	9.038	9.874	5.38	8.804	8.786	9.199	4.842	8.096	9.684
reEvaluation	0.868	0.72	0.739	0.619	0.812	0.949	0.964	0.751	0.777
Crossover	0.926	0.816	0.929	0.017	0.879	0.917	0.963	0.915	0.941
Mutation	0.943	0.977	0.936	0.98	0.839	0.909	0.937	0.923	0.938
Migration	0.809	0.989	0.958	0.987	0.499	0.993	0.956	0.988	0.567
EliteSize	0.849	-	-	0.988	-	-	0.995	-	
BestRate	0.04	-	-	0.192	-	-	0.181	-	
Replacement admission policy and ring topology									
	4 ROBOTS			16 ROBOTS			36 ROBOTS		
	MK	RANDOM	BEST	MK	RANDOM	BEST	MK	RANDOM	BEST
evolutionSteps	304	319	312	304	311	372	554	589	573
stepSize	9.29	8.149	8.769	7.008	7.37	9.953	9.465	9.307	9.94
reEvaluation	0.868	0.749	0.792	0.953	0.721	0.861	0.935	0.705	0.939
Crossover	0.999	0.983	0.96	0.83	0.955	0.455	0.996	0.848	0.991
Mutation	0.986	0.952	0.691	0.914	0.809	0.889	0.971	0.777	0.98
Migration	0.597	0.892	0.974	0.559	0.624	0.996	0.988	0.816	0.955
EliteSize	0.49	-	-	0.93	-	-	0.827	-	
BestRate	0.862	-	-	0.172	-	-	0.145	-	
No-replacement admission policy and panmictic topology									
	4 ROBOTS			16 ROBOTS			36 ROBOTS		
	MK	RANDOM	BEST	MK	RANDOM	BEST	MK	RANDOM	BEST
evolutionSteps	305	304	308	302	304	306	567	362	516
stepSize	9.895	4.04	9.547	9.731	9.146	9.8	8.832	7.526	9.988
reEvaluation	0.385	0.039	0.489	0.449	0.291	0.692	0.048	0.344	0.528
Crossover	0.828	1	0.934	0.847	0.945	0.671	0.31	0.822	0.963
Mutation	0.976	0.927	0.899	0.849	0.958	0.969	0.921	0.986	0.879
Migration	0.577	0.788	0.72	0.658	0.757	0.577	0.835	0.753	0.7
EliteSize	0.279	-	-	0.716	-	-	0.911	-	-
BestRate	0.198	-	-	0.703	-	-	0.013	-	-
No-replacement admission policy and ring topology									
	4 ROBOTS			16 ROBOTS			36 ROBOTS		
	MK	RANDOM	BEST	MK	RANDOM	BEST	MK	RANDOM	BEST
evolutionSteps	325	302	323	314	303	306	600	375	581
stepSize	9.821	9.726	9.731	9.925	8.44	9.329	9.661	9.686	9.493
reEvaluation	0.045	0.007	0.53	0.044	0.332	0.505	0.752	0.317	0.396
Crossover	0.311	0.51	0.933	0.286	0.986	0.867	0.963	0.992	0.952
Mutation	0.983	0.805	0.873	0.751	0.964	0.889	0.93	0.869	0.913
Migration	0.533	0.517	0.554	0.662	0.706	0.685	0.59	0.71	0.698
EliteSize	0.772	-	-	0.952	-	-	0.413	-	-
BestRate	0.018	-	-	0.624	-	-	0.061	-	-

Table 1: Parameters obtained with Bonesa for all admission policies and topology configurations.

Selecting migrants randomly seems always to lead to a smaller spread in performance than either selecting the best or the most different. Vis a vis the MultiKulti algorithm at least, this makes sense because this specifically aims

	4 ROBOTS			16 ROBOTS			36 ROBOTS		
	μ+1	Distr	MK	μ+1	Distr	MK	μ+1	Distr	MK
evolutionSteps	308	303	305	308	301	302	308	583	567
stepSize	9.615	4.306	9.895	9.615	5.621	9.731	9.615	8.197	8.832
reEvaluation	0.091	0.647	0.385	0.091	0.558	0.449	0.091	0.002	0.048
Crossosver	0.19	0.399	0.828	0.19	0.122	0.847	0.19	0.1	0.31
Mutation	0.978	0.908	0.976	0.978	0.86	0.849	0.978	0.606	0.921
Migration	-	-	0.577	-	-	0.658	-	-	0.835
EliteSize	-	-	0.279	-	-	0.716	-	-	0.911
BestRate	-	-	0.198	-	-	0.703	-	-	0.013

Table 2: Parameters obtained with Bonesa for the encapsulated, distributed and hybrid algorithms.

at increasing population diversity, so a larger variation in performance is to be expected.

To conclude, we select a configuration with no-replacement admission policy, MultiKulti migrant selection and a random island topology to compare with the encapsulated and distributed algorithms.

4.2 Comparing Encapsulated, Distributed and Hybrid On-line Evolution

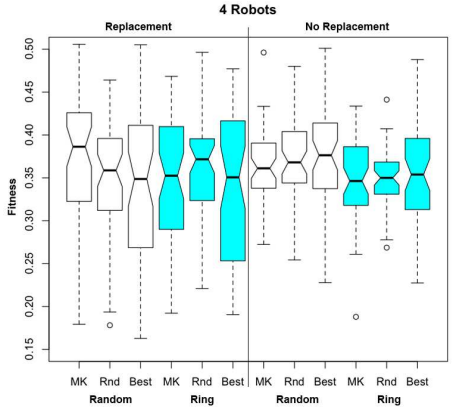
The second question we asked ourselves is whether the optimal hybrid instance we selected in the previous section outperforms its encapsulated and distributed counterparts. Table 2 shows the settings that Bonesa reported as optimal for the three algorithms that we compare for groups of 4, 16 and 36 robots. Note that the optimal parameters for $\mu + 1$ have only been calculated once: since there is no interaction between robots, $\mu + 1$'s performance and settings are independent of the number of robots. Running 50 repeats with these settings resulted in performances as reported in Figure 4d.

Even for as small a number of robots as 4, the distributed and hybrid algorithms both significantly outperform the encapsulated algorithm. The difference between the algorithms that share the population across robots is only significant for 36 robots, but even there not material. The difference with the encapsulated algorithm may lie in the exploitation of evolution's inherent parallelism, but we think this is also due to the increased diversity that stems from dividing the total population across islands. This would explain the large benefit of communication even for small numbers of robots, where the distributed algorithm actually has a smaller total population than the individual robots with $\mu + 1$.

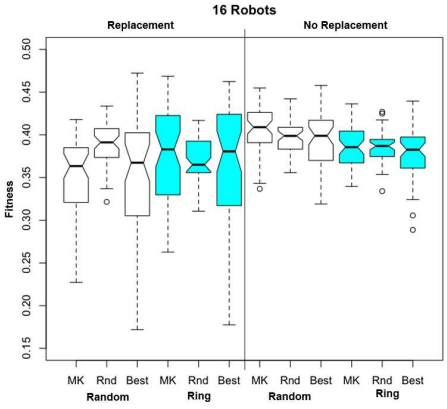
Set to the best found parameters, the hybrid algorithm causes much less communication overhead than the distributed algorithm: the latter shares genotypes among all robots at every evaluation, while the hybrid algorithm has comparatively low migration rates (0.577, 0.658 and 0.835). This reduction of communication cost comes at no significant loss of performance, and even a significant gain for 36 robots.

5 Conclusions and future work

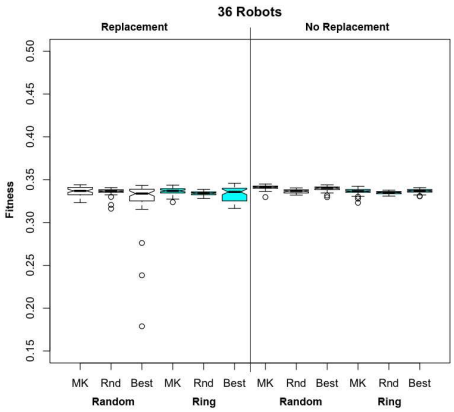
In this paper, we compared combinations of migrant selection schemes, migrant admission policies and island topologies in a hybrid algorithm for on-line, on-board Evolutionary Robotics. Results show that the migrant admission policy –which determines when a migrant is admitted into the population– is more important in performance than migrant selection or the island topology. But the most important finding is that adding migration between robots signifi-



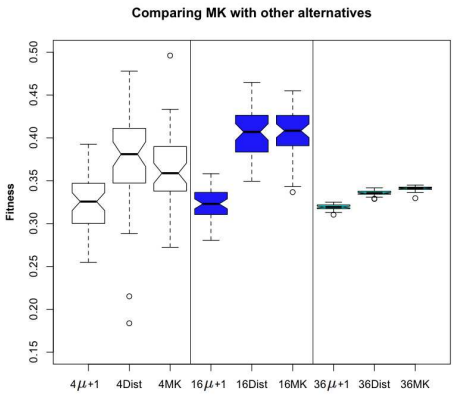
(a) Box plot of all hybrid configurations with 4 robots.



(b) Box plot of all hybrid configurations with 16 robots.



(c) Box plot of all hybrid configurations with 36 robots.



(d) Box plot comparing $\mu + 1$, distributed and multikulti migration with replacement for 4, 16 and 36 robots.

Fig. 4: Box plots of executing each algorithm with the best parameters obtained with Bonesa 50 times.

cantly and materially increases performance. We have demonstrated that adding a difference-based migrant selection scheme (MultiKulti) leads to optimal or at least near-optimal performance compared to another migration mechanisms. This migration mechanism can compete with the on-line distributed algorithm, where only an individual per robot exist, even with a lower number of data transmissions. Our aim is to continue exploring other techniques, like a self-adaptative migration mechanism to ask for new migrants when the population stagnates and perform new tests for new tasks other than the Fast-Forward. New experiments with different number of individuals in the local population also will be carried out. Also, further investigation will be performed in swarming and cooperation techniques among robots, with different communication mechanisms.

References

1. Lourdes Araujo and Juan Julián Merelo. Diversity through multiculturality: Assessing migrant choice policies in an island model. *IEEE Trans. Evolutionary Computation*, 15(4):456–469, 2011.
2. E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of heuristics*, 7(4):311–334, 2001.
3. A.E. Eiben, E. Haasdijk, and N. Bredeche. Embodied, on-line, on-board evolution for autonomous robotics. In P. Levi and S. Kernbach, editors, *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, volume 10, pages 361–382. Springer, 2010.
4. Evert Haasdijk, A. E. Eiben, and Giorgos Karafotias. On-line evolution of robot controllers by an encapsulated evolution strategy. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010. IEEE Computational Intelligence Society, IEEE Press.
5. Robert-Jan Huijsman, Evert Haasdijk, and A.E. Eiben. An on-line, on-board distributed algorithm for evolutionary robotics (TO APPEAR). In *Proceedings of the Biennial International Conference on Artificial Evolution (EA-2011)*, 2011.
6. Juan Luís Jiménez Laredo, A. E. Eiben, Maarten van Steen, and Juan Julián Merelo. Evag: a scalable peer-to-peer evolutionary algorithm. *Genetic Programming and Evolvable Machines*, 11(2):227–246, 2010.
7. Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT, 2000.
8. P. Nordin and W. Banzhaf. An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behavior*, 5(2):107–140, 1997.
9. S.K. Smit and A.E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 399–406. IEEE, 2009.
10. S.K. Smit and A.E. Eiben. Multi-problem parameter tuning (TO APPEAR). In *Proceedings of the Biennial International Conference on Artificial Evolution (EA-2011)*, 2011.
11. R.A. Watson, S.G. Ficici, and J.B. Pollack. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18, 2002.