Coupling Evolution and Information Theory for Autonomous Robotic Exploration

Guohua Zhang[1,2] and Michèle Sebag[2]

[1] Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu 610041, China
[2] TAO, CNRS − INRIA − LRI, Université Paris-Sud, 91128 Orsay Cedex, France

Abstract. This paper investigates a hybrid two-phase approach toward exploratory behavior in robotics. In a first phase, controllers are evolved to maximize the quantity of information in the sensori-motor datastream generated by the robot. In a second phase, the data acquired by the evolved controllers is used to support an information theory-based controller, selecting the most informative action in each time step.
The approach, referred to as EvITE, is shown to outperform both the evolutionary and the information theory-based approaches standalone, in terms of actual exploration of the arena. Further, the EvITE controller features some generality property, being able to efficiently explore other arenas than the one considered during the first evolutionary phase.

Keywords: Evolutionary robotics, information theory, intrinsic motivation, entropy.

1 Introduction

Quite a few disciplinary fields are concerned with building autonomous robotic controllers, ranging from optimal control to evolutionary robotics (ER) [14], machine learning (ML) and specifically reinforcement learning (RL) [19,6] and artificial intelligence (AI) [16]. For the sake of computational and experimental conveniency, many approaches rely on simulators; the relevance of the resulting controllers thus is subject to the so-called reality gap [10,17].

This paper is concerned with building robotic controllers in an in-situ, simulator-free fashion. Whereas the simulator-free setting sidesteps the reality gap, it raises the challenge of defining intrinsic criteria (optimization objective or decision hints respectively supporting the on-board evolution or decision making process) without any ground truth about the appropriateness of the robot behavior in its environment. This study is primarily motivated by swarm robotics [11], where simulator-based approaches face a super-linear computational complexity w.r.t. the number of robots in the swarm.

The proposed approach takes inspiration from the intrinsic motivation criterion [15,2,12] and from information theory-driven evolutionary robotics [3] (more in section 2), with scalability and generality as main criteria. The presented

T. Bartz-Beielstein et al. (Eds.): PPSN XIII 2014, LNCS 8672, pp. 852–861, 2014.

approach, referred to as EvITE (*Evolution and Information Theory-based Exploratory Robotics*) and introduced in section 3, combines machine learning and evolutionary principles. Formally, in a first phase, evolutionary controllers are built along the same ideas as in [3]; in a second phase the data acquired by these controllers is used to support an information theory-based controller. The experimental validation of EvITE shows that it outperforms the evolutionary and the IT controllers standalone, in terms of exploration and behavioral diversity [7,9] (section 4). Importantly, EvITE features some *generality* property, in the sense that it also achieves good exploratory performance in other arenas than the one used to train the evolutionary controllers. The paper discusses the complementarity of evolution and ML approaches and how to best combine them to build robotic controllers, and concludes with some perspectives for further research.

2 Related Work

This section discusses a few approaches to autonomous robotics, based on evolutionary robotics (ER) or machine learning (ML) or combining both.

ER (with the exception of [7,9,8], see below) considers a fitness function that independently maps each controller trajectory on $\mathbb{R}$. A first challenge is to define an appropriate search space, enabling to describe powerful controllers while supporting evolutionary optimization [22]. A second challenge is that ER faces a noisy optimization problem: finding the controller π such that the expectation of the fitness of the trajectories (where the expectation is taken over all trajectories generated from π, reflecting e.g. the robot sensor and motor noise), is maximal. How to deal with this noisy optimization problem, and keep the number of trajectories needed to estimate the expectation within reasonable computational cost has been studied for instance by [5].

In mainstream ML [18], a fixed instant reward on each (state,action) pair is defined under the Markovian assumption that the current reward only depends on the previous state. The learning process then focuses on estimating the value $Q(s, a)$, defined as the maximal cumulative reward the robot can get after selecting action a in state s. An optimal controller immediately follows from the optimal value function, by selecting in state s the action a^* maximizing $Q(s, a)$. (In practice, the learning process alternates between estimating Q and optimizing π; the details are left out for simplicity, referring the reader to [6] for a comprehensive presentation).

Hybrid ML-ER approaches include Genetic-based Machine Learning, and specifically Learning Classifier Systems variants [20,21], based on the evolutionary learning of sets of (condition-action) or (condition-action-effect) rules. While LCS rules enable in principle to control both the robot and its internal state, GBML seemingly faces scalability issues compared to Neural Nets and even more to new NN-based controller representations [22]. Another hybrid approach, based on the notion of *intrinsic motivation*, is pioneered by [15,2,12]. Let K be a trajectory, sequence of the (state s_t, action a_t) pairs generated by a controller along time, and let $\mathcal{K}_i$ denote the archive of the first i trajectories generated by the

learning/optimization process. A so-called forward model f_i is learned from $\mathcal{K}_i$, estimating the transition model in the robot environment, that is the next state s' of the robot after selecting action a in state s, where S and A respectively stands for the set of states and set of actions. The key point is that the accuracy of f_i can be estimated on-board during the next trajectory of the robot, as the robot observes the state s_{t+1} yielded by selecting action a_t in state s_t. The accuracy $Acc(f_i)$ of f_i on the next trajectory K thus defines an intrinsic information, accessible to the robot without any external ground truth.

$$f_i : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S} \qquad Acc(f_i) = Pr(s_{t+1} = f_i(s_t, a_t) | (s_t, a_t, s_{t+1}) \in K)$$

Note that the above accuracy defines a misleading fitness, as a motionless controller ($s_{t+1} = s_t$) would get a very high fitness. The intrinsic motivation (IM) fitness $\mathcal{F}_{IM}$ therefore associates to a controller the Acc increase:

$$\mathcal{F}_{IM}(\pi) = Acc(f_{i+1}) - Acc(f_i) \tag{1}$$

The optimization of fitness $\mathcal{F}_{IM}$ thus yields controllers which explore new regions of the (state,action) space, providing new samples and thereby ultimately yielding an optimal forward model. Most interestingly, $\mathcal{F}_{IM}$ does not reward the extra-exploration of noisy regions: if a (state,action) region is noisy − due to e.g. stochastic noise in the environment − repeated explorations of this region are useless as they do not improve the forward model accuracy.
A related though simpler approach, based on the so-called curiosity- or discovery-driven fitness, was proposed in [3]. To each trajectory $(s_t, a_t)_{t=1,\ldots T}$ generated by a controller π is associated the entropic fitness $\mathcal{F}_E$

$$\mathcal{F}_E(\pi) = -\sum_s p_s \log p_s \tag{2}$$

where s ranges over the states visited in the trajectory and p_s is the fraction of time the trajectory visits s. By construction, an optimal controller according to $\mathcal{F}_E$ is one uniformly visiting the state space. Along this line, the entropic fitness would provide good material in order to build an accurate forward model, too. Note however that entropic fitness does not require a forward model to be learned, contrarily to intrinsic motivation. Its limitation compared to IM is that it might tend to over-explore stochastic regions of the state space, if any.

In summary, ML and ER differ in the way they build a controller and use the memory of the building process. The result of the ML process is a value function, and the controller can be explicitly derived from the value function. ER memorizes the evolution process through the ER controller itself, expressed as e.g. the weight vector of a neural net, and through the archive of the past trajectories. This archive makes it feasible to define more sophisticated fitness functions. For instance, [9,8] characterize and exploit the difference between a trajectory and the past ones to enforce the robust and creative sampling of the trajectory space; [7,13] likewise use this diversity, possibly along a multi-objective framework; [3] further defines a discovery-driven fitness, computing the conditional entropy of the current trajectory w.r.t. the trajectory archive.

3 Ev-ITER Overview

This section presents the new EvITE scheme, inspired from [3,12] and hybridizing the evolutionary and the learning approaches in order to build an autonomous exploratory robotic controller. EvITE involves two phases. In the first phase, controllers are evolved as in [3]. In the second phase, the trajectories generated by the evolved controllers are used to initialize an entropic state-action value function. This value function, characterizing the promising state-action pairs, is used to support an information-driven controller, and the entropic value function is accordingly updated. A main issue, scalability-wise, is that the discretization of the state and action space be under the control of the robot, depending on its memory and computational resources.

3.1 Phase 1. Evolutionary Exploration

For the sake of self-containedness, let us remind the formal background of the curiosity-driven evolutionary robotics approach [3]. Let $K = (s_t, a_t)_{t=1}^{T}$ denote a T-length trajectory, where s_t (respectively a_t) denotes the sensor (resp. motor) value vector at time t ($s_t \in \mathbb{R}^s$, $a_t \in \mathbb{R}^m$). An ϵ-clustering algorithm is used to incrementally cluster the sensor and the motor spaces independently[1] [4]. To each cluster c is associated the fraction p_c of time spent in this cluster (number of times s_t belongs to c, divided by T) and the entropic fitness is defined by Eq. (2).

After a number N of generations, the best controllers in the last population are retained, and the set $\mathcal{K}$ of trajectories they generated is used to initialize the entropic value function.

3.2 Phase 2: Initializing the Entropic Value Function

EvITE starts by discretizing the state and action space. Mainstream clustering algorithms, the k-means and ϵ-clustering algorithms [4] have complementary strengths and weaknesses: while ϵ-clusters are imbalanced, k-means clustering is sensitive to the k hyper-parameter, and might additionally yield unstable clusters. For this reason, we proceed as follows. Let n_s denote the number of ϵ-clusters built from the sensor vectors s_t in the archive $\mathcal{K}$. Setting the number k of clusters to n_s, P independent runs of k-means clustering algorithm are launched on the set of sensor vectors, and the best clustering after the distortion criterion is retained, where the distortion is measured from the sum of square distance between any vector and the center of the cluster it belongs to. Likewise, letting n_a denote the number of ϵ-clusters built from the motor vectors a_t in the archive $\mathcal{K}$, n_a clusters are obtained using the best clustering solution obtained out of P independent runs of k-means with $k = n_a$ on the motor vectors in $\mathcal{K}$.

Let $i(s_t)$ (respectively $j(a_t)$) denote the index of the cluster the sensor vector s_t (resp. the motor vector a_t) belongs to. For the sake of notational simplicity,

[1] An ϵ-clustering algorithm iteratively constructs a set of clusters, by defining a new cluster for each point which is more than ϵ-far from the center of the previous clusters.

it is assumed in the following that $\mathcal{K}$ involves a single trajectory; the robot is said to execute action j in state i when $i(s_t) = i$ and $j(a_t) = j$.

To each pair i, j is associated the list $Z(i, j)$ storing all instants t following the execution of action j in state i (such that $i(s_{t-1}) = i; j(a_{t-1}) = j$). Let $S(i, j)$ denote the multi-set of states for t in $Z(i, j)$, that is the list of all states that the robot visited just after executing action j in state i. Let $Q(i, j)$ denote the entropy of $S(i, j)$: the higher $Q(i, j)$, the less one can predict the behavior of the robot after selecting action j in state i. $Z(i, j)$ and $S(i, j)$ are built and

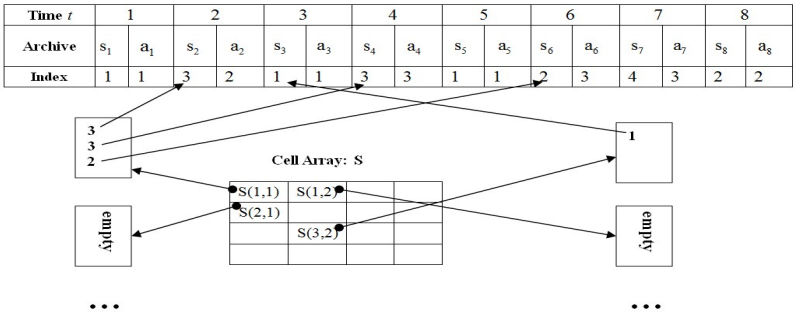


Fig. 1. Computing the entropic value function from a 8-length trajectory (top), with $n_s = n_a = 4$. The 4×4 matrix S is built, where list $S(i, j)$ is used to compute entropy $Q(i, j)$ when not empty.

maintained online (Fig. 1). A sliding window is used to comply with the robot limited memory resources, where only the last λ elements in $Z(i, j)$ and $S(i, j)$ are retained.

3.3 Phase 2. Information-Driven Navigation

Based on the entropic value function, an information-driven controller is defined as follows, considering three modes.

In the first mode, referred to as **babbling mode** and triggered when the robot is visiting a state i where few actions have been tried (the size of $Z(i, j)$ is less than λ, for all actions j), the action cluster index is selected uniformly in $1 \ldots n_a$.

The second mode is the main mode, referred to as **curious mode**. In this mode, the action with optimal score is selected, where

$$score(j|i) = (1 - \alpha)Q(i, j) + \alpha(1 - Pr(i|(i, j))) \tag{3}$$

where the last term $Pr(i|(i, j))$ is the probability of staying in state i after selecting action j in state i (estimated from the frequency of $i(s_{t-1}) = i, j(a_{t-1}) = j, i(s_t) = i$ for t in $Z(i, j)$), and α a hyper-parameter controlling the balance between the former and the latter terms. The intuition is that, everything being equal, exploration is better enforced by selecting an action that modifies the robot state.

The third mode, referred to as **exploratory mode**, is meant to prevent the degenerate behaviors possibly incurred in the *curious mode* (e.g. dancing in front of a wall). Here, one simply selects the action less selectedthe last μ times state i was visited.

In all three modes, letting j denote the index of the selected action cluster at time t, then the actual motor vector a_t is selected uniformly in the j-th action cluster.

Overall, the babbling mode is triggered when arriving in an unknown state. Otherwise, the curious mode is selected with probability 95% and the exploratory mode is selected with probability 5%. In each time step, the $Z(i,j)$ and $S(i,j)$ lists are updated. Formally, after having selected action j in state i at time t, the last elements in $Z(i,j)$ and $S(i,j)$ are removed if necessary (if $|Z(i,j)| > \lambda$) and indices $t+1$ and $k = i(s_{t+1})$ are respectively added to $Z(i,j)$ and $S(i,j)$.

4 Experimental Validation

This section reports on the experimental validation of EvITE. The primary goal of the experiments is to assess the performance of the proposed scheme in terms of the exploration indicators below, comparatively to the evolutionary curiosity-driven scheme [3] and the intrinsic motivation scheme [12] standalone[2]. Another goal is to assess the generality of the resulting controllers, i.e. their ability to explore new arenas, distinct from the one used in the first phase of EvITE.

For the sake of reproducibility, the experimental setting uses the Webot simulator emulating an E-puck robot with 8 infra-red sensors and 2 motors. The evolutionary curiosity baseline is implemented using same setting as in [3]: the robot population is evolved for 200 generations. The 1st phase in EvITE uses the trajectories of the best robots in the 200-th generation of Curiosity to build the $Z(i,j)$ and $S(i,j)$ data. The intrinsic motivation baseline proceeds using an empty $Z(i,j)$ and $S(i,j)$, based on the fact that $Q(i,j)$ is a proxy for the accuracy of the forward model in state i,j (Eq. 1). The hyper-parameter setting is as follows. Window lengths λ and μ are respectively set to 500 and 60. The ϵ parameter used for respectively clustering the sensor and motor vectors is set to 450 and 3000. Each EvITE and IM run considers a sequence of 2,000 epochs, where an epoch is a robot starting from the same starting point (lower left corner in the arenas, Fig. 2) and navigating in the arena for 2,000 time steps (175 time steps are required to cross the arena at average speed). Three arenas are considered: the easy one is taken from [9,3] (referred to as hard arena in [9], Fig 2.a); a harder one referred to as graph-arena (Fig 2.b); and a maze-like arena (Fig 2.c). The exploratory performance of Curiosity, IM and EvITE is comparatively displayed on the easy (Fig. 2. d), maze (Fig. 2.e) and maze (Fig. 2.f) arenas, showing the number of squares p_1 visited at least once per run (averaged

[2] The comparison with other ER approaches, specifically [20,22], faces the difficulty of defining a fitness function: building a fitness function from the exploration indicators is inappropriate since computing them requires some ground truth; but the whole motivation of the approach is to handle cases where the ground truth is not available.

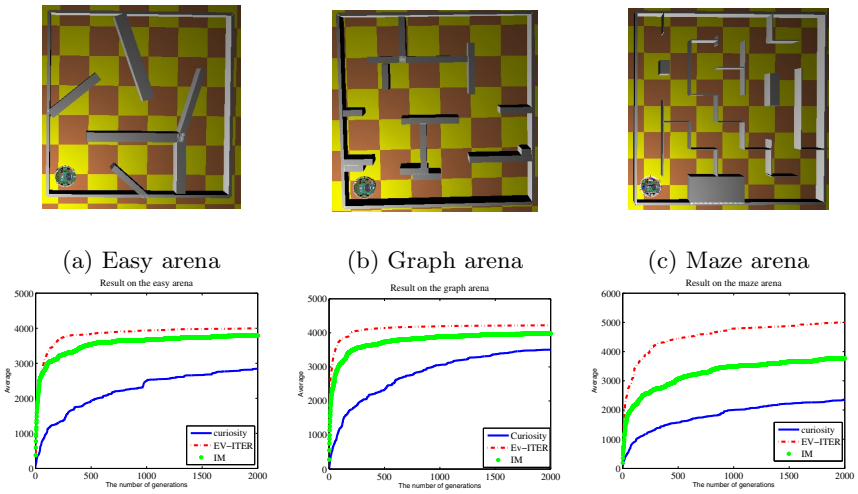


Fig. 2. Comparative performances of Curiosity, IM and EvITE on the (a) easy arena, (b) graph arena and (c) maze arena (respectively 0.6 m ×0.6 m, 0.6 m ×0.6 m, and 0.7 m ×0.7 m). The number of squares visited at least once (each arena involving 10,000 squares for comparison) is reported, averaged out of 15 independent runs. It is reminded that EvITE entropic value function is initialized from the trajectories on the Easy arena, and the curiosity-driven controllers are trained on the Easy arena too.

out of 15 independent runs, 10,000 squares for each arena). Furthermore, this result supports the generality of the EvITE approach, by noting that the EvITE controller performs well on the graph and maze arenas, even though the entropic value function is initialized by training on the easy arena. Comparatively, the curiosity-driven controllers trained on the easy arena do poorly on the other arenas. The performance of EvITE is significantly higher than for the intrinsic motivation alone, which itself outperforms the curiosity-driven approach, all the more so as the complexity of the arena increases. A more detailed account of the performance indicators p_1 and p_2 is reported in Table 1, indicating the average and median number of squares visited once or twice, averaged on 2,000 epochs and 15 runs. The generality is visually assessed on Fig. 3, showing the actual robot trajectories after 500 and 2,000 epochs. The top (resp. middle, bottom) rows display the behavior of Curiosity (resp. EvITE, IM), showing the arena visited during the first 500 epochs and the area visited during all 2,000 epochs, on the easy arena (columns 1 and 2), on the graph arena (3rd column) and on the maze arena (4th column). It is seen that Curiosity is lagging behind the other two approaches in all cases. On the easy and medium arenas, the performances of IM are visually a bit behind those of EvITE for 500 epochs (complementary results omitted due to space restrictions), and they catch up for 2,000 epochs.

Table 1. Comparative performances of EvITE, IM and Curiosity on the easy, graph and maze arenas, reporting the average and median (std.dev.) number of squares visited out of 15 runs

		IM		Curiosity		Ev-ITER	
		1 visits	2 visits	1 visits	2 visits	1 visits	2 visits
Easy arena	Median	3760	1898	2884	2106	4105	2514
	Average (std.dev.)	3796.7 (354.21)	1921.9 (7.5607)	2843.1 (198.9906)	2104 (134.8306)	3999.1 (236.0734)	2501.3 (219.4420)
Graph arena	Median	3693	2123	3232	1960	3995	1774
	Average std.dev.	3974.8 (264.81)	2136.4 (0.11499)	3510.1 (186.6)	1942.1 (24.56)	4222.6 (80.479)	1825.1 (43.036)
Maze arena	Median	3614	1831	2102	1251	4553	2050
	Average std.dev.	3779 (455.31)	1788.2 (155.96)	2362.5 (192.91)	1279.9 (19.21)	5005.5 (245.86)	2082.4 (158.95)

(a) Curiosity:500

(b) Curiosity: 2000

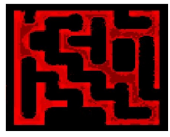
(c) Curiosity: 2000

(d) Curiosity: 2000

(e) Ev-ITER:500

(f) Ev-ITER: 2000

(g) Ev-ITER: 2000

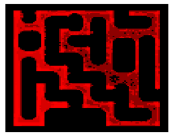


(h) Ev-ITER: 2000

(i) IM:500

(j) IM: 2000

(k) IM:2000



(l) IM:2000

Fig. 3. Trajectories of the evolutionary curiosity (top row), EvITE (middle row) and IM (bottom row) on the easy, graph and maze arenas, cumulative over 500 robots and 2,000 robots

On the maze arena finally, the performances of IM are behind those of EvITE for both 500 and 2,000 epochs (see the middle corridors in the maze).

These results establish the merits of the hybrid EvITE approach. On the one hand, it significantly improves on the evolutionary curiosity approach alone. On the other hand, it also improves on the intrinsic motivation approach, as it is shown to visit more densely the regions far from the starting point. Last but not least, the EvITE controller features a quite decent generality as it reaches good

exploratory performances when the 2nd phase is conducted in another arena than the one considered in the 1st phase. Complementary experiments, omitted due to the space restrictions, show that the exploratory performance is almost as good when the 1st phase is conducted in the easy arena, than when it is conducted in the same arena as the one where the 2nd phase takes place.

5 Discussion and Perspectives

This paper presents a new combination of EC and ML approaches toward autonomous exploration in in-situ robotics. The challenge of defining intrinsic criteria available on-board is addressed by taking inspiration from [3,12]. On the top of these pioneering works, the EvITE scheme proceeds as a 2-phase process. The first phase proceeds exactly as in [3].The second phase exploits the data gathered in the first phase in order to support a learning approach. The impact of these data is evidenced by comparing the EvITE performances to that of IM. The experimental results (Fig. 2) suggest that the additional information provided to the EvITE controller is never compensated for by the IM controller: the IM performance plateaus well below the EvITE performance.

These results suggest that one strength of the evolutionary approach is to be able to gather relevant data in a fast and efficient manner; once these data have been acquired, then deterministic ML might be in better shape to exploit them thoroughly. The originality of this coupling is to use evolution to provide informative data to ML, whereas the mainstream hybridation scheme uses evolution to optimize the ML solution.

It is worth emphasizing the fact that the EvITE controller can yield good performances also in new arenas, thus featuring some generality property, although admittedly the arenas used in the 2nd phase are not very different from the one used in the 1st phase.

Still the possibility of using different environments in the 1st and 2nd phases opens interesting perspectives, particularly in terms of *safe* robotic training. For the sake of a safe in-situ robot training, it makes sense indeed to consider a simple and dangerless arena to acquire the initial data, and to launch the robot in a new and possibly more hazardous arena, using the data as prior knowledge to prevent or mitigate the dangers of blind exploration. Another perspective for further study is to incrementally revise and specialise the sensor and action clusters considered by EvITE, along the same lines as in [12]. A third perspective is to involve the user in the loop along an interactive optimization setting [1], with the goal of achieving other target behaviors on the top of the exploratory behavior.

References

1. Akrour, R., Schoenauer, M., Sebag, M.: Preference-based policy learning. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part I. LNCS, vol. 6911, pp. 12–27. Springer, Heidelberg (2011)

2. Baranès, A., Oudeyer, P.Y.: R-iac: Robust intrinsically motivated exploration and active learning. IEEE Transactions on Autonomous Mental Development 1(3), 155–169 (2009)
3. Delarboulas, P., Schoenauer, M., Sebag, M.: Open-ended evolutionary robotics: an information theoretic approach. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 334–343. Springer, Heidelberg (2010)
4. Duda, P.O., Hart, P.E.: Pattern Classification and Scene analysis. John Wiley and sons (1973)
5. Heidrich-Meisner, V., Igel, C.: Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In: Int. Conf. on Machine Learning, pp. 401–408 (2009)
6. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The Int. Jal of Robotics Research 32(11), 1238–1274 (2013)
7. Koos, S., Mouret, J.B., Doncieux, S.: The transferability approach: Crossing the reality gap in evolutionary robotics. IEEE Trans. on Evolutionary Computation 17(1), 122–145 (2013)
8. Lehman, J., Risi, S., D'Ambrosio, D.B., Stanley, K.O.: Rewarding reactivity to evolve robust controllers without multiple trials or noise. Artificial Life 13, 379–386 (2012)
9. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. Artificial Life 11, 329 (2008)
10. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic lifeforms. Nature 406(6799), 974–978 (2000)
11. Liu, W., Winfield, A.F.: Modeling and optimization of adaptive foraging in swarm robotic systems. The Int. Jal of Robotics Research 29(14), 1743–1760 (2010)
12. Lopes, M., Lang, T., Toussaint, M., Oudeyer, P.-Y.: Exploration in Model-based Reinforcement Learning by Empirically Estimating Learning Progress. In: NIPS, pp. 206–214 (2012)
13. Mouret, J.B., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: An empirical study. Evolutionary Computation 20(1), 91–133 (2012)
14. Nolfi, S., Floreano, D., Floreano, D.: Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. MIT Press, Cambridge (2000)
15. Oudeyer, P.Y., Kaplan, F., Hafner, V.V.: Intrinsic motivation systems for autonomous mental development. IEEE Trans. on Evolutionary Computation 11(2), 265–286 (2007)
16. Pfeifer, R., Gomez, G.: Interacting with the real world: design principles for intelligent systems. Artificial life and Robotics 9(1), 1–6 (2005)
17. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. The Int. Jal of Robotics Research 27(2), 157–173 (2008)
18. Sutton, R., Barto, A.: Reinforcement learning: An introduction. Cambridge Univ. Press (1998)
19. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
20. Williams, H., Browne, W.N.: Integration of Learning Classifier Systems with simultaneous localisation and mapping for autonomous robotics. In: CEC 2012, pp. 1–8 (2012)
21. Hurst, J., Bull, L., Melhuish, C.: TCS learning classifier system controller on a real robot. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 588–597. Springer, Heidelberg (2002)
22. Koutnk, J., Cuccu, G., Schmidhuber, J.: Evolving large-scale neural networks for vision-based reinforcement learning. In: GECCO 2013, pp. 1061–1068 (2013)