Exploratory Robotic Controllers : An Evolution and Information Theory Driven Approach

Guohua Zhang

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE d'Informatique, ED 427
Laboratoire de Recherche en Informatique

Informatique

THÈSE DE DOCTORAT

soutenue le 24/09/2015

par

Guohua ZHANG

Exploratory Robotic Controllers: An Evolution and Information Theory Driven Approach

Directrice de thèse :	Michèle Sebag	DR CNRS, Université Paris-Sud, LRI/TAO
Co-encadrant de thèse :	Jingzhong Zhang	Professeur , Institut d'informatique appliquée Académie des Sciences de Chine, Chengdu

Composition du jury :

Rapporteurs :	Jin-Kao Hao	Professeur, Université d'Angers, LERIA
	Wensheng Zhang	Professeur, Institut d'automatisation, Académie des Sciences de Chine
Examinateurs :	Hao Li	DR CNRS, Université Paris-Sud,GALaC
	Xiaolin Qin	Matîre de Conférence, Institut d'informatique appliquée, Académie des Sciences de Chine, Chengdu
Directrice de thèse :	Michèle Sebag	DR CNRS, Université Paris-Sud, LRI/TAO
Member Invité:	Jingzhong Zhang	Professeur, Institut d'informatique appliquée, Académie des Sciences de Chine, Chengdu

UNIVERSITÉ PARIS-SUD 11

École Doctorale D'informatique, ED 427

P H D T H E S I S

to obtain the title of

PhD of Science

of the Université Paris-Sud 11

Specialty : Computer Science

Defended by

Guohua ZHANG

Exploratory Robotic Controllers: An Evolution and Information Theory Driven Approach

Thesis Advisor: Michèle SEBAG and Jingzhong ZHANG

TAO Team
defended on 24 September , 2015

Jury :

Reviewers :	Wensheng Zhang	-	Institute of Automation Chinese Academy of Sciences, China
	Jin-Kao Hao	-	Université de Angers, LERIA, France
Examinators :	Hao li	-	Université Paris-Sud
	Xiaolin qin	-	Chengdu Institute of Computer Applications, Chinese Academy of Sciences, China
Advisor :	Michèle sebag	-	Université Paris-Sud
Invited :	Jingzhong Zhang	-	Chengdu Institute of Computer Applications, Chinese Academy of Sciences, China

EXPLORATION ROBOTIQUE AUTONOME HYBRIDANT ÉVOLUTION ET THÉORIE DE L'INFORMATION

Thése de doctorat

École Doctorale d'Informatique, ED 427

Université Paris-Sud 11

Par **Guohua ZHANG**

Présentée et soutenue publiquement

Le 24 septembre 2014

Directrice de thèse:	Michèle Sebag	Université Paris Sud
Co-encadrant de thèse:	Jingzhong zhang	Institut d'informatique appliquée Académie des Sciences de Chine, Chengdu

Devant le jury ci-dessous :

Hao li,	DR CNRS, Université Paris Sud	(Examinateur)
Xiaolin qin,	Matîre de Conférence, Institut d'informatique appliquée Académie des Sciences de Chine, Chengdu	(Examinateur)
Jin-Kao Hao,	Professeur, Université d'Angers, LERIA	(Rapporteur)
Wensheng zhang,	Professeur, Institut d'automatisation Académie des Sciences de Chine	(Rapporteur)
Michèle Sebag,	DR CNRS, Université Paris-Sud, LRI/TAO	(Directrice de Thèse)
Jingzhong zhang,	Professeur, Institut d'informatique appliquée Académie des Sciences de Chine, Chengdu	(Invité)

Rapporteurs :

Jin-Kao Hao,	Professeur, Université de Angers, LERIA, France
Wensheng zhang,	Professeur, Institut d'automatisation, Académie des Sciences de Chine

Abstract

This thesis is concerned with building autonomous exploratory robotic controllers in an online, on-board approach, with no requirement for ground truth or human intervention in the experimental setting.

This study is primarily motivated by autonomous robotics, specifically autonomous robot swarms. In this context, one faces two difficulties. Firstly, standard simulator-based approaches are hardly effective due to computational efficiency and accuracy reasons. On the one hand, the simulator accuracy is hindered by the variability of the hardware; on the other hand, this approach faces a super-linear computational complexity w.r.t. the number of robots in the swarm. Secondly, the standard goal-driven approach used for controller design does not apply as there is no explicit objective function at the individual level, since the objective is defined at the swarm level.

A first step toward autonomous exploratory controllers is proposed in the thesis. The *Evolution & Information Theory-based Exploratory Robotics* (Ev-ITER) approach is based on the hybridization of two approaches stemming from Evolutionary Robotics and from Reinforcement Learning, with the goal of getting the best of both worlds: (i) primary controllers, or crawling controllers, are evolved in order to generate sensori-motor trajectories with high entropy; (ii) the data repository built from the crawling controllers is exploited, providing prior knowledge to secondary controllers, inspired from the intrinsic robust motivation setting and achieving the thorough exploration of the environment.

The contributions of the thesis are threefold. Firstly, Ev-ITER fulfills the desired requirement: it runs online, on-board and without requiring any ground truth or support. Secondly, Ev-ITER outperforms both the evolutionary and the information theory-based approaches standalone, in terms of actual exploration of the arena. Thirdly and most importantly, the Ev-ITER controller features some generality property, being able to efficiently explore other arenas than the one considered during the first evolutionary phase. It must be emphasized that the generality of the learned controller with respect to the considered environment has rarely been considered, neither in the reinforcement learning, nor in evolutionary robotics.

Résumé en Français

Cette thèse porte sur la conception de contrôleurs pour robots explorateurs autonomes basée sur une approche en ligne (online) intégrée, ne nécessitant pas de vérité terrain ni d'intervention de l'expert humain au cours du processus d'entrainement.

Le travail présenté se focalise sur le domaine de la robotique autonome et plus particulièrement la conception de controleurs robotiques pour les essaims de robots.

Ce contexte présente deux difficultés spécifiques. Premièrement, les approches basées sur l'usage de simulateur sont d'efficacité limitée: d'une part, la précision du simulateur est limitée compte tenu de la variabilité des robots élémentaires; d'autre part, la complexité de la simulation est super-linéaire en fonction du nombre de robots de l'essaim. Deuxièmement, les approches guidées par le but se heurtent au fait que la fonction objectif n'est pas définie au niveau du robot individuel, mais au niveau de l'essaim.

Une première étape vers la conception de contrôleur explorateur autonome est proposée dans cette thèse. L'approche proposée, appelée exploration robotique fondée sur l'évolution et l'information (Ev-ITER) se fonde sur l'hybridation de la robotique évolutionnaire et de l'apprentissage par renforcement utilisant l'entropie. Cette approche procède en deux phases: (i) dans une première phase l'évolution artificielle est utilisée pour générer des contrôleurs primaires (crawlers), dont les trajectoires sont d'entropie élevée dans l'espace sensori-moteur; (ii) dans une seconde phase, l'archive des trajectoires acquises par les controleurs primaires est exploitée pour définir les controleurs secondaires, inspirés de la motivation intrinsèque robuste et permettant l'exploration rigoureuse de l'environnement.

Les contributions de cette thèse sont les suivantes. Premièrement, comme désiré Ev-ITER peut être lancé en ligne, et sans nécessiter de vérité terrain ou d'assistance. Deuxièmement, Ev-ITER surpasse les approches autonomes en robotique évolutionnaire en terme d'exploration de l'arène. Troisièmement, le contrôleur Ev-ITER est doté d'une certaine généralité, dans la mesure où il est capable d'explorer efficacement d'autres arènes que celle considérée pendant la première phase de l'évolution. Il est à souligner que la généralité du contrôleur appris vis-á-vis de l'environnement d'entrainement a rarement été considérée en apprentissage par renforcement ou en robotique évolutionnaire.

Acknowledgements

It is a pleasure to acknowledge my supervisors and some of the colleagues and friends who have contributed to this dissertation.

First and foremost, I'm extremely grateful to my supervisors, Michèle Sebag and Jingzhong zhang. They not only provided excellent guidance and inspired me, but also helped to organize, refine and structure my ideas. I especially thank Michèle for teaching me hand by hand. She taught me, how important it is to present your ideas well, and how to do so. Without her help and dedicated assistant throughout my three years' Phd. time , this dissertation would have never been accomplished. I am deeply thankful for all their advice, patience and help.

I deeply thank the members of the jury who took some of their precious time for my thesis. I add here a particular acknowledgement to the reviewers for providing meaningful feedback. Their comments and suggestions are very constructive for improving this dissertation.

I would like to thank TAO team leader Marc Schoenauer, my kind colleagues (in alphabetical order): Antoine, Anne, Asma, Basile, Dawei, Guillaume, Jialin, Nikolaus, Nicolas, Marie-Carol, Marie-Liesse, Mostepha, Mouadh, Mustafa, Ouassim, Olga, Thomas, Riad, Weijia, yifan, etc. They helped me on both my research work and my life in these years. I am especially grateful to Dawei, Riad and Weijia for sharing their experience and for many fruitful discussions we had.

I also thank all my new and old closest friends: Yangbin Tang, Weihua Yang, Weihua He, Jihong Yu, Jiuqiang Chen, Cong Zeng, Jingyi Bin, Chen Wang, Jianqiang Cheng, Guanyu Li, Kai Yang, Alessandro Leite, Chuan Xu, Qiang Sun and Yandong Bai. The lovely time we spent together will forever stay in my memory.

I should also mention that my graduate studies in France were supported by the China Scholarship Council.

Most importantly, this dissertation is dedicated to my family, especially my parents. Nothing would have come true without their support. All I have and will accomplish are only possible because of their love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is concerned with the building of autonomous exploratory robotic controllers in an *in-situ* manner, where the learning and optimization of the controller takes place on the robot itself, as opposed to, on a simulation platform or *in-silico*. Quite a few disciplinary fields are relevant to autonomous robotics, ranging from optimal control [Zhou *et al.* 1996] to artificial intelligence (AI) [Pfeifer & Gomez 2005], evolutionary robotics (ER) [Nolfi & Floreano 2000] and machine learning (ML) (specifically reinforcement learning (RL) [Sutton & Barto 1998, Duda *et al.* 2012]. This thesis is the cross-road of evolutionary robotics (section 1.2) and reinforcement learning (section 1.3). Let us first present the research questions investigated in the presented work.

1.1 Research Background

This study is primarily motivated by autonomous robotics, specifically autonomous robot swarms (Fig. 1.1), taking inspiration from the SYMBRION European project (European Integrated Project 216342, 2008−2013). Autonomous robot swarms aim at designing robust, scalable and flexible collective behaviors, where large numbers of robots are coordinated through simple controllers and local interactions [Brambilla *et al.* 2013, Arvin *et al.* 2014]. The autonomy of the individual robot is an essential characteristics of swarms [Brambilla *et al.* 2013]. In this context, one faces two difficulties:

Firstly, the standard simulator-based approach is ineffective. On the one hand, the computational complexity is super-linear with respect to the number of robots in the swarm; on the other hand, the simulator accuracy is challenged by the variability of the hardware; controllers learned or optimized in simulation are prone to the so-called *reality gap*, meaning that the optimal behavior *in-silico* does not translate into an efficient behavior *in-situ*.

Secondly, the standard goal-driven approach used for controller design does not apply as there is no explicit objective function. More specifically, the objective is defined in terms of the swarm behavior whereas the design concerns the individual robot controller.

(a) Individual robot (b) Individual robot (c) A swarm robotic system

Figure 1.1: Examples of existing robots (a) Individual robot: RobotCub, complex child humanoid robot; (b) Individual robot: e-puck, relatively simple individual robot; (c) the swarm robotic system: SwarmBots. Adapted from [Lenaertz 2012].

The research question investigated in this manuscript concerns how to define rewards, that is, incentives guiding the individual robot behavior in the swarm context. The proposed approach builds upon previous work in evolutionary robotics and reinforcement learning [Delarboulas *et al.* 2010,Akrour *et al.* 2014,Lopes *et al.* 2012], showing the feasibility of defining internal and external rewards under the requirements of autonomous, ground truth-less settings.

1.2 Evolutionary Robotics

Evolutionary robotics [Nolfi & Floreano 2000] (chapter 2) is a field in which Evolutionary Computation (EC) is applied to the design of both real and simulated autonomous robots.

The bulk of research in ER concerns simulated robots, for the sake of computational and experimental conveniency. On the computational side, EC is known for requiring a huge number of evaluations in order to yield good results; but the time and effort required to conduct thousands of robotic evaluations *in-situ*, is overwhelming. On the experimental side, evaluating preliminary controllers (and many controllers) entails safety hazards for the robot itself [Koos *et al.* 2013]. The dark side of evolutionary robotics in simulation is the so-called *reality gap* problem, already mentioned [Jakobi *et al.* 1995,Lipson *et al.* 2006]: controllers evolved in simulation often perform much worse on the real robot, e.g. biped walking gaits evolved in simulation cannot run efficiently in the real world [Boeing *et al.* 2004].

Some work in online on-board evolutionary robotics have been conducted to achieve obstacle avoidance and object attraction [Nordin & Banzhaf 1997], obstacle avoidance based on vision [Marocco & Floreano 2002], gait learning in a quadruped

robot [Hornby *et al.* 2000a], and/or to overcome the reality gap and/or to adapt to robotic failures [Lipson *et al.* 2006]. A common feature of on-line on-board ER is to require considerable care and efforts from the human designer. The key question regards the assessment of the robot behavior (the optimization objective): to which extent can this assessment be done in an autonomous way.

1.3 Machine Learning

Reinforcement learning [Sutton & Barto 1998] (chapter 3) is the field of Machine Learning interested in learning and optimizing policies, or equivalently controllers, associating to each state an action in order for the learning agent (the robot, here) to maximize the rewards gathered by its behavior. Reinforcement learning is known to be a hard problem, due to a mixture of fundamental, algorithmic and practical issues. Many of these issues are manifested in the robotics setting [Kober & Peters 2012, Kormushev *et al.* 2013]. The two main difficulties are related to the Markov Decision Process framework at the core of RL, which does not always reflect the real-world context; another difficulty is to define a good reward function within the MDP setting, conducive to the desired behaviors.

The difficulties of reward design have motivated the design of a number of approaches, concerned with implicit or unknown rewards. For example, Inverse Reinforcement Learning (IRL) [Ng *et al.* 2000] learns the reward function from the demonstrations of an expert. In Preference-based Reinforcement Learning (PBRL) [Wirth & Fürnkranz 2013c, Akrour 2014], the reward function is learned based on the expert feedback about the robot behaviors. While these approaches all relax the expertise requirement from the human designer, they still require her intervention in the learning or optimization loop. A new setting, referred to as *intrinsic motivation* [Baranès & Oudeyer 2009, Oudeyer *et al.* 2012], proposes that rewards be built-in and autonomously measured by the agent itself along its trajectory, akin a computational "instinct".

1.4 Main Contributions

The presented work is concerned with building exploratory robotic controllers in an *in-situ* approach, addressing the challenge of defining intrinsic rewards without any ground truth about the appropriateness of the robot behavior in its environment. The main contributions are as follows:

1. A hybrid two-phase Evolution and Information Theory-Driven Exploratory Robotics (Ev-ITER) approach is proposed, combining machine learning and

evolutionary principles. Formally, Ev-ITER-1st phase builds primary controllers, referred to as crawling controllers, using Evolutionary Robotics by taking inspiration from the information theory-based approach presented in [Delarboulas *et al.* 2010]; additionally, this information-theory based approach is extended to take into account the entropy of the actuators. The crawling controllers gather a data repository, related to the trajectories in a first source environment. In the second phase, this data repository is used to support an information theory-based controller, selecting the most informative action in each time step. Further, this Ev-ITER scheme is shown to outperform both the evolutionary and the information theory-based approaches standalone, in terms of actual exploration of the arena.

2. The Ev-ITER approach is designed to run online, on-board with no ground truth and no human intervention, thus avoiding the reality gap; in contrast, many existing autonomous robotic algorithms [Lehman & Stanley 2008, Williams & Browne 2012, Koutník *et al.* 2013, Koutník *et al.* 2014] involve some ground truth information in order to compute the exploration indicators (e.g. when applied for simultaneous localization and mapping in [Williams & Browne 2012]).

3. Lastly, and most importantly, the Ev-ITER controller features some generality property w.r.t. the robotic environments. The exploration efficiency is also observed when the Ev-ITER controller is launched in a target environment, different from the source environment considered by the crawling controllers. This property of generality and robust exploration across environments is a most original contribution of the presented work.
   Its potential applications are manifold, typically when dealing with environments of different difficulty: a pre-training in the source environment would result in minimizing the exploration time needed to build a map of the target environment. Another expected benefit is to have the 1st-Phase taking place in simulation, while the 2nd-Phase takes place in-situ.

1.5 Thesis Outlines

The thesis manuscript is organized as follows:
Chapter 2 presents Evolutionary Robotics, more particularly focussing on algorithm deployment in-situ, and the *reality gap* issue. A second focus regards the design of intrinsic fitness functions that can be computed on the robot itself.
Chapter 3 presents some reinforcement learning approaches aimed at autonomous

robotics and discusses their strengths and weaknesses. Included is a discussion of the limitations of RL and policy learning with respect to exploratory robotics, a presentation of RL and policy learning with implicit rewards, a description of the notion of *intrinsic motivation* and *discovery* approaches.
Chapter 4 describes the algorithmic contribution of the thesis, the Ev-ITER algorithm, a new combination of Evolutionary Robotics and Reinforcement Learning approaches toward autonomous exploration in *in-situ* robotics. The generality of Ev-ITER is discussed.
Chapter 5 is devoted to the empirical validation of the proposed approach, considering different arenas. The main limitation of this work is that no actual experimentation *in-situ* were possible at the moment of writing the manuscript.
Chapter 6 concludes this Ph.D thesis by outlining some future avenues for research.

Chapter 2

Evolutionary Robotics

Evolutionary robotics (ER) aims to apply evolutionary computation techniques to the design of both real and simulated autonomous robots. In this chapter, we first present a brief general introduction to evolutionary computation, with particular focus on the Evolution Strategy [Rechenberg 1973]. We thereafter review different categories of fitness functions used in the field of ER. Then the challenge of transferring controllers obtained through simulation to real robots, known as the *reality gap*, is discussed. Finally, fitnesses that can be computed on the robot itself (on-board) are presented; these enable the use of evolutionary computation algorithms in-situ, thereby sidestepping the reality gap issue.

2.1 A Brief Introduction to Evolutionary Computation

Evolutionary Computation (EC) [Fogel 2006] uses computational models of evolutionary processes as key inspiration in the design and implementation of computer-based problem solving systems. There are a variety of evolutionary computational modes that have been proposed and studied, which we will refer to as evolutionary algorithms (EAs) [Back *et al.* 2008]. Thus, the term EAs is frequently used interchangeably with EC systems in the literature. These EC algorithms share the common background of being remotely inspired from Darwins's principles of natural selection and blind variations thereof [Darwin 1859], where individuals are competing with each other for survival and reproduction in an environment that can only host a limited number of individuals [Eiben & Smith 2003]. Although simplistic from a biologist's viewpoint, these EC algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

From a practical point of view, EC algorithms are population-based metaheuristics that provide the human engineer with a set of tools to address particular optimization problems. The core principles are built upon two complementary mechanisms, inspired from Darwin's original principles: blind variations and survival of the fittest. Fig 2.1 (proposed by Zhang et al. in [Zhang *et al.* 2011]) describes the general framework with three fundamental operators (in bisque in figure) and two optional operators (in yellow in figure) for most EC algorithms. The basic EC

algorithm involves 3 steps: *'population initialization'*, *'fitness evaluation and selection'*, and *'population reproduction and variation'*. Besides the above three necessary steps, EC algorithms sometimes additionally perform an *'algorithm adaptation'* or a *'local search'* (LS) procedure. EC algorithms involving LS are known as memetic algorithms [Ong *et al.* 2010].

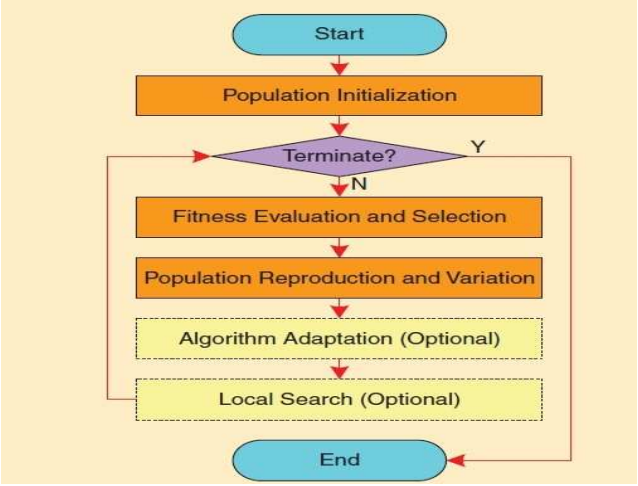


Figure 2.1: The general EC framework. Evolutionary Process: starting from a population of randomly generated individuals, each individual is evaluated and associated a measure of performance. Individuals are thereafter selected depending on their performance. The selected individuals go through the variation process (mutation and recombination), thus generating a new population. The new population is then evaluated again and the iteration continues until a termination criterion is met.

In particular, although different EC algorithms have a similar framework in implementation and algorithmic characteristics, their particular implementations differ in many details. A main difference regards the representation of the individuals, usually dictated by the target application problem; the various representations which have been proposed include bit-strings, real-valued vectors, Lisp expressions, and neural networks. Another difference regards the relative importance of mutation and crossover (recombination), as well as their particular implementation, which differ widely across EC algorithms. Finally, the stopping criterion is problem-dependent.

The origins of EA can be traced back to at least the 1950s, and since the 1970s several evolutionary methodologies have been proposed, including evolutionary programming (EP), evolution strategies (ESs), genetic algorithms (GAs), genetic programming (GP), and differential evolution (DE). A more detailed description will

be provided for ESs, as this evolutionary algorithm will be used in the experimental section of this manuscript. While all five paradigms rely on similar concepts, they are applied to different types of problems.

2.1.1 Evolutionary Programming

Evolutionary programming (EP) was originally applied to the evolution of finite state automata for machine learning problems [Fogel *et al.* 1966]. Traditionally, EP has used representations that are tailored to the problem domain. For example, in real-valued optimization problems, the individuals within the population are real-valued vectors. Successful applications of this approach are shown in robot navigation [Kim *et al.* 2001] and in robot hand manipulation problems [Fukuda *et al.* 1999].

2.1.2 Genetic Algorithms

Genetic algorithms (GAs) [Goldberg & Holland 1988] are often concerned with solving combinatorial optimization problems. Solutions are represented in binary as strings of 1s and 0s, but other encodings are also possible, such as graphs, Lisp expressions, and real-valued vectors. GA has a good application value in the design of robotics controllers. For example, GA is used to obtain an automatic design of the type-2 non-singleton fuzzy-logic controller [Martínez-Soto *et al.* 2014] and to solve the inverse kinematics problem of a six-joint Stanford robotic manipulator under the constrain of minimizing the error at the end effector [Köker 2013].

2.1.3 Genetic Programming

Genetic programming (GP) [Koza 1992] is a method to *evolve computer programs* and can also be used in logical expressions. This sub-field is based on individuals represented as tree structures. Some Lisp-languages that naturally embody tree structures are frequently used with GP, although other function languages can also be adapted in order to do it. GP has been applied to the design of robotics controllers in multiple cases, for example: the application of GP to the evolution of robot morphology [Gregor *et al.* 2012], the design of a controller used in developing a fast gait for a quadruped robot [Seo *et al.* 2010], and the design of controllers used in multi- robot scenarios [Kala 2012].

2.1.4 Differential Evolution

Differential evolution (DE) [Storn & Price 1997, Price *et al.* 2006] is a more recent method proposed for global numerical optimization. Solutions are represented by

vectors of real-values. This approach can be used over a large number of optimization problems [Das & Suganthan 2011]. For example, DE is used as an evolutionary alternative method to automatically obtain robotic behaviors [Cruz-Álvarez *et al.* 2013], to enhance localization of mobile robots [Lisowski *et al.* 2011], and to solve a non-linear dynamic optimization problem on the structure-control design of a five-bar parallel robot [Villarreal-Cervantes *et al.* 2010].

2.2 Evolution Strategies

One of the major EC paradigms, Evolution Strategies (ESs) [Rechenberg 1978] are specifically designed for continuous optimization. Due to initial interest in hydrodynamic problems, ESs typically use real-valued vector representation [Spears *et al.* 1993]. In this context the main variation operator considered is the mutation. In ESs, mutations are mainly represented by Gaussian mutations. A parent x generates an offspring y as follows:

$$y - x + \sigma\mathcal{N}(0, C) \tag{2.1}$$

where σ denotes the step-size, $\mathcal{N}(0, C)$ denotes the standard multivariate normal variables with mean 0 and covariance matrix C.

The key point in ES algorithms is the adaptation of the parameters of the process, in particular the adaptation of the step-size and the adaptation of the covariance matrix.

2.2.1 Adaptation of the step-size

The adaptation of the critical step-size σ is different from one algorithm to another, and it is this specification that is used to differentiate the various ESs. It is important to have an adaptative step-size, because if the step-size is constant and too small w.r.t. the distance to the optimum, the new individual will be close to the parent and the progression will be slow. In the other case, if the step-size is constant and too large with regard to the distance to the optimum, the probability that the new individual might be better than its parent will be too small. We present two different rules for adaptation the step-size : One is one-fifth rule and the other is cumulative step-size adaptation.

- **One-fifth rule**

  The adaptation of the step-size σ proceeds in various ways. One simple well known approach is the one-fifth rule [Rechenberg 1973]: If more than 20% of mutated offspring lead to fitness improvements within the last N generations,

then σ value is multiplied by 1.22. If less than 20% of the offspring obtain better fitness, then the σ is divided by 1.22. This approach and the parameters are designed to be optimal on the sphere test function: $f(x) = \sum_{i=1}^{d} x_i^2; x \in \mathbb{R}^d$ [Michalewicz 1996], which is supposed to represent the typical fitness landscape for many optimization problems when sufficiently close to the optimum.

- **Cumulative step-size adaptation**

  The cumulative step-size adaptation (CSA) proposed in [Hansen & Ostermeier 1996,Hansen & Ostermeier 2001] is a well-known algorithm for choosing the step-size. The principle of this method is to compare the length of the path followed by the algorithm to the length of the path followed under a random selection. If the path followed by the algorithm is larger than the path under random selection then the step-size is increased. In the other case, the step-size is decreased.

2.2.2 Adaptation of the covariance matrix

The acknowledged best approach in continuous evolutionary evolution is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen & Ostermeier 1996, Hansen & Ostermeier 2001]. The rule used for adapting the step-size is the CSA mentioned above. The key point is that the CMA-ES updates a full covariance matrix for the sample distribution. We here only mention the rank-one update situation (Algo.1) [Hansen & Ostermeier 2001] Besides, there are more complex options, but this is the same idea. Consequently, CMA-ES learns all pairwise dependencies between all parameters.

Algorithm 1 Covariance Matrix Adaptation: Rank-one update

Initialize $m \in \mathbb{R}^n$, and $C = I$, set σ=1, learning rate $c_{cov} \approx \frac{2}{n^2}$
while not terminate
$x_i = m + \sigma y_i, y_i \sim \mathcal{N}_i(0, C)$,
$m \leftarrow m + \sigma y_w$, where $y_w = \sum_{i=1}^{\mu} w_i y_i : \lambda$
$C \leftarrow (1 - c_{cov})C + c_{cov}\mu_w \underbrace{y_w y_w^T}_{rank-one}$ where $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$

Besides, a multi-objective evolutionary algorithm is the Multi-Objective (MO)CAM-ES [Igel *et al.* 2007] based on CMA-ES : briefly, the same adaptation implemented by the original CMA-ES is used to adapt the mutation operator carried by each individual, whenever its application is successful, *i.e.,* whenever it succeeds in generating a fitter offspring.

The ESs algorithm has been applied to the design of robotics controllers in multiple contributions. Successful applications of this approach are shown in single [De Croon *et al.* 2013, Bredeche *et al.* 2010] and multi-robot problems [Schultz *et al.* 1996, Pessin *et al.* 2010].

2.3 EC and Robotics

Evolutionary computation techniques have been most widely and successfully applied to the robot design process. [Doncieux *et al.* 2011] reviewed the main techniques developed in the robotics field and then distinguished four use cases for the application of EC methods to the field of robotics, which include parameter tuning, evolutionary aided design, online evolutionary adaptation and automatic synthesis.

2.3.1 Parameter Tuning

Evolutionary algorithms are now mature tools for black-box optimization. As they do not impose any constraint on the objective function(s), they can be employed to tune some robot parameters the optimal value of which is not known and cannot be found neither by analytical method (i.e. method not known) nor by an exhaustive search (i.e. too many parameters). In this context, finding optimized parameters is the goal of parameter tuning and generally comes at the end of the design process. This has been used for example in the optimization parameters of PID controllers for a 6-DOF robot arm [Kwok & Sheng 1994], and the optimization of bio-inspired artificial intelligence systems [Floreano & Mattiussi 2008].

2.3.2 Evolutionary Aided Design

Using evolutionary algorithms as an analysis and exploration tool instead of optimization is a growing trend in the field of robotics. In this context, evolutionary computation methods are employed to explore the design space of the system and propose a variety of solutions to the experts, who can analyze the results in order to gain a deeper understanding of the system. The experts are then able to propose new solutions (whose parameters might be further tuned with EAs) in a further step. This approach is used for example in the design of UAV's controllers [Hauert *et al.* 2009] and in the friction stir welding problem [Bandaru *et al.* 2011]. Multi-objective evolutionary algorithms are a special kind of evolutionary algorithm designed to find the best trade-offs between multiple objectives [Deb *et al.* 2001, Zhou *et al.* 2011]. This type of algorithm has been used to find relations between design parameters in a process called *innovization* [Deb & Srinivasan 2006, Deb &

Srinivasan 2008]. This approach has been successfully employed to design motors [Deb & Srinivasan 2008] and controllers of a flapping-wing robot [Doncieux & Hamdaoui 2011].

2.3.3 Online Evolutionary Adaptation

Evolutionary algorithms are applied to the robotic field not only in an off-line manner but also in on-line manner. In this context, embodied evolution consists in using EA not only during the design step, but also during robot lifetime, in order to allow it to adapt on-line to drastically changing situations (in terms of environment or of robot features). Advantages of this approach include the ability to address a new class of problems (problems that require on-line learning), the parallelization of the adaptation (a direct consequence of the population-based search) and a natural way to address the reality gap (as design constraints enforce on-board algorithms). This online evolutionary adaptation is currently being explored from different perspectives, ranging from endowing robots with some kind of resilient capacity [Bongard *et al.* 2006] with regards to environmental changes, to adapting known evolutionary algorithms to perform online evolution for single robot or multiple robots [Watson *et al.* 1999] or to a addressing environment-driven evolutionary adaptation [Bredeche & Montanier 2010].

2.3.4 Automatic Synthesis

Evolutionary algorithms are employed not only to optimize the robot's controller but also to optimize the overall design, i.e. a mechatronic device and its control system can be also automatically designed at the same time by an EA. This approach was pioneered by [Sims 1994], which demonstrated how the morphology and the neural systems of artificial creatures can be generated automatically with an EA. This approach is used for example in the Golem project where the robot morphology and the robot controller are optimized simultaneously [Lipson & Pollack 2000]. Evolutionary Synthesis is one promising use of EA, the long term goal of which is to exploit robot features and the environment better than an engineer would do. However, due to its challenging goal, it is also the less mature use of ER as many issues remain to be studied.

2.4 Fitness Functions

Evolutionary algorithms aim at finding controllers that solve best a given task in a given environment [Nolfi *et al.* 1994]. Therefore, by modifying the fitness function,

the task or the environment, the researcher can strongly affect the evolutionary process. Ideally, a fitness function in ER should reveal how well the behavior of the controller solves the given task. In practice, many different fitness function types are used in ER experiments, which can be categorized by the quantity of a priori information on the controller that the designer integrates to the evolutionary process [Nolfi & Floreano 2001, Nelson *et al.* 2009, Montanier 2013].

2.4.1 The fitness function

In ER, a fitness function is a particular type of objective function that is responsible for determining which solutions within a population are better at solving the particular problem at hand. In other words, some performance indicators must be computationally defined and aggregated to determine whether a solution will survive and reproduce at a given stage of the evolutionary process. This aggregated function is referred to as fitness function.

In particular, each design solution is referred to as *controller* in the field of evolutionary robotics. The term *controller* is used to describe the computational portion of an autonomous mobile robot system (either real or simulated) that receives information from the robot's sensors, processes this information, and produces actuator or motor commands that cause the robot to move or interact with its environment. The controller in this sense might be thought of as the brain of the robot, and some ER researchers use this terminology [Nelson *et al.* 2009].

In ER, the fitness function plays a very important role in guiding the EC methods to obtain the best individual controllers with a large population of controllers. The aim of EC methods is to optimize this function. Thus individual controllers in a large population of controllers are selected or replaced based on this measure. Fig 2.2 (proposed in [Hartland 2009]) illustrates a standard evolutionary robotic process, i.e. an optimization fitness function process. A population of random controllers is created in the first generation. Each controller's performance is evaluated based on a fitness function. The best controllers are selected and undergo variation operators to generate offspring. Along the course of evolution, the controllers improve in order to maximize the fitness function.

2.4.2 Classification of fitness function

Previous work on fitness functions for evolutionary robotics focused on the amount of prior knowledge included in the fitness function [Nolfi & Floreano 2001, Nelson *et al.* 2009, Montanier 2013]. First, [Nolfi & Floreano 2001] proposed a classification of fitness functions with respect to three dimensions: explicit/implicit (measuring

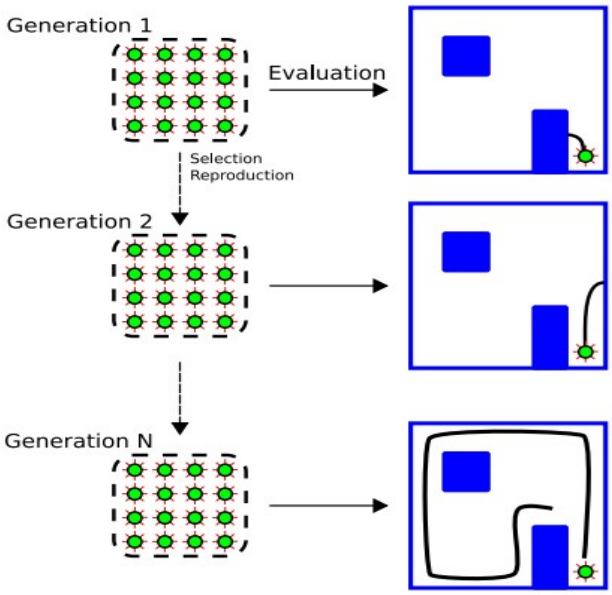


Figure 2.2: *Evolutionary robotics standard process: The first generation involves randomly generated robot controllers; each controller is evaluated according to the fitness function. The best individuals tends to be selected in order to produce new individuals. The new individuals undergo variations. The new individuals replace the old ones, leading to the next generation. Thanks to the fitness function, the adequate control characteristics emerge within the individuals, increasing the performance from time to time, until the algorithm reaches some termination criterion.* [Hartland 2009]

the way the goal is achieved versus measuring the level of attainment of the goal), external/internal (measuring fitness through an external observer versus measuring it internally with the robot), and functional/behavioral (rewarding a particular working modality versus the quality of the behavior). Second, [Nelson *et al.* 2009] focused on a single axis that represents the amount of a priori knowledge incorporated in the fitness function, and each class is listed in Table 2.1. Third, taking inspiration from [Nelson *et al.* 2009] to build a classification, [Montanier 2013] defined four main types of fitness function seen in the literature, and the characteristics of each class are discussed below.

All classifications mentioned above rely on the same claim: exploiting prior knowledge helps ER to find solutions quickly, but it prevents from discovering original solutions. To make fair comparisons between approaches, therefore, both the performance and the level of autonomy of the evolutionary process must always be taken into account. Furthermore, experiments with the novelty search show that prior knowledge can be misleading [Lehman & Stanley 2008, Lehman & Stan-

ley 2011].

Fitness function class	A Priori knowledge incorporated
Training data fitness functions (for use with training data sets)	Very high
Behavioral fitness functions	High
Functional incremental fitness functions	Moderate-high
Tailored fitness functions	Moderate
Environmental incremental fitness functions	Moderate
Competitive and co-competitive selection	Very low-moderate
Aggregate fitness functions	Very low

Table 2.1: Fitness function classes according to [Nelson *et al.* 2009]. See text for discussion.

Let us discuss the different types of fitness functions, following [Montanier 2013] :

Behavioral fitness function: Behavioral fitness functions are task-specific hand-formulated functions that measure various aspects of what a robot is doing and how it is doing it [Nelson *et al.* 2009]. For instance in [Jakobi 1998], the fitness function is considered a behavioral fitness function because it bases fitness on local motor behaviors and sensor responses and does not directly measure partial or overall completion. Another example is found in the locomotion of an octopod robots [Gomi & Ide 1998]. This approach lets few degrees of freedom to be optimized by the evolutionary process, which implies that the human engineer has a precise idea of how to perform the task. Hence, if this class of fitness function is employed, the human engineer should provide a large amount of knowledge on the problem to solve. These types of function generally include several sub-functions that are combined in a weighted sum, e.g. [Banzhaf *et al.* 1997] evolved 4 separated behaviors using embodied evolution and GP.

Tailored fitness functions: Tailored fitness functions are based on the measure of task completeness, but they may also contain behavioral terms as detailed in the previous paragraph. For example, in a photo-taxis task, a tailored fitness function might contain two parts: one is rewarding a robot that arrives at the light source; another one is maximized when the robot faces the sun. This type of fitness function is task-specific but tailored by the human engineer to accommodate the given problem. It is one of the most used class of fitness function in the ER field. Among the achievements made, one can count ball collection [Hoffmann

& Pfister 1996], coordinated movements [Matt Quinn *et al.* 2003], sequential tasks [Doncieux & Mouret 2010], and gait learning for a quadruped robot [Hornby *et al.* 2000a]. Within these approaches, the human engineer should know the elements necessary to the success of the task.

Aggregate fitness functions: Aggregate fitness functions reward the accomplishment of a given task or sub-task but use no information from the human engineer on how to do the task. This type of function aggregates all aspects of a robot's behavior in a single term. This is sometimes called *all-in-one*. For example, in a foraging scenario, a robot is located and gathers objects and then deposits them at a specific position (or a "nest"). The fitness of an evolving controller is computed based only on whether or not it completes the task. To be specific, an example for an aggregate fitness function for this task would be one that counts the number of objects at the nest after the end of a trial period. Aggregate fitness functions have been applied successfully in multiple cases such as gait evolution in legged robots [Zykov *et al.* 2004, Chernova & Veloso 2004], simpler locomotion [Zufferey *et al.* 2002, Di Paolo 2004] and object pushing [Hornby *et al.* 2000a] tasks.

Until recently, aggregate fitness selection have been dismissed by the ER community because of the so-called "*bootstrap problem*" [Kawai *et al.* 2001]. The bootstrap problem occurs when all individuals in the randomly initialized populations have same very low fitness, preventing evolution from getting it started and discovering promising regions. In order to overcome the bootstrap problem, some specific methods have been applied such as applying environmental incremental evolution in conjunction with aggregate selection [Nakamura *et al.* 2000] and using a preliminary bootstrap mode that gives way to aggregate selection later in evolution.

Implicit fitness functions: Implicit fitness functions operate at a more indirect, abstract level: reward is given for completing some task but the robot is free to achieve it in any possible way [Bird *et al.* 2008]. That is to say, when the task to perform is not known beforehand by the human engineer, and might change with time, the implicit fitness functions are considered by the human engineer. In this context the optimization process is based on the pressure to survive.

The maximization of the fitness function may cause the development of different strategies depending on the environment at hand, possibly involving other robots. Therefore, this type of fitness function can be applied to several scenarios without any modifications: For example, this approach is mainly used in the Embodied ER [Montanier 2013].We have also found one application investigating the notion of

creativity, where [Bird *et al.* 2008] use this approach to study the traces of robots as drawing resulting the pressure to harvest energy in order to survive.Also interestingly, this approach is used for genetic algorithm-based agent scheduling [Prashanth & Andresen 2001].

2.5 The Reality Gap Problem

The reality gap problem manifests itself as controllers evolved in simulation are underperforming when ported on real robots. Reality gap is the most critical issue with regard to practical applications. In theory, the reality gap would not exist if the optimization process could be achieved directly on the target robotics setup. In ER, however, solutions are commonly evolved in simulation for the purpose of speeding the search and experimental conveniency (Section 2.5.1). In practice, even if many works in ER are successful in building non-trivial and efficient controllers that correspond to original and complex behaviors [Prieto *et al.* 2010], these controllers are often locked in the simulated world because their results hardly translate from simulated to real world. This failure of evolved solutions to "cross the gap" from simulation to reality is termed the "reality gap" problem [Jakobi *et al.* 1995] (Section 2.5.2); as said, this is one of the major challenges facing the ER field. Quite a few solutions have been proposed to address this challenge and significant progress is being made (Section 2.6).

2.5.1 Evolution on simulated vs physical robots

As of now, several works have actually achieved evolution on physical robots, such as for evolving collision-free navigation on a Khepera [Floreano *et al.* 1994], optimizing the walking gait of an AIBO robot [Hornby *et al.* 2000b], of a pneumatic hexapod robot with complex dynamics [Lipson *et al.* 2006] or even a humanoid robot [Wolff *et al.* 2008]. While the optimization on the physical robot guarantees the relevance of the obtained result, robots have multiple limitations, listed below, which make them ill-adapted to run evolutionary algorithms on-board [Matarić & Cliff 1996, Koos *et al.* 2013]:

Time-consuming: In practice, performing evaluation on a physical robot can be very time-consuming. For instance in [Floreano *et al.* 1994], an avoidance control scheme is evolved on a real robot named Khepera. In this work, a generation takes approximately 39 minutes, making 65 hours for 100 generations to achieve the design of a wander behavior. It is extremely time consuming both from the robot and from the human supervisor viewpoint. The same experiments could be

performed in simulation in a matter of minutes.

Cost: Physical Robots are expensive devices. On one hand, as the behavior that corresponds to a given solution is not known before its evolution, there is the risk that harmful behaviors might be executed onto the robot. On the other hand, the physical hardware of a robotic system cannot survive the necessary continuous testing without constant maintenance and repairs. None of these problems exists in simulation.

Battery Lifetime: The unavoidable need to recharge robot batteries slows down further the experimental procedure. In most of the Khepera-based experiments described, the robot was tethered thus eliminating both the on-board power and the computation problem. However, tethering is not possible on all platforms and in all domains, nor does it scale up to multi-robot co-evolution experiments. Compared to a physical robot, a simulator does not require recharging.

For these reasons, simulation models are an appealing way to run evolutionary algorithms in a fully secure set-up, while significantly speeding up the optimization process [Harvey *et al.* 1992]. However, in reality, accurate simulators can be even slower than experiments, which leads to prohibitively long optimization processes. To obtain simulation models with lower computational costs, it is sometimes necessary to neglect some complex physical phenomena, which leads to simpler simulators, of course less accurate, but also faster [Koos *et al.* 2013].

2.5.2 How the Reality Gap Manifests itself

The difficulty of accurately simulating physical systems is well known in robotics [Brooks 1995]. Since it is impossible to simulate all details of a physical system, any abstraction made in a simulation may be exploited by the evolutionary computation method and may result in behavior that is ill-suited to reality.

For the sake of computational and experimental conveniency, many ER research works rely on simulators. The best controllers found in *in-silico* are then transferred onto the real robot. However, evolutionary algorithms often exploit simulation's discrepancies in an opportunistic manner to achieve high fitness values with unrealistic behaviors. If one transfers a controller designed in simulation that relies on badly modeled phenomena, the behavior observed in simulation does not match the one observed in reality, yielding the "*reality gap*" [Jakobi *et al.* 1995]. For instance in [Boeing *et al.* 2004], biped walking gaits are evolved *in-silico* but cannot run efficiently *in-situ*, i.e. in the real world. Many reality gap problems are also reported

in [Palmer *et al.* 2009], regarding a 12-DOF bipedal walking robot.

The reality gap problem remains a critical issue in ER as it often prevents the transfer of evolutionary results to real robots. More generally, it occurs whenever a controller is designed in simulation before application to a physical target robot. Therefore, crossing the reality gap in ER is of particular importance.

2.6 Gap Avoidance

The work in [Koos *et al.* 2013] distinguishes three main types of approaches of dealing with the reality gap problem:

2.6.1 Reality-based Optimization

In this type of approach, optimization takes place, fully or partly, on the real device.

As mentioned, one extreme approach to reduce the reality gap is to evolve controllers directly on the robots, as done in [Floreano & Mondada 1998], where an avoidance control is evolved on a Khepera mobile robot. In this work, the optimization required to achieve design of a desired behavior takes about 60 hours for 8000 evaluations. Other similar approaches have been implemented on real robots [Hemker *et al.* 2006, Zykov *et al.* 2004].

An alternative to these approaches is the use of both simulators and physical robots. For instance in [Pollack *et al.* 2000], the goal consists of co-evolving morphologies and controllers in the GOLEM project; the solutions were mostly evolved in simulation and only the last generations of the optimization process were conducted in reality. First, the robot morphology and its controller were co-evolved in a simulator, and then an embodied evolution took place on a population of physical robots having the best morphology for crossing the reality gap. A similar work is reported in [Nolfi *et al.* 1994] where a mobile Khepera robot addressed a navigation task with 30000 evaluations in simulation followed by 3000 evaluations on the physical robot.

2.6.2 Simulation-based Optimization

As said, simulation-based optimization approaches are used by some researchers because of the prohibitive computational cost of performing direct optimization in reality [Saunders *et al.* 2011]. A natural approach to dealing with the reality gap would be to consider more accurate simulation models. However, simulation models are often trade-offs between accuracy and computational cost. Accurate models can lead to very high computational costs, which also are hardly compatible with

optimization techniques. Besides, for some devices, such as bird-sized unmanned aerial vehicles that rely on little-known dynamics [de Margerie *et al.* 2007], perfect simulations are still out of reach.

Another approach to dealing with the reality gap consists of building a minimal simulation [Jakobi 1997] by only modeling meaningful parts of the target behavior. The unwanted phenomena are hidden in an envelope of noise or not modeled at all so that the evolved solutions cannot exploit them and have to be robust enough to achieve high fitness values. This approach has been successfully applied to designing walking gaits for an octopod robot [Jakobi 1998]. Moreover, the more realistic the amount of noise is, the better the transfer should be [Miglino *et al.* 1995]. The robustness of the behaviors can also be obtained by evaluating the solutions in different simulated environments and initial conditions as in [Thompson *et al.* 1999].

Some other works deal with the reality gap as an environment variation to be overcome online. In [Floreano & Urzelai 2001], the synaptic plasticity of neural network controllers is used to learn several sub-behaviors and also to overcome the gap when a solution is transferred onto the real device, by adapting online to the "new" environment. The robot can also explicitly build an approximate model of its environment, in order to use it as a reference and then adapt to the environment variation. For instance in [Hartland & Bredeche 2006], an anticipation module allows to build a model of the motor consequences in the simulated environment. Then, once in reality, some differences are encountered between this model and the current environment, a correction module performs an online adaptation to improve the behavior and to overcome the gap.

2.6.3 Robot-in-the-loop Simulation-based Optimization

These approaches rely mostly on simulators but also allow a few transfer experiments during the optimization. One way is to resort to co-evolution between simulators and controllers; the other way relies on a so-called surrogate model.

A first approach to dealing with the reality gap consists of resorting to co-evolution to improve both controllers and simulators at the same time. However, such co-evolutionary methods rely on the assumption that the simulation model can become accurate enough allow perfect transfer with only few experiments. In [Bongard & Lipson 2004], the exploration-estimation algorithm (EEA) evolves two populations: simulators and controllers. The simulators have to model the previously observed real data, and the controller that best discriminates between these simulators is transferred onto the real device to generate new meaningful learning data for the modeling part. This process is iterated until a good simulator is found and thereafter relevant controllers for a given task are built using it. This approach

has been successfully implemented with a four-legged robot [Bongard *et al.* 2006]. A similar method based on multi-objective evaluation of the solutions has been applied to a stabilization task with a simulated quadrotor helicopter [Koos *et al.* 2009]. Another similar EEA algorithm is the back-to-reality algorithm [Zagal & Ruiz-Del-Solar 2007], which does not resort to a disagreement measure, but tries to reduce the fitness variation observed between simulation and reality. As for EEA, it resorts to an update heuristic based on a disagreement measure that allows to reduce the number of experiments required to explore efficiently the solution space. The approach is applied to a ball-kicking task with a Sony AIBO robot.

The optimization process can itself directly rely on a so-called surrogate model by evaluating the individuals with a simple model of the fitness function instead of building an entire simulation model. The surrogate model has to be upgraded during the optimization process by conducting some test experiments depending on a given update heuristic; for instance, such an approach has successfully been applied to fast humanoid locomotion [Hemker *et al.* 2006]. Outside of ER, similar approaches have been applied to reality gap problems in the field of reinforcement learning. Abbeel et al. notably applied such techniques to aerobatic helicopter flight [Abbeel *et al.* 2007].

2.7 Beyond classic ER

2.7.1 On-line On-board Evolution

A categorization of evolutionary robotics algorithms has been proposed by [Eiben *et al.* 2010a], depending on when, where and how evolution takes place:

1. off-line or design time vs. on-line or run time (when)
2. on-board or intrinsic vs. off-board or extrinsic (where)
3. in an encapsulated or centralised vs. distributed manner (how)

While mostly off-line and extrinsic fitnesses are considered in evolutionary robotics, new issues must be considered to achieve on-line, on-board robotic evolution [Karafotias *et al.* 2011]:

- On-board evolution implies (possibly very) limited processing power and memory, thus the evolutionary algorithm must deal with limited computational and memory resources, limiting population size and number of evaluations;
- On-line evolution requires that the robots autonomously load and evaluate controllers without human intervention or any other preparation: the evaluation of a controller simply picks up the task where the previous evaluation left

off. This introduces significant noise in fitness evaluations because the starting conditions of an evaluation obviously can have great impact on a controller performance;

- Because the evolutionary algorithm has to be able to contend with unforeseen circumstances, it must either be able to (self-) adapt its parameter values as it operates or its parameters must be set to robust values that produce good performance under various conditions.

- The fitness function must be defined such that it can be computed on-board, without ground truth available; such fitness functions are referred to as intrinsic[1]. Such intrinsic fitness functions must not require extensive, "absolute", prior knowledge; this contrasts for instance with Novelty Search [Lehman & Stanley 2008], which requires the robot to know its position, and where all other robots ended up their trajectories.

In summary, the on-line on-board approach is that robot controllers are evolving during (and not before) their operational period and the computational processes behind evolution all take place inside (and not outside) the robots [Eiben *et al.* 2010b].

The on-line on-board approach has been successfully applied to obstacle avoidance and object attraction [Nordin & Banzhaf 1997], obstacle avoidance based on vision [Marocco & Floreano 2002], and gait learning in a quadruped robot [Hornby *et al.* 2000a]. However, these contributions have dealt with the above issues through tailoring evolutionary algorithms to the task at hand. Because of the lack of general mechanism to deal with all issues of on-line on-board algorithms, these contributions hardly extend to general ER in reality. This approach has been considered for problems involving multiple robots [Watson *et al.* 1999].

2.8 Intrinsic Fitness Functions

Two approaches have been designed to support Evolutionary Robotics in the context of online-onboard evolution, the *intrinsic motivation*, pioneered by [Oudeyer *et al.* 2007, Baranès & Oudeyer 2009, Oudeyer *et al.* 2012] and the curiosity- and discovery-driven controller optimization [Delarboulas *et al.* 2010]. As these approaches are rooted on Machine Learning and Information theory concepts, on the one hand, and they are the main inspirations behind the proposed contributions of this manuscript, they will be described in detail in Chapter 4.

[1] In some cases, a built-in fitness can be used to measure the robot reaction w.r.t. manually defined experimental conditions, e.g. the robot fitness is measured from the amount of light it perceives and the experimenter moves the light.

2.9 Discussion

In summary, the state of the art in Evolutionary Robotics presents a number of achievements, which address in different ways two interdependent issues:

The first one is to encode the application objective into an optimization criterion; this encoding represents a transfer of information from the human designer to the problem solving environment. It is common in the ER framework [Floreano & Mondada 1998] that fitness design proceeds by trials and errors: controllers optimizing a given fitness function show the inadequacies of this fitness function, that is, how far are the optima of the fitness function to address the designer goal. Accordingly, the fitness is manually refined to forbid the discovery of inappropriate solutions. This process, which might involve a few iteration steps, is referred to as *fitness shaping* process.

The second issue regards the actual computation of the optimization criterion. This step requires that either the ground truth involved in the fitness be available (as in simulation-based approaches) or that the fitness only requires information that is available "for free" to the robot.

Our approach will essentially aims at addressing both issues in an integrated way. Before presenting it, let us likewise describe the Machine Learning-based approaches to Robotics.

Chapter 3

Machine Learning

This chapter introduces some machine learning (ML) approaches aimed at autonomous robotics and discusses their strengths and weaknesses.

After a brief overview of ML, focusing more specifically on reinforcement learning (RL) and policy learning (section 3.1), their limitations with respect to exploratory robotics are discussed in section 3.2. The definition of an appropriate reward function in particular raises critical issues when considering *in situ* robotics (as opposed to, simulation-based robotic control). Section 3.3 therefore presents the state of the art related to RL and policy learning with implicit or unknown rewards.

Section 3.4 introduces the notion of *intrinsic motivation*, originated from cognitive science. Its algorithmic formalization, pioneered by [Schmidhuber 1991] and investigated thoroughly by [Oudeyer *et al.* 2007, Baranès & Oudeyer 2009, Oudeyer *et al.* 2012] is thereafter detailed. The key issue is to extract rewards from autonomous exploration, in a way compatible with the robot bounded computational and memory resources, and with no access to ground truth. Another approach, rooted in the multi-armed bandit framework, is presented with the goal of efficiently discovering all states within a limited distance from a starting state (section 3.5).

The chapter concludes with a discussion on the respective strengths and limitations of Evolutionary Robotics and Machine Learning-based Robotics regarding the design of exploratory robotic controllers. This discussion will inspire the approach investigated in our work, described in chapter 4, at the crossroad of ER and ML-based Robotics.

3.1 An overview of machine learning

Aimed at building intelligent agents, the field of machine learning inherits its goals and methodologies from both fields of *artificial intelligence* (AI) [Pfeifer & Gomez 2005] on the one hand, and statistics and data analysis [Davis & Sampson 1986, Silverman 1986, Dunlop & Tamhane 2000, Rice 2006] on the other hand. AI aims at building computational agents able to achieve reasoning and efficient decision making based on the available information and their knowledge (about the

world, the goals and the methods). Machine Learning aims at automatically acquiring such knowledge from the available data (e.g. sensor data, expert demonstrations, semantic Web) with some guidance of the human experts or teachers.

Machine Learning basically comes in two flavors. Statistical Machine Learning [Bishop 1995, Vapnik & Vapnik 1998, Vapnik 1999, Bolton & Hand 2002] heavily relies upon statistics and data analysis. Symbolic Machine Learning is more inspired from the so-called Good Old Fashion AI [Haugeland 1985], where the intelligent computational agent is provided with background knowledge and reasoning abilities. Modern ML tends to borrow all related fields (statistics, probability theory, data mining, pattern recognition, artificial intelligence, adaptive control, and theoretical computer science) their principles and algorithms to best exploit the available data and achieve the targeted goals.

Formally, ML goal is to build models, algorithms or strategies that automatically improve their performances through being provided with data or by experience [Mitchell & Michell 1997, Blum 2007], and adapt themselves to changes in the environment. Ideally, intelligent softwares should display the ML abilities along their life, achieving the so-called *lifelong learning* ability. Most machine learning algorithms have emerged during the last two decades; their maturity is witnessed as they achieve breakthrough performance in many application domains not amenable to standard, specification-based software engineering.

Besides its numerous applications in robotics (e.g. [Kober & Peters 2012, Pilarski *et al.* 2012, Modayil *et al.* 2014]), machine learning has been applied successfully to natural language processing [Manning & Schütze 1999], computer vision [Saxena *et al.* 2009], speech and handwriting recognition [LeCun *et al.* 2004], network security [Laskov *et al.* 2004], monitoring of electric appliances [Murata & Onoda 2002], drug discovery [Warmuth *et al.* 2003], neurosciences [Richiardi *et al.* 2013], and recommender systems to name a few.

3.1.1 Types of ML algorithms

As of now, machine learning algorithms can be classified into three categories depending on the input and expected output of the algorithms:

- **Supervised learning**:

  In supervised machine learning, an application domain is represented using descriptive features; a particular feature, called class or label, is to be explained or predicted from the other features. Supervised ML starts with a set of examples, where each example is made of a description referred to as instance, and the associated label value, that is, the value of the instance. In

propositional logic, which will be the only representation considered in the following, a training dataset $\mathcal{E} = \{(x_i, y_i), x_i \in X, y_i \in Y, i = 1 \dots n\}$ involves pairs (x_i, y_i), where instance x_i is represented as a vector of attribute values in the instance space X and y_i is associated class, element of the label space Y.

A supervised learning algorithm learns a function f, $f : X \longrightarrow Y$, such that $f(x) = y$ approximates the (unknown) label y associated with any further instance x in X. If space Y is a finite unordered set, the learning task is referred to as *classification*; if Y is the real-value space or a subset thereof, the learning task is referred to as *regression*.

State-of-art supervised learning algorithms include decision trees [Quinlan 1993], linear regression [Bishop 2006, Russell & Norvig 2010], artificial neural networks [LeCun *et al.* 1989, Paugam-Moisy *et al.* 2006, Krizhevsky *et al.* 2012], support vector machines [Boser *et al.* 1992] and kernel-based approaches [Schölkopf & Smola 2002].

- **Unsupervised learning**: Unsupervised learning considers a dataset similar to that of supervised learning, except for the labels, which are missing. $\mathcal{E} = \{x_i, x_i \in X, i = 1, ...N\}$. The purpose of unsupervised learning is to summarize the instances by grouping them in clusters, or by estimating the data distribution.

  Unsupervised learning algorithms include k-means clustering [Jain 2010, Celebi *et al.* 2013], ε- means algorithm [Duda *et al.* 2012], principal component analysis [Acharyya 2008], and Gaussian mixture models learned by expectation maximisation [Nodelman *et al.* 2012, Yildirim *et al.* 2014].

- **Reinforcement learning (RL)**: RL aims at sequential decision making, based on the exploration of the environment and of the agent action space [Sutton & Barto 1998]. Formally, the goal is to devise a policy that maximises the cumulative reward received during the agent lifetime (see below). RL has many applications in robotics, especially in mobile robot control [Kober *et al.* 2013, Kormushev *et al.* 2013].

The interested reader is referred to [Bishop 2006, Hastie *et al.* 2009, Duda *et al.* 2012, Michalski *et al.* 2013] for a comprehensive presentation of supervised and unsupervised learning[1]. The ML algorithms most relevant to our goal, reinforcement learning algorithms, are presented in next section.

[1] Two clustering algorithms, specifically k-means and ε-clustering algorithms [Duda *et al.* 2012] will be used and presented in Chapter 4.

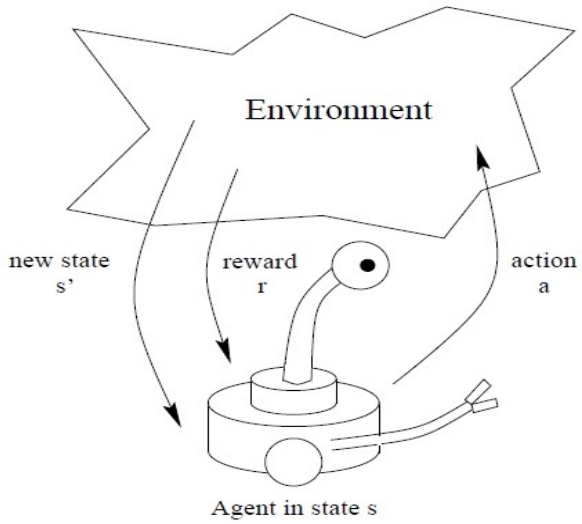


Figure 3.1: A reinforcement learning agent acting in an environment. Adapted from [Blynel 2000].

3.1.2 Reinforcement Learning

Reinforcement learning is a very active area of machine learning, receiving considerable attention from decision theory, operation research, and control engineering, where it has been called "heuristic dynamic programming" [Werbos 1987] and "neurodynamic programming" [Bertsekas & Tsitsiklis 1995]. In short RL is the problem faced by an agent or robot that must learn an appropriate behavior through trial-and-error interactions with a dynamic environment, as depicted in Fig 3.1. At each step t of interaction the agent perceives its current state s_t from the environment; the agent accordingly chooses some action a_t; upon this action, the agent arrives in a new state s_{t+1} and receives feedback about its action in the form of a reward signal r_t. Its goal is to maximize the total reward it receives over time.

Classical reinforcement learning approaches are based on the *Markov Decision Process* assumption (MDP; [Puterman 2009]), that is, the problem is formalized as a five-tuple: $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where:

- $\mathcal{S}$ is the *state space*. A state $s \in \mathcal{S}$ contains all relevant information about the current situation of the robot in the environment, required to select an action and to predict accordingly its future state. For example, in the navigation task a state can be described from the robot position and/or its sensor values. The state space can be discrete or continuous. In the navigation problem, $\mathcal{S}$ is a continuous space (e.g. the robot position is a real valued vector) or a discrete space (in a grid world).

- $\mathcal{A}$ is the *action space*. An action $a \in \mathcal{A}$ is used to control the state of the system like a motor instruction in the navigation task. The action space $\mathcal{A}$ include all possible decisions of the robot in any state (e.g. motor activation). Similar to the state space, the action space can be discrete or continuous.

- $T \in (\mathcal{P}_{\mathcal{S}})^{\mathcal{S}\times\mathcal{A}}$ is a *transition function*, with $\mathcal{P}_{\mathcal{S}}$ denoting the set of probability distributions over $\mathcal{S}$. The transition function T defines the conditional probability $T(s' \mid s, a)$ of arriving at next state s' by selecting action a in the state s.

- $R : S \times A \to \mathbb{R}$ is a *reward function* that defines the instant reward received by the robot through selecting action a in state s. In the navigation task, a reward function usually involves penalties (reflect the energy costs for taken actions) and bonuses (for reaching target positions).

- $\gamma \in [0, 1]$ is a *discount factor* indicating that the rewards should be gathered as early as possibly: the reward gathered at time step $t + 1$ worths less than the reward gathered at time step t, everything being equal.

The most common task in reinforcement learning is to discover an optimal policy π^* that maps the state to actions so as to maximize the expected return J, defined as the cumulative discounted reward gathered over time. Formally, for each policy π the policy return $J(\pi)$, the expected discounted reward collected by π over time, is defined as:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t)) | s_0 \sim \mathcal{D})\right] \tag{3.1}$$

where T is the time horizon (possibly infinite) and initial state s is drawn after the initial state distribution $\mathcal{D}$. RL aims at finding the optimal policy $\pi^* = argmax J(\pi)$. The main RL algorithms are based on learning the value function V_{π} [Sutton 1988], with

$$V_{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t)) | s_0 \sim \mathcal{D})\right] = R(s, \pi(s)) + \gamma \sum_{s}^{'} p(s, a, s') V_{\pi}(s')$$

and the optimal value function:

$$V^*(s) = max_{\pi} V_{\pi}(s)$$

While supervised learning can be applied to learn policies, too[2], the learning approach most relevant to robotic control is reinforcement learning, discussed below.

[2]For instance, assuming that $(s_0, a_0, \ldots s_T)$ records an expert demonstration, with actions a_t

3.2 Challenges in robotic RL

RL is known to be a hard problem, due to a mixture of fundamental, algorithmic and practical issues. Many of these issues are manifested in the robotics setting [Kober & Peters 2012, Kormushev *et al.* 2013].

3.2.1 Curse of dimensionality

As the state and action spaces of most robots are inherently high-dimensional, continuous[3], robotic systems often face the so-called "Curse of dimensionality" coined by Bellman [Bellman 1957]. One of the most common examples is humanoid robots, which involve high dimensional states and actions due to their many degrees of freedom. For instance in the ball-padding task shown in Figure 3.2, the robot state consists of its joint angles and velocities for each seven degrees of freedom as well as the Cartesian position and velocity of the ball, and the robot's actions are torques or accelerations. Then this robotic system has $2 \times (7 + 3) = 20$ state dimensions and 7-dimensional continuous actions [Kober & Peters 2012].

Such a high dimensionality sets a major challenge for the reinforcement learning discipline. In pure robotics, this challenge is handled by robotic engineers through a (manual) hierarchical task decomposition, that partially shifts complexity toward a sub-tasks, on a lower layer of complexity.

Classical reinforcement learning approaches often consider a grid-based representation with discrete states and actions, often referred to as a *grid-world*. In the ball-padding example, we may simplify the task by controlling the robot in racket space (which is lower-dimensional as the racket is orientation-invariant around the string's mounting point) with an operational space control law [Nakanishi *et al.* 2008]. RL researchers commonly use quite a few tools of computational abstractions to deal with high dimensionality, ranging from adaptive discretizations [Busoniu *et al.* 2010] and function approximation approaches [Sutton 1988] to macro-actions or options [Barto & Mahadevan 2003, Hart & Grupen 2011].

---

ranging in a finite action space, this trajectory can induce a classification problem, where state s_t being labelled as falling in class a_t [Lagoudakis & Parr 2003a, Lagoudakis & Parr 2003b]. As noted by many authors however [Chang *et al.* 2015], the resulting classifier suffers from the fact that expert demonstrations do not visit the bad state regions. For this reason, if ever the classifier makes a mistake and deviates from the good state region, it does not know how to recover. The policies learned by supervised learning thus suffer from a limited training coverage. See also section 3.3.1.

[3] Note that an action space $\mathcal{A} = \mathbb{R}^d$, with d a few dozens, which is common in robotics, is considered to be large in RL [Powell 2012].

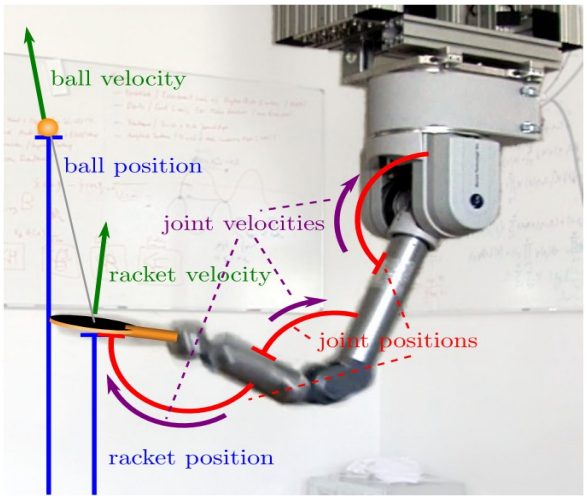


Figure 3.2: The sate space used in the modelling of a robot reinforcement learning task of paddling a ball. Adapted from [Kober & Peters 2012].

3.2.2 Curse of real-world samples

RL learns from data representing the agent trajectories. In robotics, the acquisition of such real-world samples is expensive in terms of time, labor and, potentially, financial cost. More specifically:

- Firstly, robotic hardware used to be expensive and require careful maintenance to face wear and tear. It is true to say that the entry ticket in the field is much decreased in the recent years, at least for what concerns companion robots such as Nao or study robots like iCub[4].
- Secondly, a significant expertise remains needed to set up experiments and acquire usable data. In particular, the experimenter must carefully design and supervise the experiments.
- Thirdly, the data acquisition process faces issues related to discretization of time and delays. As reinforcement learning algorithms are inherently implemented on a digital computer, the discretization of time is unavoidable although physical systems are inherently continuous time systems. In turn, time discretization of the actuation can generate undesirable artifacts (e.g., the

[4]In 2011, the Nao and iCub are equipped in the experiment, with the prices at $15,600 and about $3000,000 (est.) respectively [Felch & Granger 2011], and into 2015, their price drop to $7,990 and about $275,000 (the price information from web site:
1. http://robohub.org/nao-next-gen-now-available-for-the-consumer-market
2. http://www.icub.org/bazaar.php).

distortion of distance between states). Besides, all physical systems exhibit delays in sensing and actuation, for example, the state of the setup (represented by the filtered sensor signals) may frequently lag behind the real state due to processing and communication delays.

- Finally, the decision making process faces the classical constraints of dynamic systems: the movement cannot be paused and actions must be selected subject to time-budget constraints.

3.2.3 Curse of modelling issues

One way to offset the cost of real-world interaction is to use accurate models as simulators. It is often the case that a policy is trained in simulation and subsequently transferred to the real robot. Unfortunately, building a sufficiently accurate model of the robot and its environment is challenging and it often requires very many data samples. As said in Section 2.5, simulated behavior is often observed to deviate from the one observed in the real robot; this phenomenon is referred to as **reality gap** problem [Jakobi *et al.* 1995, Bongard & Lipson 2004, Lipson *et al.* 2006].

For tasks where there is no stability or safety issue (the robot does not require active control to remain in a safe state or return to it), the transfer onto the real robot of the policy learned in simulation often works well [Kober & Peters 2011]. Nevertheless, tasks can often be learned better in the real world than in simulation due to complex mechanical interactions (including contacts and friction) that have proven difficult (or too computationally expensive) to model accurately. Additionally, it is often the case that the learning algorithm can and does exploit the inaccuracies of the simulator.

In some settings referred to as unstable [Kober & Peters 2011], small variations have drastic consequences. For example, in a pole balancing task, the equilibrium of the upright pole is very brittle and constant control is required to stabilize the system. Policy transfer often performs poorly in this setting.

3.2.4 Curse of goal and reward specification

In robot RL, an often underestimated problem is the goal specification, which is achieved through designing a good reward function. As mentioned, the goal of RL algorithms is to maximize the cumulative long-term reward. In practice, designing a good reward function in robot reinforcement learning often is a daunting task. In many domains, providing rewards only upon task achievement, e.g., when a table tennis robot wins the game, will result in an apparently simple, binary reward specification. However, a robot may receive such a reward so rarely that it is unlikely

to ever succeed in the lifetime of a real-world system. Hence, instead of using only simple binary rewards, additional prior knowledge in form of additional rewards frequently needs to be provided, possibly iteratively, along some so-called *reward shaping* process. The reward shaping is a technique that provides localized feedback based on prior knowledge to guide the learning process [Ng *et al.* 1999, Brys *et al.* 2014, Kim *et al.* 2015]. The trade-off between different factors may also be essential as hitting a table tennis ball very hard may result in a high score but is likely to damage the robot. RL algorithms are also notorious for exploiting the reward function in unexpected ways, especially when the RL is done locally and not globally.

In some cases the domain can be most naturally represented using a high dimensional state and action space. However, this representation is hardly conducive to reinforcement learning due to both computational and statistical limitations. In such cases, a reward skillfully specified in terms of the features of a simpler, lower dimensional space in which the learning algorithm operates can prove remarkably effective. There is a trade-off between the complexity of the reward function and the complexity of the learning problem. For example, an outdoor robot named Crusher [Ratliff *et al.* 2007] reasons about the world on a long time horizon scale as if it was a very simple, holonomic robot operating on a fine grid of continuous costs. However, the actual problem consists of minimizing both the time to reach the goal and the risk of the robot behavior; these two objectives can hardly be modelled in such a simple state space.

Most generally the reward function in reinforcement learning plays the same role as the fitness function in evolutionary algorithms, and the problems encountered when designing a reward function are similar to those related with fitness design in evolutionary algorithms, explaining to some extent how difficult the design of a good reward function is (section 3.2.4).

For this reason, various algorithms have been developed to overcome the difficulty of reward design, specifically:

- Inverse Reinforcement Learning (IRL) [Ng *et al.* 2000] is presented in section 3.3.1, where the reward function is learned based on expert demonstrations;
- Preference-based Reinforcement Learning (PBRL) [Wirth & Fürnkranz 2013c, Akrour 2014] is presented in section 3.3.2, where the fitness function is learned based on the expert feedback;
- *Intrinsic motivation* [Baranès & Oudeyer 2009, Oudeyer *et al.* 2012], can be viewed as a particular type of RL architecture [Sutton & Barto 1998], where rewards are not designed by the human expert or engineer but built-in and

autonomously measured by the agent itself, akin a computational "instinct" (section 3.4).

3.3 Policy Learning with no Explicit Reward

As said, the reward function satisfies the MDP assumption in the standard RL MDP setting. In quite a few contexts, the reward function does not comply with the MDP setting, or is not naturally present in the environment. Let us examine two concrete such cases. One case is when the reward is defined from the demonstrations of an expert (section 3.3.1). Another case is where the expert neither defines an appropriate reward function nor demonstrates a quasi-optimal policy; instead, the expert only provides feedback as to whether the current policy improves on the previous ones (section 3.3.2).

3.3.1 Inverse reinforcement learning

Similar to standard RL, IRL assumes that the agent is acting in a Markov Decision Process framework, except for the fact that the reward function of the MDP is not known to the agent. This can also be written as a MDP without a reward specified, denoted by MDP\R. IRL is a paradigm for learning a reward function from the demonstrations of an expert [Ng *et al.* 2000, Zhifei & Joo 2012, Muelling *et al.* 2014]. Formally, let $(s_0, a_0 s_1, \dots s_T)$ denote an expert trajectory. Assuming that the expert's behavior is optimal (according to his $-$ hidden $-$ reward function), the idea is to learn a reward function r such that the policy associating action a_t to state s_t is optimal in terms of cumulative discounted reward with respect to reward function r. Once r is learned, then standard RL can be applied, with the benefit that this reward function makes it possible to extend and/or adapt the expert trajectory, typically when the robot leaves the regions visited by the expert trajectories.

This general IRL algorithm is displayed in Algorithm 2 [Muelling *et al.* 2014]. Generally, most IRL approaches rely on a given model of the environment or assume that it can be accurately learned from the demonstrations. A set of expert features is first defined (e.g., for a car driving task, the informed features include the speed of the car, the number of pedestrians the car is bumping into, whether the car is leaving the road); the sought reward function R is defined as a weighted linear combination of these m features f_i (with positive weights):

$$R(s,a) = \sum_{i=1}^{m} w_i f_i(s,a) = \mathbf{w}^T \mathbf{f}(s,a) \tag{3.2}$$

where $\mathbf{w} \in \mathbb{R}^{+,m}$ and $\mathbf{f}(s,a) \in \mathbb{R}^m$. For a given trajectory $\tau = s_0 a_0, \ldots, s_T$, the feature counts are given by $f_i^\tau = \sum_{t=1}^{H} \gamma^t f_i(s_t, a_t)$. For each feature and each considered policy, one considers the expected feature count associated to the policy, that is, the cumulative discounted value of this feature along the policy trajectory, in expectation ($\mathbb{E}_\pi[f]$).

Considering an initial weight vector $\mathbf{w}$ and the associated policy, IRL iteratively updates the weight vector, by considering that the optimal objective value is the one reached by the expert trajectory itself (line 6). The stopping criterion is the convergence of the weight vector.

Algorithm 2 General IRL Algorithm

1: **Input:** $D = \{\tau\}_{p=1}^{P}$ expert demonstrations
2: **Initialize:** reward feature weights $\mathbf{w}^0, j = 1$
expert feature counts $\mathbb{E}_{\pi_0}[f] = \frac{1}{P} \sum_{\tau \in D} \mathbf{f}^\tau$
3: **repeat**
4: **Optimize** π_j based on $\mathbf{w}^{j-1}$
5: **Estimate f**
6: **Update** $\mathbf{w}^j$ such that $(\mathbf{w}^j)^T \mathbb{E}_{\pi_j}[f] < \mathbf{w}^j \mathbb{E}_{\pi_0}[f]$
7: $j \leftarrow j + 1$
8: **until** $\|\mathbf{w}^j - \mathbf{w}^{j-1}\|_2 < \varepsilon$

The problem of IRL is, by definition, ill-posed [Ng *et al.* 2000] since different rewards can produce the same behavior [Ng *et al.* 1999]; accordingly, a demonstration cannot lead to define a single reward signal, neither to discriminate among an infinite set of reward functions. This indeterminacy is addressed [Syed *et al.* 2008, Van der Spek 2014] by requiring the features weights w_i to be positive with $\sum_i w_i^* = 1$ [Syed *et al.* 2008]. Each basis reward function f_i has a corresponding basis value function $V^i(\pi)$, with $V^i(\pi^E)$ the basis value function associated with the expert demonstration. Since by linearity

$$V(\pi) = \sum_i w_i^* V^i(\pi)$$

it therefore follows that the difference between $V(\pi)$ and $V(\pi^E)$ is upper bounded by $K \max_i V^i(\pi) - V^i(\pi^E)$, with K the number of basis functions. The goal then becomes to find a policy π^A solution of the following min max problem:

$$\pi^* = \arg \min_\pi \max_i |V^i(\pi) - V^i(\pi^E)| \tag{3.3}$$

yielding weights w^*.

There are many other ways to resolve the indeterminacy or to perform IRL. For example, [Ratliff *et al.* 2006] suggested a maximum margin planning approach.

[Ziebart *et al.* 2008] suggested an algorithm where the principle of maximum entropy was exploited. Other techniques are using a Bayesian nonparametric mixture model [Michini & How 2012] or score-based classification [Geist *et al.* 2013]. A recent review of IRL algorithms can be found in [Zhifei & Joo 2012].

3.3.2 Preference-based Reinforcement Learning

Preference-based Reinforcement learning (PBRL) is a novel research direction combining RL and preference learning [Fürnkranz & Hüllermeier 2010, Akrour *et al.* 2011a]. Compared with the conventional RL, it does not assume the availability of a reward signal, but only requires preference judgments about policies, trajectories, states or actions [Wirth & Fürnkranz 2013a]. There are two main approaches to representing preference, namely in terms of *utility functions* evaluating individual alternatives or *preference relations* comparing pairs of competing alternatives [Fürnkranz *et al.* 2012].

In PBPI (Preference-based Policy Iteration) [Fürnkranz *et al.* 2012] and APIALP (A Policy Iteration Algorithm for Learning from Preference Feedback) approach [Wirth & Fürnkranz 2013b], the principle is to compare actions a and a' in a given state s, given a policy π used ever after (roll-out policy). The user thus emits a preference among the two trajectories, which translates into a preference among actions in a given state.

In [Akrour *et al.* 2011a, Akrour *et al.* 2011b, Wilson *et al.* 2012, Busa-Fekete *et al.* 2013], the user is asked his preferences among (fragments of) trajectories. The preference judgment is used to learn a trajectory ranker, which can be used for creating an improved policy by utilizing evolutionary strategies [Busa-Fekete *et al.* 2013] or Bayesian optimization [Akrour *et al.* 2011a,Akrour *et al.* 2011b,Wilson *et al.* 2012].

For instance in Preference-based Policy Learning (PPL) [Akrour *et al.* 2011a], the agent demonstrates a few policies, receives the expert's preferences about the demonstrated policies, constructs a utility function on the trajectory space compatible with all expert preferences, uses it in a self-training phase, and demonstrates in the next iteration the policy maximizing the current utility function. In particular, in the iteration process, it is assumed that there exists a utility function U that is linear in terms of features: $U(s,a) = \langle \mathbf{w}\phi(s,a) \rangle$ [Akrour *et al.* 2011a, Akrour *et al.* 2011b]. The utility function of a policy can then be written as:

$$U(\pi) = \langle \mathbf{w}\phi^{\tau}(\pi) \rangle$$

where the weight vector $\mathbf{w}$ is determined by standard preference learning and ϕ^{τ} denote the discounted expectation of the features in ϕ as in the IRL setting. This

can be achieved by solving Eq. 3.4 under the preference constraints, where $T_i \succ T_j$ stands for trajectory T_i is preferred over trajectory T_j, and $C_i = \phi^\tau(T_i)$, $C_j = \phi^\tau(T_j)$ stand for the discounted features count of the state representation, as in IRL.

$$\begin{aligned} &\text{Minimize} \frac{1}{2} \parallel \mathbf{w} \parallel^2 + c \sum_{i,j, C_i \succ C_j \in \zeta} \xi_{i,j} \\ &\text{s.t.} \langle \mathbf{w}, C_i \rangle - \langle \mathbf{w}, C_j \rangle \geq 1 - \xi_{i,j} \quad \text{and} \quad \xi_{i,j} \geq 0 \quad \text{for all} \quad T_i \succ T_j \end{aligned} \tag{3.4}$$

The utility function defined by the weight vector $\mathbf{w}$ associates a value to a policy, enabling to define an order on the policy space Π. The PPL process thus alternates between learning a utility function from the preference constraints, finding a policy maximizing this utility function (with an exploration term), displaying a demonstration based on this policy and receiving the user's preference judgment about whether the new demonstration improve upon the previous best one. The maximization part is achieved using an evolutionary strategy, more specially the $(1 + \lambda) - ES$ algorithm by [Auger 2005] in [Akrour *et al.* 2011a], or Bayesian optimization in [Akrour *et al.* 2014].

In [Busa-Fekete *et al.* 2013], stochastic optimization (CMA-ES [Hansen & Ostermeier 1996, Hansen & Ostermeier 2001]) is used for optimizing the parameters of a parametric policy, and it is performed directly in a policy space. Each candidate policy π of the current iteration is used to sample a limited amount of trajectories. The pairwise preference relation is now used to estimate how often $T_i \succ T_j$. Using a racing algorithm witch utilizes Hoeffeding bounds enables the determination of a ranking for the policies based on the fraction of dominating trajectories [Heidrich-Meisner & Igel 2009]. This ranking is then used within the CMA-ES framework to create new policies.

The main merit of the Preference-based Reinforcement Learning is that it relaxes the expertise requirement: it does not require an expert to design the reward function, nor to demonstrate an optimal policy, nor even, to know how to solve the task [Akrour *et al.* 2014]. The only assumption done is that the teacher can compare two demonstrations and assess which one is more conducive to achieve the goal.

Preference-based Reinforcement Learning is driven by the human being preferences. The next section will examine other approaches, where the learning agent is internally driven and referred to as autotelic [Csikszentmihalyi & Csikzentmihaly 1991, Csikszentmihalyi 2000].

3.4 Intrinsic Motivation

This section presents an overview of the intrinsic motivation system. Generally, intrinsic motivation is a mechanism that guides curiosity-driven exploration, that

was initially studied in psychology and is now also being approached in neuroscience (section 3.4.1). Intrinsically motivated exploration, inspired from these approaches, has been devised (section 3.4.2). In section 3.4.3 we present three computational models of intrinsic motivation. Such a computational model applied in robotics, referred to as Intelligent Adaptive Curiosity, will be reviewed in detail in section 3.4.4. Finally, in section 3.4.5, hybrid approaches based on intrinsic motivation will be reviewed, overcoming some limitations of intrinsically motivated exploration methods.

3.4.1 Definitions

The notion of intrinsic motivation appears with some different though related content, in psychology, neuroscience and robotics.

Psychology

The concept of intrinsic motivation has been introduced in the 1950s in animal psychology [Harlow 1950] and has been further elaborated in human psychology [Deci & Ryan 1985]. Intrinsic motivation was identified in animals and humans as the set of processes which push organisms to spontaneously explore their environment even when their basic needs such as food or water are satisfied. More generally, in psychology, an activity is characterized as intrinsically motivated when there is no apparent reward except the activity itself [Ryan & Deci 2000]. Following this idea, most children playful or explorative activities can be characterized as being intrinsically motivated. Also, much adult behaviour seem to belong to this category: free problem- solving (solving puzzles, crosswords), creative activities (painting, singing, writing during leisure time), gardening, hiking, etc [Kaplan & Oudeyer 2007]. Quite a few theories of intrinsic motivation have been elaborated to understand which features of given activities could make them intrinsically motivating or "interesting" for a particular person at a particular moment of time. In this context, "interestingness" was proposed to be understood as related to concepts such as novelty [Hull 1943, Montgomery 1954], reduction of cognitive dissonances [Festinger 1957, Kagan 1972], optimal incongruity [Berlyne 1960], effectance and personal causation [De Charms 1968, White 1959], or optimal challenge [Csikszentmihalyi 1997].

Neuroscience

Independently, some neuroscientific studies suggest that the neuromodulator dopamine has long been associated with reward learning and rewarded behav-

ior [Schultz 1998, Di Chiara 1999]. Recent studies have focused on the idea that dopamine not only plays a critical role in the extrinsic motivational control of behaviors aimed at harvesting explicit rewards, but also in the processing of types of intrinsic motivation associated with novelty and exploration [Dayan & Balleine 2002, Kakade & Dayan 2002], such as the memorization of novel information [Lisman & Grace 2005] and the learning of novel actions [Redgrave & Gurney 2006]. A key issue is whether dopamine neurons report a " prediction error" or a " reward prediction error" [Horvitz 2000]. After [Panksepp 1998], there is ample evidence to suggest the existence of a SEEKING system responsible for exploratory behaviours: *This harmoniously operating neuroemotional system drives and energizes many mental complexities that humans experience as persistent feelings of interest, curiosity, sensation seeking and, in the presence of a sufficiently complex cortex, the search for higher meaning* [Panksepp 1998] p.145. This suggests that intrinsic motivation systems could be present in the brain in some form or another and that signals reporting prediction error could play a critical role in this context.

Robotics

Following the pioneering work of Schmidhuber [Schmidhuber 1991], the concept of intrinsic motivation has been used in machine learning and developmental robotics [Oudeyer *et al.* 2007, Baranès & Oudeyer 2009, Oudeyer *et al.* 2013, Schlesinger 2013] as a means for developing artificial systems that can autonomously learn several different skills. *The idea is that intelligent machines and robots could autonomously acquire skills and knowledge under the guidance of intrinsic motivations, and later exploit such knowledge and skills so to accomplish the tasks that are useful for the user in a more efficient and faster way than if they would have to acquire them from scratch. This possibility would clearly enhance the utility of intelligent artificial systems* [Baldassarre & Mirolli 2013]. A key idea of such approaches to intrinsic motivation is that learning progress in sensorimotor activities can generate intrinsic rewards in and for itself, and drive such spontaneous exploration [Gottlieb *et al.* 2013]. Learning progress refers to the infant's improvement of his predictions or control over activity they practice, which can also be described as reduction of uncertainty [Friston *et al.* 2012].

In short, intrinsic motivation refers to a mechanism pushing individuals to select and engage in activities for their own sake because they are inherently interesting or enjoyable. Extrinsic motivation is contrasted with intrinsic motivation after [Ryan & Deci 2000]: *Extrinsic motivation is a construct that pertains whenever an activity is*

done in order to attain some separable outcome. Extrinsic motivation thus contrasts with intrinsic motivation, which refers to doing an activity simply for the enjoyment of the activity itself, rather than its instrumental value. Accordingly and as proposed in [Oudeyer & Kaplan 2007], a central feature that differentiates intrinsic and extrinsic motivation is the instrumentalization of the activity[5].

3.4.2 Intrinsically motivated exploration

Robots are expected to deal with a wide variety of tasks like manipulating objects or interacting with humans in a changing environment. In such an open world setting, not all relevant information is known at design time. The challenge is to enable the robot to interact with its physical and social environment, to learn cumulatively novel skills that were not initially programmed, in a way that is analogous to human development, that is, without requiring the robot to be reprogrammed by the designer. In such contexts, reinforcement learning or evolutionary robotics approaches seem to be limited as they proceed by associating a specific reward or fitness function to each task to learn.

In order to allow robots to learn more autonomously a wider diversity of tasks, a few researchers have started to address the problem of designing intrinsic motivation systems to drive active learning, inspired by research in developmental psychology and neuroscience [Oudeyer & Kaplan 2008, Baranes & Oudeyer 2013, Moulin-Frier *et al.* 2013]. The idea is that a robot controlled by such systems would be able to autonomously explore its environment not to fulfil predefined tasks but driven by some form of intrinsic motivation that pushes it to search for situations where learning happens efficiently. Technically, such control systems can be viewed as particular types of reinforcement learning architectures [Sutton & Barto 1998], where rewards are not provided externally by the experimenter but self-generated by the agent itself. The term "intrinsically motivated reinforcement learning" has been used in this context [Barto *et al.* 2004].

3.4.3 Computational models of intrinsic motivations

Computational architectures based on intrinsic motivation have been developed since the 1990s, and can be categorised based on the measures that are used by the learning agent to evaluate the intrinsic interestingness of an activity or a situation. Three broad types of measures of interestingness have been proposed to implement

[5] Let us examplify the activity instrumentalization as follows. Assuming that a person works for money, then her work is not done for its own sake but for the separate outcome of getting money; the person is extrinsically motivated. Assuming on the contrary that the person works for the sake of her work meaning, then her behavior is intrinsically motivated.

intrinsic motivation [Oudeyer & Kaplan 2007, Oudeyer & Kaplan 2008, Baldassarre & Mirolli 2013].

3.4.3.1 Knowledge-based models

A first computational approach to intrinsic motivation is related to the difference between the outcome observed and the expectation of the robot. Most proposed models of intrinsic motivation are knowledge-based as they depend on the stimuli perceived by the learning agent (and on their relations with the agent expectations, including those related to the results of the agent actions) rather than on the agent skills. Within this approach, there exist two approaches depending on the way knowledge and expectations are represented: an information theoretic /distributional framework and a prediction framework.

Information theoretic and distributional models. This approach is based on distribution-based representations, where the agent estimates probabilities of observing certain events in particular contexts. More precisely, the agent internally builds and estimates a probability distribution of events across the whole space of possible events, e.g. depending on its actions. Finally, the quality of this distribution estimate is characterized with the concept of entropy.

- **Empowerment:** [Capdepuy *et al.* 2007] defined a measure for the maximum amount of information that an agent could send from its actuators to its sensors via the environment, called empowerment.

- **Information gain motivation:** In [Ryan & Deci 2000], intrinsic motivation is related to the natural human propensity to learn and assimilate. Assimilation is viewed as a type of compression, i.e., new inputs are embedded in old schemas [Bruner 1991, Schmidhuber 2010]. *In information theoretic terms, this notion of assimilation or of "pleasure of learning" can be modeled by the decrease of uncertainty in the knowledge that the robot has of the world after an event has happened* [Oudeyer & Kaplan 2008]. For instance, this information gain motivation has been used in [Roy & McCallum 2001].

- **Uncertainty motivation (UM):** The tendency to be intrinsically attracted by novelty has often been used as an example in the literature on intrinsic motivation. The motivation for introducing novelty is to avoid model habituation; typically human babies get bored by constant stimulation and are attracted to novel stimuli. A straightforward computational implementation is to associate with every observed event, a reward which is inversely proportional to

its probability of observation. This reward computation mechanism can then be integrated within a reinforcement learning architecture. The system compares the predicted next state to the actual next state, and if the prediction is incorrect, novelty is considered to be high. For example, UM-like mechanisms based modes have been implemented in [Huang & Weng 2002, Huang & Weng 2004].

Some other information theoretic and distributional models such as distributional surprise motivation and distributional familiarity motivation have also been used [Oudeyer & Kaplan 2007].

Predictive models. This approach is based on the use of predictors (e.g. neural network or support vector machines) that make direct predictions about future events, as knowledge and expectations are not always easily represented by probability distributions. In this kind of computational models of intrinsic motivation system, these predictors are typically used to predict some properties or sensorimotor states that will happen in the future given the current sensorimotor context and possibly the past sensorimotor context. The main point is that the ground truth, the event that actually happens is known at the next time step or after a short delay; therefore, the prediction can be compared to the ground truth and the difference thereof is used as a signal:

- **Predictive novelty motivation:** A simple novelty-based intrinsic motivation is to directly use the prediction error as reward, where interesting situations are those for which the prediction errors are highest, as in [Barto *et al.* 2004] for instance.

- **Intermediate level of novelty motivation:** Human beings seem attracted by situations which are neither completely uncertain not completely certain. [Hunt 1965] proposed the concept of optimal incongruity. He argued that interesting stimuli are those where there was a discrepancy between the perceived and standard levels of the stimuli. [Berlyne 1960] developed similar notions as he observed that the most rewarding situations were those with an intermediate level of novelty, between already familiar and completely new situations. One manner to model optimal incongruity is to use a threshold that defines this intermediate level of novelty, where interesting situations are related to both prediction error and the threshold.

- **Learning progress motivation (LPM):** Here, intrinsic motivation is modelled by rewarding the agent when predictions improve over time. Thus, the

agent is expected to maximize the decrease in its prediction error, i.e. effectively rewarding knowledge acquisition *per se*. This mechanism corresponds to the concept of epistemic curiosity proposed by Berlyne, which was defined as a "drive to know" that was aroused by conceptual puzzles and gaps in knowledge [Berlyne 1965]. A first computational formalization thereof was proposed in [Schmidhuber 1991], which described a model of curiosity that rewards agents when prediction errors decrease over time. Another analogous computational formalization was proposed in [Oudeyer & Kaplan 2007] under the name of Intelligent Adaptive Curiosity (IAC), together with a mechanism for automatically dividing the whole sensorimotor space into subregions within which to compute the learning progress and on which to focus learning.

Two other prediction-based intrinsic motivations are Predictive familiarity motivation and Predictive surprise motivation, which have also been introduced in [Oudeyer & Kaplan 2007, Baldassarre & Mirolli 2013].

3.4.3.2 Competence-based models

A second major computational approach to intrinsic motivation, referred to as competence-based models (CBIM), is based on measuring the agent competence for achieving self-determined tasks. In CBIM the agent is typically rewarded when its ability to accomplish a goal improves, independently from the origin of the goal [Chentanez *et al.* 2004, Schembri *et al.* 2007, Baranes & Oudeyer 2013, Santucci *et al.* 2013]. Importantly, competence is dependent on goals: some states, out of all possible states, are selected as desired states, and hence the agent works to achieve them. Indeed, CBIM is directly inspired from psychological theories of effectance [White 1959], causation [De Charms 1968], "Flow" [Csikszentmihalyi 1997] and competence and self-determination [Ryan & Deci 2000]. Basically, these approaches argue that what motivates people is the degree of control the can have on other people, external objects and themselves, or in other words, the amount of effective interaction. Besides, in an analogous manner, the concept of optimal challenge has been proposed, such as in the abovementioned theory of "Flow".

A more recent model of competence based intrinsic motivations is referred to as Self-Adaptive Goal Generation Robust-Intelligent Adaptive Curiosity (SAGG-RIAC) [Baranes & Oudeyer 2010], which considers as interesting the local improvement of its competence to reach high-level self-generated goals; [Baranes & Oudeyer 2013] goes further and develops goal-oriented exploration algorithms where the agent self-determines goals where to make progress.

3.4.3.3 Morphological models

A third approach is related to the structural relationship among multiple sensorimotor channels, and is based on comparison information characterising several pieces of stimuli perceived at the same time in several parts of the sensory input.

In the following, two examples of morphology-based intrinsic motivation proposed by [Oudeyer & Kaplan 2007] are described; they both rely on the formalization the sensorimotor flow experienced by a robot. Let $SM(t)$ denote the vector of all sensorimotor values at time t, with $\langle SM(T)\rangle_\tau$ the average of the sensorimotor vector over the last τ time steps. We will use the notation $SM(\rightarrow t)$ to denote a sequence of sensorimotor vectors up to time t, with $r(SM(\rightarrow t))$ the associated reward. Let ε denote a very small constant. Two typical examples of this type of intrinsic motivation mechanism are as follows:

1. **Stability motivation** aims at keeping the sensorimotor flow close from its average value.

$$r(SM(\rightarrow t)) \propto \frac{1}{\|SM(t) - \langle SM(T)\rangle_\tau\| + \varepsilon}$$

2. **Variance motivation** aims at a high variance of the sensorimotor vector.

$$r(SM(\rightarrow t)) \propto (\|SM(t) - \langle SM(T)\rangle_\tau\|)$$

The choice of the model depends on the context. For example, stability motivation can be used to decrease the inherent instability of perception and support a tracking behavior [Kaplan & Oudeyer 2003]. On the contrary, variance motivation could lead to explore unknown sensorimotor contingencies far from equilibrium. Another morphological model, based on information theory, has been studied in [Sporns & Lungarella 2006], investigating how various information theoretic cost functions to be optimised by a sensorimotor system led to self-organized coordinated behaviours.

3.4.4 Intelligent Adaptive Curiosity

Since the presented work is inspired by the models proposed and used in [Oudeyer *et al.* 2005, Oudeyer *et al.* 2007, Baranès & Oudeyer 2009, Lopes *et al.* 2012, Baranes & Oudeyer 2013], let us review these models in more detail. This section focuses in particular on Intelligent *Adaptive Curiosity* (IAC) [Oudeyer *et al.* 2007] and *Robust Intelligent Adaptive Curiosity* (R-IAC) [Baranès & Oudeyer 2009], using as intrinsic reward the learning progress motivation mentioned above. The architecture R-IAC is a refinement of the IAC architecture, as described in [Baranes & Oudeyer 2013]. The central contribution of both IAC and R-IAC systems lie in the way rewards

are defined and computed, i.e. through region-based hierarchical multiresolution evaluation of learning progress [Baranès & Oudeyer 2009]. Such an approach based on the optimization of learning progress (prediction progress) belongs to the family of knowledge-based intrinsic motivation systems mentioned above (section 3.4.3.1).

As shown in Fig 3.3, IAC and R-IAC rely on the same general architecture [Baranès & Oudeyer 2009], which will be technically detailed in the IAC context. A major component of IAC is dividing the sensory-motor space into independent regions handled by local learning experts. Another major contribution is a quantitative measure for learning progress of an exploring intelligent agent.

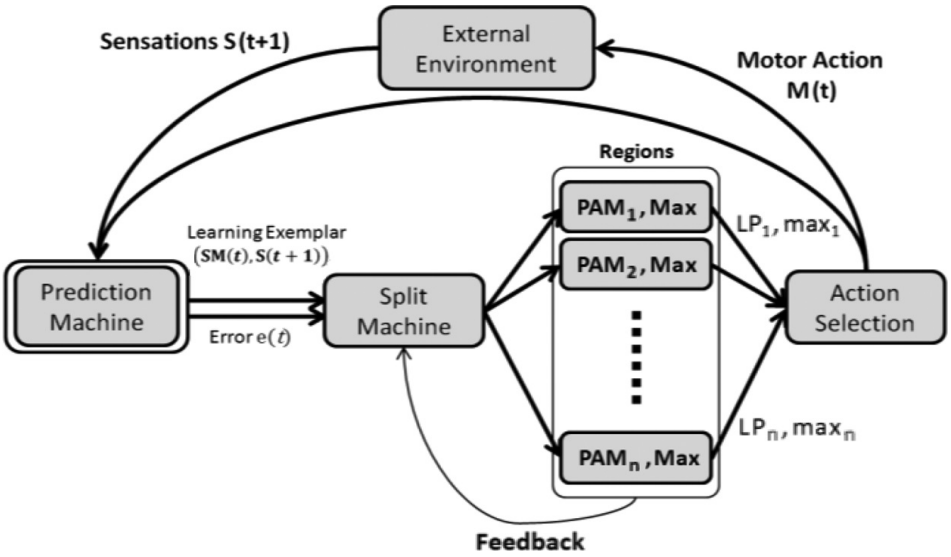


Figure 3.3: General architecture of IAC and R-IAC adapted frmo [Baranès & Oudeyer 2009]. The Prediction Machine PM is used to create a forward model of the world, and measures the quality of its predictions (errors values). Then, a split machine cuts the sensorimotor space into different regions, whose quality of learning over time is examined by Prediction Analysis Machines. Then, an Action Selection system is used to choose experiments to perform.

The key idea of IAC is that the drive to learn is based on maximizing the learning progress. This is achieved by creating a memory of all the exemplars $(SM(t), S(t+1))$ encountered by the robot and splitting this memory into similarity-based regions. Each region is characterized by its (disjoint) set of exemplars. Each region is associated with a specific learning machine, called an expert, that is responsible for the prediction of $S(t+1)$ given $SM(t)$ when $SM(t)$ is a situation which is covered by its associated region. More formally, the expert is trying to map the sensorimotor information at time t to the sensory outcome at time $t+1$:

$SM(t) \rightarrow S(t+1)$. Each region monitors the errors of its expert over time and generates a measure of learning progress, which is essentially the change in the current mean error rate (relatively to the previous mean error rate). The robot explores the regions that will expectedly yield the maximal learning progress.

On each time step the robot achieves action selection based on its current memory. It takes in a vector of the current sensory information and generates a list of potential actions (or a sample of potential actions if the list is infinite). Then it concatenates each candidate action with the current sensory information and probes the memory to find all matching regions. The region which gives the highest potential learning progress is selected after an ε-greedy mechanism (with some low probability ε, a random action is selected). The sensorimotor vector corresponding to the selected action is added to the selected region as an exemplar of the region.

When a region offers a learnable transition model (yielding from a sensorimotor context and an action to the next sensorimotor context), initially its expert will make good progress and this region will be chosen frequently. As the expert succeeds in learning the transition model, its progress will slow down, and the learning progress of other regions will outpass this region. In this way, IAC guides the robot to explore its environment in a sensible and adaptive way, focusing on those aspects where it can make the best gains, and ignoring aspects that have already been learned, or *are unlearnable*. It is important to note that IAC is robust with respect to non-deterministic transition functions.

The main processing loop of IAC works as follows (the specific algorithm will be given in next chapter 4):

- Let $S(t)$ be the sensor vector corresponding to the current situation.
- Create a list of potential actions. If the action space is continuous, generate a sample of candidate actions.
- For each candidate action, $M(t)$, query the IAC memory with $S(t).M(t)$, and determine the associated region's learning progress.
- With some high probability, choose the action associated with maximal learning progress in this region, otherwise choose a random action.
- Execute the chosen action on the robot and observe the outcome $S(t+1)$.
- Train the expert from the chosen region on the mapping: $SM(t) \rightarrow S(t+1)$.

Note that IAC could be viewed as active learning algorithms that are particularly suited for learning forward models in raw sensorimotor spaces with large unlearnable regions (due to locally very stochastic transition models for instance). Instead,

R-IAC is far superior to IAC in a complex sensorimotor space where only a small subspace is "interesting", i.e., neither unlearnable nor trivial [Baranès & Oudeyer 2009].

3.4.5 Hybrid approaches

The IAC and R-IAC intrinsically motivated active exploration methods suffer from the main two limitations of unlearnability and unboundedness [Oudeyer *et al.* 2013], respectively when the agent sensor vector is high-dimensional (in particular with continuous sensors) and when the agent faces an open environment. The SAGG-RIAC (Self-Adaptive Goal Generation R-IAC) [Baranes & Oudeyer 2010] proposes to address these challenges by hybridizing the IAC and R-IAC schemes with the Shifting Setpoint Algorithm, where models are built along tubes in the motion space between desired points [Schaal & Atkeson 1994]. They define a mutilevel active learning algorithms, using motor babbling to build the inverse model along tubes in actuator space, from start positions to goal positions.

3.5 Discovery

Most intrinsic motivation-related approaches have been assessed empirically, due to the lack of well-defined performance measure for autonomous learning agents, and the lack of theoretical framework supporting the closed-form analysis of their performances. In the particular case of [Auer *et al.* 2011, Lim & Auer 2012], such a theoretical framework and algorithms have been proposed and analysed. The context is that of an MDP without external rewards. [Auer *et al.* 2011] proposed a performance measure for such contexts, and [Lim & Auer 2012] designed an algorithm, referred to as **UcbExplore**, which explores a controlled Markov process by discovering reachable states.

3.5.1 Discovering reachable states in a controlled Markov process

The algorithm **UcbExplorep** aims at the discovery of reachable states, exploiting the fact that these states can be learned (i.e. discovered) incrementally. The main idea of the algorithm is to formulate the discovery of a reachable state as a task; in each iteration a task is selected after the *Optimism in front of the unknown* principle at the core of the Multi-Armed Bandit algorithms [Auer *et al.* 2002]. In each time step, a state is selected in an optimistic way. *By optimistic we mean choosing the easiest state to reach—the one that seems to be reachable in the shortest number of steps from the starting point s_0, based on information collected so far.* The algorithm maintains a set $\mathcal{K}$ of known states and a set $\mathcal{U}$ of unknown states. A state s is L-known when a policy π_s for that state can reaches s in $(1+\varepsilon)L$ steps, while $\mathcal{U}$ includes

all states which have been identified as candidate states, potential members of $\mathcal{K}$. The algorithm consists of the following steps, where L denotes the budget (number of time steps) allowed to reach states:

1. **State discovery**

   For each known state in $\mathcal{K}$ all actions are systematically explored, in order to discover all relevant neighboring states. As the environment admits incremental learning, either one of the unknown neighboring states is reachable, or all reachable states are already known.

2. **Compute optimistic policy**

   For an unknown neighboring state s_u, compute an optimistic policy π_{s_u} (consistent with the current observations so far) that reaches s_u with a minimum number of time steps. In case no unknown neighboring state is optimistically reachable in L steps, terminate the algorithm. Otherwise, choosing a state s_u and corresponding policy π_{s_u}, the algorithm goes to next step.

3. **Policy evaluation**

   The policy π_{s_u} is evaluated, then the policy π_{s_u} is executed several times (as the underlying transition model is assumed to be non-deterministic). If s_u is indeed reached in at most $(1+\varepsilon)L$ steps, s_u becomes a new known state and the algorithm goes to Step 1. If not, the algorithm continues by choosing another state s_u and its policy π_{s_u} in Step 2.

The most computationally intensive step of the algorithm is the last one, checking the quality of policy π_{s_u}.

3.5.2 Analysis of algorithm UcbExplore

Each major iteration of the algorithm is referred to as a "round". A successful round consists of i) finding a neighboring state (step 1); ii) finding a policy to reach it (step 2); iii) checking that this policy succeeds in circa L steps on average; upon success the new state is removed from $\mathcal{U}$ and added to $\mathcal{K}$. Algorithm **UcbExplore** thus incrementally discovers all states reachable in circa L states, continously upgrading its knowledge about the environment by tackling increasingly more complex goals, that is, discovering farther away states.

Note that this approach can be likened to the novelty search algorithm [Lehman & Stanley 2008], where the goal simply is to discover new individuls. More formally, [Lehman & Stanley 2008] uses as fitness objective the "distance" of an individual w.r.t. the individuals in the agent memory or archive. Accordingly, the agent

gradually gathers a sample of diverse individuals in its memory. Thereby, it creates an open-ended memory-based evolutionary framework, aimed at novelty [Lehman & Stanley 2011].

3.6 Discussion

As said, the work presented in this manuscript is at the crossroad of Evolutionary Robotics (ER, chapter 2) and Reinforcement Learning-based Robotics, described in this chapter.

The differences among the two approaches are threefold. A first issue regards their input and how the prior knowledge involved in the algorithmic process is provided by the human designer. A second issue regards the search space and the optimization process. A third, related and most important issue, regards how the information gathered along the search is reused.

3.6.1 Prior knowledge

In mainstream RL, the prior knowledge is expressed through the reward function defined on the state-action space. As noted, the reward function encapsulates a high expertise as it is responsible for ensuring that i) the associated optimal controllers actually achieve the desired behavior; and ii) the underlying optimization process can be efficiently conducted. The definition of an accurate reward function, thus an RL bottleneck, can be addressed using Inverse Reinforcement Learning, using the expert demonstrations to actually learn a reward function; IRL however requires the expert to know and be able to demonstrate the desired behavior. Preference-based RL, relaxing the expertise requirement, uses the human in the loop to incrementally learn an optimization objective defined on the policy space.

Evolutionary robotics, like direct policy search [Kober *et al.* 2013], starts with an optimization objective (also called fitness) defined on the policy space. The definition of a good fitness function raises similar difficulties as the definition of a reward function: the associated optimal controllers must achieve the desired behavior and the fitness function must induce a doable optimization problem. The main difference is that the fitness function is not subject to the Markovian assumption: it can consider the whole trajectory and use external, non-stationary information[6]. In particular, ER optimization objective can refer to external and non-stationary information; RL could hardly (or not tractably) do the same.

[6]For instance, it makes it possible to reward a controller for its first visit to a given location, but not for its subsequent visits. In a Markovian setting, this would require the state space to involve a specific feature, indicating whether this location has already been visited.

3.6.2 Optimization issues

Mainstream RL aims at learning and optimizing the value function on the state or state-action space; this optimization is alternated with the policy optimization. This value function induces the optimal policy: by greedily selecting the action with best value in the current state, or leading to the state with best value.

ER, like direct policy search, conducts optimization in the controller space, with two specificities. Most importantly, policy optimization corresponds to a stochastic optimization problem: find the controller maximizing the fitness expectation, taken over the trajectory distribution induced by the starting state, the transition model, the noise in the actions. The fitness expectation is approximated by an empirical average; an important issue is to keep the computational cost of this approximation within reasonable bounds [Heidrich-Meisner & Igel 2009].

In both cases, a key issue is to maintain a tradeoff between the representation of the search space (the state-action space, the value function on the state-action space, the controller space), which must be sufficiently rich to support complex behaviors, and the difficulty of the associated optimization problem [Koutník *et al.* 2013].

3.6.3 Knowledge gained along search

Both approaches exploit the data acquired through the learning or optimization process in a different way. In RL, the main acquired knowledge is the value function (and to a lesser extent the transition model), which is gradually refined along the process; the desired behavior then (trivially) derives by greedy optimization of the value function.

In ER, the acquired information is encapsulated by the controller population on the one hand (e.g. a distribution on the weight vector of a neural net architecture), and by an archive of the trajectories of past controllers on the other hand. This archive makes it feasible to define more sophisticated fitness functions, not satisfying the Markovian assumption. For instance, [Lehman & Stanley 2008, Lehman *et al.* 2012] characterize and exploit the difference between a trajectory and the past ones to enforce the robust sampling of the trajectory space for creative design; [Mouret & Doncieux 2012, Koos *et al.* 2013] likewise use this diversity, possibly along a multi-objective framework; [Delarboulas *et al.* 2010] further defines a discovery-driven fitness, computing the conditional entropy of the current trajectory w.r.t. the trajectory archive.

3.6.4 A hybrid goal

The above comparisons yield to identify desirable features and ask the corresponding questions. Firstly, describing the target controller through a value function appears to be desirable as far as it allows the value function to be modified along the controller lifetime, achieving lifelong learning. As with all incremental learning/optimization processes, this raises the question of the initialization of the process. A second requirement thus is that the initial value function should encapsulate a sufficiently global information on the environment, in order to support non-local action selection. A second question thus is how this "initial/global" information encapsulated in the initial value function should be gathered.

The next chapter, presenting the contribution of our work, is an attempt to answer these questions.

Chapter 4

The Ev-ITER approach

This chapter presents our main contribution to online, on-board robotics, at the crossroad of Evolutionary Robotics and Machine Learning-based Robotics [Hurst *et al.* 2002, Williams & Browne 2012, Koutník *et al.* 2013, Parra *et al.* 2014, Wang *et al.* 2015]. We first define our goal in Section 4.1, which is to provide the agent with an intrinsic motivation or "instinct", supporting the building of an autonomous exploratory controller. The desired properties of such an intrinsic motivation, focussing on the generality of the controller, are discussed. The proposed approach inherits from the intrinsic motivation [Baranès & Oudeyer 2009, Oudeyer *et al.* 2012] and the information theory-based [Delarboulas *et al.* 2010] approaches, respectively discussed in chapter 3 and chapter 2; these algorithms are described with unified notations for the sake of completeness and clarity (section 4.2). The proposed approach, called *Evolution and Information Theory-Driven Exploratory Robotics* (Ev-ITER) and first presented in [Zhang & Sebag 2014], is described in section 4.3. The chapter concludes with a discussion and some perspectives for further research.

4.1 Position of the problen

As detailed in the previous chapters, several disciplinary fields are concerned with building autonomous robotic controllers:

- Optimal control is concerned with model-based settings in continous domains[1];
- Machine learning and specifically reinforcement learning is concerned with model-based settings in discrete domains, and model-free settings in discrete or continuous domains [Kober *et al.* 2013], where the prior knowledge is usually provided in terms of a reward function attached to each state or state-action pair;
- Evolutionary robotics is concerned with the direct optimization of the controller (and also possibly of the robot architecture), assuming strong domain knowledge and computational resources; ER is most often used together

[1] As said, optimal control is outside the scope of the presented work.

with a simulator, at the expense of the so-called *reality gap* [Lipson & Pollack 2000, Saxena *et al.* 2008].

The last two disciplines present different trade-offs between the assumedly available prior knowledge on the one hand and the computational resources on the other hand. At one extreme, ER requires but a coarse prior knowledge (the controller performance is defined at the trajectory level) which is exploited using strong resources (computational resources when the controller is computed and optimized in simulation; human efforts when the controller is optimized *in-situ*). At the other extreme, RL requires a strong prior knowledge (the performance is defined at the fine-grained level of the state-action pair) and is expected to make a more parcimonious use of the computational resources (although algorithms with provable guarantees of convergence toward the optimum strategy raise scalability issues with respect to the size of the state and action space).

Exploratory robotics, as pioneered by [Schmidhuber 1991] and further investigated by [Oudeyer *et al.* 2007, Baranès & Oudeyer 2009, Montanier & Bredeche 2011a, Mouret & Doncieux 2012, Mouret & Doncieux 2012, Bredeche *et al.* 2012, Oudeyer *et al.* 2013] tackles yet another trade-off. On the one hand, it considers a model-free setting, where the transition model is unknown. On the other hand, a main challenge is to define *a priori* a reward either at the coarse trajectory or at the fine-grained state-action level, such that i) it does not require ground truth about the appropriateness of the robot behavior in its environment; ii) it enforces an "interesting" behavior. The research question investigated in this PhD work is to build exploratory robotic controllers, using an *in-situ* approach[2].

The proposed approach will hybridize two approaches stemming from Evolutionary Robotics and from Reinforcement Learning, with the goal of getting the best of both worlds:

- Evolutionary robotics will be used to build primary controllers, referred to as *crawling controllers*, using and extending (section 4.3.1) [Delarboulas *et al.* 2010] for a few generations;
- The sensori-motor data gathered by the crawling controllers is used to provide prior knowledge to secondary controllers, referred to as Ev-ITER controllers,

[2] As said, this study is primarily motivated by swarm robotics [O'Dowd *et al.* 2011, Brambilla *et al.* 2013]. Swarm robotics aims at designing robust, scalable and flexible collective behaviors for the coordination of large numbers of robots through simple controllers and local interactions. In this context, the standard simulator-based approach is ineffective. On the one hand, the computational complexity is super-linear with respect to the number of robots in the swarm; the environment is highly dynamic due to the fact that the actions of the robots in the systems are coupled with one another. On the other hand, the simulator accuracy is hindered by the variability of the hardware.

taking inspiration from the intrinsic robust motivation [Schmidhuber 1991, Oudeyer *et al.* 2007, Baranès & Oudeyer 2009, Oudeyer *et al.* 2012]; the Ev-ITER controllers are essentially deterministic[3];

- Furthermore, the generality of the Ev-ITER controllers will be tested, considering new environments different from the training environment.

Overall, the goal of Ev-ITER controllers is to be able to explore the training environment and similar environments in a principled and effective way. The main contributions of the presented approach are the following.

- Firstly, it acknowledges that a preliminary exploration of the environment is needed to prime the pump and start gathering information. This preliminary exploration corresponds to the initial stage of *babbling* in [Oudeyer *et al.* 2007, Baranès & Oudeyer 2009], uniformly selecting the move actions and thus following a Brownian motion.
  In Ev-ITER, this preliminary exploration is achieved through launching a very short evolutionary robotic process.
  A main point is that the controller trajectories obtained by this first phase are exploited, in contrast with the fact that ER does not exploit *per se* the information gathered by the controllers; the controller trajectories are thrown away as they only serve to compute the fitness of the controllers (although some specific information, e.g. the end of the trajectory, is archived in the Novelty Search approach [Lehman & Stanley 2008, Mouret & Doncieux 2012]).
  Overall, this first phase aims at building data resources, supporting the further computational stages, under the constraints of limited memory and computational resources; the criterion is to be more effective than a Brownian movement in building this data repository.

- Secondly, this data repository is exploited by a principled and essentially deterministic stragegy (although, as said, it is slightly mixed with a random controller in order to avoid getting stuck in dead ends depending on the environment).

- Thirdly and importantly, this strategy is effective in other environments than the training one. It must be emphasized that the generality of the learned controller with respect to the considered environment has rarely been considered, neither in the reinforcement learning[4], nor in evolutionary robotics.

[3] The mixing of the deterministic strategy with a uniformly random one (selecting uniformly the actions with a very low probability) is used to prevent the robot from getting stuck in dead ends, depending on the environment.

[4] Transfer reinforcement learning (see e.g. [Taylor & Stone 2009, Konidaris *et al.* 2012] among

4.2 Formal Background

For the sake of clarity and using unified notations, this section recaps the algorithms of intrinsic motivation [Oudeyer *et al.* 2007,Baranès & Oudeyer 2009,Oudeyer *et al.* 2012,Lopes *et al.* 2012] and curiosity-driven discovery [Delarboulas *et al.* 2010].

4.2.1 Notations

- $K = (s_t, a_t)_{t=1}^{T}$ denotes a T-length trajectory, where s_t (respectively a_t) denotes the sensor (resp. motor) value vector at time t ($s_t \in \mathbb{R}^s$, $a_t \in \mathbb{R}^m$).
- s_t^i (respectively a_t^i) denotes the i^{th} vector value of s_t (resp. a_t).
- $sm_t = (s_t, a_t)$ denotes the concatenation of the sensory and motor data of the robot at time t, i.e., the sensori-motor value vectors at time t.
- S and A respectively stands for the set of states and set of actions.
- $\mathcal{K}_{\to g} = \{K_1, \dots K_g\}$ denotes the archive of the trajectories generated by the learning/optimization process until step g.

4.2.2 Intrinsic motivation

Let $\mathcal{K}_{\to i}$ denote the archive of the first i trajectories generated by the intrinsic motivation process. A forward model f_i is learned from $\mathcal{K}_i$, estimating the transition model in the robot environment, specifically the next state s' of the robot after selecting action a in state s:

$$f_i : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S} \tag{4.1}$$

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_{i+1} thus defines an intrinsic information, accessible to the robot without any external ground truth.

$$Acc(f_i) = Pr(s_{t+1} = f_i(s_t, a_t) | (s_t, a_t, s_{t+1}) \in K_{i+1}) \tag{4.2}$$

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

many others) is mostly concerned with how to *adapt* a solution controller under variations of the reward or transition models.

$$\mathcal{F}_{IM}(\pi) = Acc(f_{i+1}) - Acc(f_i) \tag{4.3}$$

Some hyperparameters are involved in the computation of the fitness, typically specifying how the accuracy is tested (size of the test set K_{i+1}, size and selection of the training set extracted from $\mathcal{K}_{\rightarrow i}$ used to learn f_i); these parameters critically control the variance of the accuracy and therefore the noise of $\mathcal{F}_{IM}$. The state and action spaces usually are discretized in a regular, recursive way, where each grid cell is refined along the learning and evolution process to split the grid cells where the prediction accuracy remains consistently low.

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 after sufficient exploration, and thus yield a null contribution to $\mathcal{F}_{IM}$.

4.2.3 Curiosity-driven Evolutionary Robotics

Curiosity-driven ER, first introduced by [Delarboulas *et al.* 2010], likewise exploits the past trajectories to define a fitness function. Likewise, it considers a continuous state and action space. The discretization is achieved incrementally using a deterministic process, meant to enforce a steady fitness function (below).

4.2.3.1 Clustering the sensori-motor space

This discretization is based on a standard clustering algorithm [Duda *et al.* 2001] with linear complexity in the size of the trajectories. The basic clustering algorithms are the k-means (Alg.3) and the ε-clustering algorithm (Alg. 4). Let $\mathcal{E}$ be a set of n points $\{x_1 \dots x_n\}$ in a space X endowed with a metric or dissimilarity function d.

The k-means algorithm specifies the number of clusters, set to k. Each cluster C_i is initialized with a (uniformly or heuristically chosen) point in $\mathcal{E}$. The k-means algorithm then proceeds by incrementally associating a point x_i to the closest cluster (where the distance of a point x to a cluster C is a hyper-parameter of the algorithm, e.g. considering the minimum or the average distance between x and the points in C [Kleinberg 2003]. As the final set of k clusters heavily depends on the initialization of the algorithm and on the order of the points, k-means usually iterates the above process, where each round starts by taking as initial points the average or median point in each cluster found in the previous round, and the process is shown to

converge toward a (local) minimum (e.g. of the distortion function in the case where the distance between a point x and a cluster C is the average over x_i in C of $d(x, x_i)$, where the distortion of sample $\mathcal{E}$ w.r.t. clustering $\mathcal{C} = \{C_1, \ldots C_k\}$ is defined as:

$$D(\mathcal{E}) = \sum_{i=1}^{n} d^2(x_i, \mathcal{C}) \qquad d(x, \mathcal{C}) = min_{j=1}^{k} d(x, C_j) \tag{4.4}$$

While k-means specifies the number of desired clusters (which might thus have very diverse size depending on the sample and on the initialization), parameter ε specifies the maximum diameter of a cluster. The ε-clustering algorithm sets the first point x_1 to the first cluster. Iteratively, each point x_i is compared to the existing clusters; if the nearest cluster is within a distance less than ε of x_i, x_i is allocated to this cluster; otherwise, a new cluster is created and initialized to $\{x_i\}$.

The reason why ε-clustering is preferred over k-means is its stability with respect to a dynamically extended sample $\mathcal{E}$. The distortion of the clustering is bounded by $n\varepsilon$ by construction, and the algorithm does not require several rounds in order to converge toward a local optimum of the distortion as is the case for the k-means algorithm. This property allows to gradually extend the clustering along evolution, as more trajectories are considered and other regions of the sensori-motor space are visited, while enforcing a stationary fitness function based on the clusters (see below).

The price to pay for this stability property under the extension of $\mathcal{E}$ is that the number of clusters yielded by ε-clustering is unbounded. In the worst case, it varies exponentially in $\mathcal{O}(\varepsilon^d)$ if points x_i belong to a bounded region of $\mathbb{R}^d$. In our motivating application, the designer must thus choose ε after some preliminary trials to ensure that the ε-clustering algorithm complies with the bounded memory resources of the robot. Conditionally to a decent number of clusters, the ε-clustering algorithm complies with the bounded computational resources of the robot, and can thus be run online, on-board.

4.2.3.2 Clustering-based evolutionary fitnesses

Let $\mathcal{C}$ be the ε-clustering built from the controller trajectories recorded so far.

Let K be a controller trajectory. For each cluster C_i in $\mathcal{C}$, let p_i denote the fraction of sensori-motor vectors sm_t in K belonging to this cluster. The entropy of trajectory K is classically defined as:

$$\text{Entropy}(K) = -\sum_{C_i \in \mathcal{C}} p_i \log p_i \tag{4.5}$$

Algorithm 3 k-means Algorithm

1: **Input:** k the desired number of clusters; $\mathcal{E} = \{x_1, \dots x_n\}$ points.
2: **Output:** $\mathcal{C} = \{C_1, \dots C_k\}$ clustering
3: **for** $i = 1...k$ **do**
4: $C_i = \{c_i\}$ where c_i is a uniformly chosen point in $\mathcal{E}$ with no replacement (all distinct)
5: **end for**
6: **for** $i = 1...n$ **do**
7: $j(i) = argmin_{j=1...k}\{d(x_i, C_j)\}$
8: **end for**
9: repeat
10: **for** $t = 1...k$ **do**
11: $C_i = \{c_i = \frac{\sum_{k \ s.t. \ j(k)==i(t)=i} x_k}{\sum_{k/j(k)=i} 1}$
12: **end for**
13: goto line 4
14: until C does not change

Algorithm 4 ε-means Algorithm

1: **Input:** ε the desired diameter of a cluster; $\mathcal{E} = \{x_1, \dots x_n\}$ points.
2: **Output:** $\mathcal{C} = \{C_1, \dots C_k\}$ clustering
3: $\mathcal{C} = \{C_1\}, C_1 = \{x_1\}$;
4: **for** $i = 2...n$ **do**
5: $j(i) = argmin_{C_j \in \mathcal{C}}\{d(x_i, C_j)\}$
6: **if** $d(x_t, C_{j(i)}) > \varepsilon$ **then**
7: $\mathcal{C} \bigcup = C = \{(x_i\}$;
8: **end if**
9: **end for**

The curiosity-driven fitness $\mathcal{F}_E$ associates to a controller π the entropy of a T-long trajectory thereof. Note that this fitness yields a stochastic value, due to two phenomenons: Firstly, the trajectory entropy depends on the initial position of the robot. In many studies, the experimental setting requires that all robot trajectories start in the same position to limit the variability; while such a requirement is easily met in simulations, some human effort is required to fulfill this condition for *in-situ* experiments, and one would rather like the robot trajectory with a given controller to start where the trajectory with the previous controller ended, as in open-ended evolution [Montanier & Bredeche 2011b]. An alternative is to consider long trajectories, in order to decrease the dependency on the initial state and discard the beginning of the trajectory, akin the burn-in period of a Markov chain.
Secondly, the sensori-motor trajectory is affected by the sensor and actuator noise. It is argued however by [Delarboulas *et al.* 2010] that the fitness is robust w.r.t experimental noise (e.g. when the sensor value suddenly jumps to the maximum value and gets back to its value afterwards), as p_i is by construction robust w.r.t. outliers.

The curiosity fitness of a controller finally can be computed on-board, with limited computational and memory resources (the memory resources being controlled depending on parameter ε). The clustering $\mathcal{C}$ is extended as more trajectories and more controllers are considered; the point is that the fitness defined from this non-stationary clustering is consistent, i.e. would give the same fitness value for a trajectory, whatever the generation this trajectory has been observed.

The incremental stability of the clustering is used to define another fitness, dubbed discovery-driven fitness [Delarboulas *et al.* 2010]. Let m_i define the number of sensori-motor vectors in the evolutionary robotic archive $\mathcal{K}$, falling in cluster C_i, with $m = \sum_i m_i$, and $q_i = m_i/m$. The entropy $-\sum q_i log(q_i)$, called population entropy, reflects how the past trajectories have been exploring the sensori-motor space. It then makes sense to consider how much additional information an individual trajectory brings in, relatively to the information gathered by the previous generations. The differential entropy of a T-long trajectory K, noted $DiffEntropy(K)$, is defined with same notations as above, as:

$$DiffEntropy(K) = -\sum_{C_i \in \mathcal{C}} \left(\frac{m_i + p_i T}{m+T} \log \frac{m_i + p_i T}{m+T} - \frac{m_i}{m} \log \frac{m_i}{m} \right) \tag{4.6}$$

The discovery-driven fitness $\mathcal{F}_D$ associates to a controller π the differential entropy of a T-long trajectory thereof. The intuition is that the discovery driven fitness defines a dynamic optimization landscape, where a controller is rewarded for discovering new clusters of sensori-motor states, and for visiting more the clusters

	IM	Curiosity	Discovery
No ground truth	$*$	$*$	$*$
Limited memory	(training set; test set)	clusters	clusters, # visits
Limited computation	$*$	$*$	$*$
Robustness	$*$	$**$	$**$
Sensitivity w.r.t. environment	$-$	$*$	$-$
Sensitivity w.r.t. hyper-parameters	$--$	$-$	$-$

Table 4.1: Respective merits of Intrinsic Motivation, Curiosity-Driven and Discovery-Driven Criteria. Legend: $*$ means that the criterion is satisfied; $-$, that it is not.

which were rarely visited by its ancestors. Again, this cumulative fitness definition relies on the fact that all controllers share the same clusters, and that the evolutionary mechanism maintains and updates the set of clusters and the number of visits they have received along all previous generations.

Both fitness objectives are maximized using evolution strategies: the (1+1)-ES is used online, on-board.

4.2.4 Getting the best of both worlds

Our goal, defining an optimization objective conducive to building efficient exploratory controllers, is associated with several requirements; these requirements are listed below, and how they are fulfilled by the three above criteria – intrinsic motivation, curiosity and discovery – is reported in Table 4.1.

1. No ground truth required; the criterion must require no prior knowledge, or truth signal obtained through a complex experimental setting (e.g. light signalling that the robot is doing well);

2. The criterion must be computable on-board: it must be compatible with limited memory and computational resources;

3. The criterion must be robust w.r.t sensor and actuator noise;

Additional aspects include the sensitivity w.r.t. the experimental setting and specifically the robot arena, and the sensitivity w.r.t. the hyper-parameters of the algorithm.

By construction, none of the three criteria requires any ground truth external to the robot; the ground truth signal is provided by the robot environment itself, through the sensor information. Regarding the memory requirements, intrinsic motivation must store enough sensori-motor vectors to enable building and assessing the forward model: as the IM criterion is defined by the increase in the forward model accuracy, a rather precise assessment is required, implying large training and test sets. W.r.t. the curiosity and discovery-driven fitness, one only requires to store the cluster centers (for curiosity), plus the number of times they have been visited (for discovery). These requirements are quite compatible with the limited memory resources available on-board, provided that the ε parameter is set to an appropriate value.

Regarding the computational effort, the clustering-based approaches required very limited computational effort; the intrinsic motivation approach requires to embed and launch a machine learning algorithm, which might be more expensive; still, decision trees or random forests can be used to achieve fast learning with limited computational cost.

Regarding the robustness wrt sensor noise, all approaches are robust; the higher robustness of the clustering-based approaches is due to the fact that events with low probability p would contribute p to IM fitness, and $plogp$ to clustering-driven approaches.

The robustness w.r.t. algorithm hyper-parameters, also an important aspect for reproducible experiments and further transfer to industrial partners, raises complex questions. Learning a transition model, like all learning tasks, notoriously significantly depends on the choice of the learning algorithm and the calibration of its hyper-parameters [Hutter *et al.* 2015]; it is true that intrinsic motivation only involves the increase or decrease of the transition accuracy; still, the bad choice of the learning hyper-parameters (and of the size and selection of the training and test sets) could impact $\mathcal{F}_{IM}$, adversely affecting the signal to noise ratio, and thus harming the exploratory process. In the clustering-based approaches, one must adjust the only parameter ε. The proper adjustment of ε commands the whole process: too high and there will be a single cluster, making the fitness a trivial one; too low and the number of clusters becomes very large (or infinite in a truly continuous sensor space), also yielding a trivial fitness. Parameter ε must be adjusted to match the information richness of the environment. If the environment does not present sufficient variations (a desert), the fitness also is trivial and return 0 (the robot sees a single state). Quite the contrary, if ε is too large and distinct sensori-motor vectors falls in distinct clusters, then the fitness also trivially returns $\log T$, T being the length of the trajectory.

A weakness of the curiosity-driven approach is its sensitivity to stochastic environments and periodic behaviors: a controller exhibiting periodic behaviors with a long period (e.g. dancing in a corner of the environment) will get a high fitness. Quite the contrary, the intrinsic motivation is immune as said to stochastic transition models, and will also stand heteroscedastic environments where the noise of the transition model varies depending on the region of the space, by simply exploring more the regions with higher transition noise. The discovery-driven approach, which rewards the discovery of unvisited or rare states, is also less sensitive than the curiosity-driven approach than stochastic environments.

Let us finally consider the generality w.r.t. environments. In a new environment, the transition model basically needs be learned from scratch, and the intrinsic motivation approach must therefore starts anew. For the curiosity-driven approach, if the state and action spaces do not change, then the process could continue, reuse and extend the available clustering (but ε-clustering is not very computationally expensive anyway). In the discovery-driven process, one would rather restart the process as the global entropy gathered in a previous arena is hardly relevant for a new arena.

4.3 Ev-ITER overview

This section describes the Ev-ITER scheme, aimed at the best of the two intrinsic motivation and clustering-based approaches. Ev-ITER involves three phases:
The first phase builds robotic crawlers, built by evolutionary robotics by taking inspiration from [Delarboulas *et al.* 2010], and considering a training environment.
The second phase runs the best robotic crawlers built in the first phase, in the training arena, and a data repository made of triplets (state, action, next state) is built from their trajectories.
In the third phase, the data repository is used and updated to support the quasi-deterministic Ev-ITER controller, operating either in the training environment – referred to as source environment – or in another environment – referred to as target environment.

4.3.1 Phase 1: Building robotic crawlers

The robotic crawlers, like in [Delarboulas *et al.* 2010], are multi-layer perceptrons neural networks (Fig. 4.1). The input nodes receives the sensor values. The values of the hidden nodes on the intermediate layer are computed from the input node

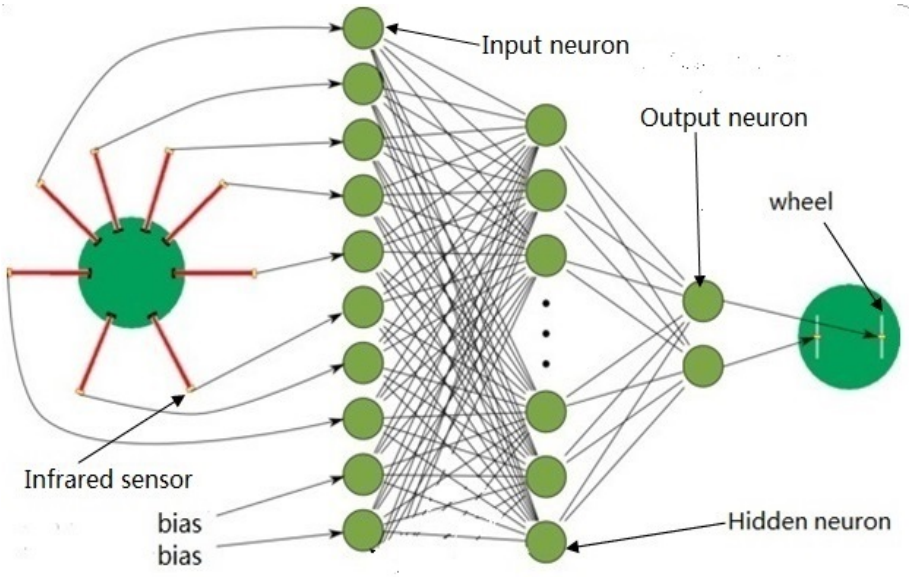


Figure 4.1: The Neural Network architecture of the robotic crawlers: the robot sensor values on the input nodes are mapped onto the hidden node values, and these are mapped onto actuator values.

values using a non-linear transformation of a linear combination of the input:

$$h_j(\mathbf{x}) = g\left(\sum_{i=1}^{s+1} w_{i,j}x_i\right) \tag{4.7}$$

where $\mathbf{x} = (x_1, \ldots x_s)$ is a sensor vector, w_i, j are the first layer weights of the NN, ($\mathbf{w}$ in $\mathbb{R}^{(s+1)\times h}$ where s is the number of sensors and h is the number of hidden neurons[5], and g is a non-linear bounded activation function, usually the sigmoid function (Fig 4.2):

$$g(z) = \frac{1}{1 + exp^{-a\cdot z}}$$

The hidden nodes are likewise used to compute the actuator values, with

$$a_j(\mathbf{x}) = g\left(\sum_{i=1}^{h+1} w'_{i,j}h_i\right) \tag{4.8}$$

[5]By convention, the sensor vector in $\mathbb{R}^s$ is embedded onto $\mathbb{R}^{s+1}$ by concatenation with a constant value 1. In this way, the bias b in the standard equation

$$y = g(\sum_{i=1}^{s} w_i x_i + b)$$

is represented as the last coordinate w_{s+1} of the weight vector $\mathbf{w}$.

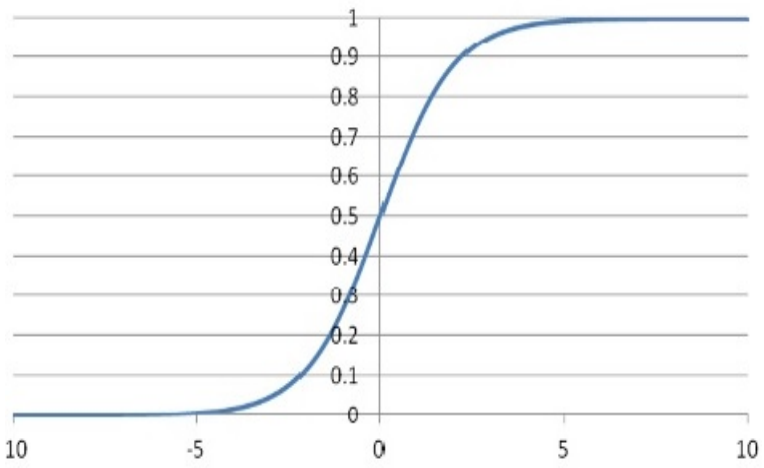


Figure 4.2: The sigmoid function.

with w'_i, j the second layer weights of the NN, in $\mathbb{R}^{(h+1)\times m}$ and $\mathbf{a} = (a_1, \ldots a_m)$ the vector of actuator values of the robot.

The first phase in Ev-ITER thus optimizes the weight vector of the robotic crawlers, with search space $\mathbb{R}^d$ and $d = (s+1) \times h + (h+1) \times m = h(s+m+1)+m$.

Three optimization modes are considered. Besides the curiosity-driven and discovery-driven objectives detailed in the previous section, another objective is considered, the entropy of the output node vectors $\mathbf{a}$. Formally, let respectively $\mathbf{h}_t$ and $\mathbf{a}_t$ denote the hidden node vector and the output node vector computed from the sensor node s_t; both vectors capture the diversity of the sensor value vector $\mathbf{s}_t$, and are responsible to generate the actual moves of the robot. The algorithm thus records the $\mathbf{a}_t$ vectors for $t = 1 \ldots T$, clusters them using ε-clustering, and uses their entropy as optimization objective.

The rationale for considering the entropy of the output nodes[6] as a good incentive for exploratory behaviors is as follows. On the one hand, the hidden nodes constitute a compressed representation of the sensor vector. In particular, the dimension of the hidden node vector is a hyper-parameter of the approach, under the control of the design engineer, whereas the dimension of the sensor vector can be very large (typically if the robot is equipped with cameras with a few thousand or million pixels). Likewise, the output nodes constitute a compressed representation of the hidden nodes, and their dimension is fixed: the number of actuators is much smaller than the number of sensors.

On the other hand, the hidden nodes are predictive of the actuator values; the

[6] Another possibility is to consider the entropy of the hidden nodes. This perspective is left for further research.

diversity of the actuator vectors $\mathbf{a}_t$ depends (through weights $\mathbf{w}$') on the diversity of the hidden node vectors $\mathbf{h}_t$. The only remaining question is why maximizing the entropy of the output nodes can be conducive to an exploratory behavior. This is explained by considering the entire chain of perception / hidden node / action: $\mathbf{s}_t$ reflects the position of the robot at step t; $\mathbf{h}_t$ is deterministically computed from $\mathbf{s}_t$; $\mathbf{a}_t$ is deterministically computed from $\mathbf{h}_t$; s_{t+1} depends on $\mathbf{s}_t$ and $\mathbf{a}_t$, through the transition model. Assuming that a high diversity (entropy) of the $\mathbf{s}_t$s is associated with an efficient exploratory behavior, it follows that a high diversity of the hidden nodes $\mathbf{h}_t$s and of the output nodes $\mathbf{a}_t$ also is associated with a high diversity of the sensor vectors s_t.

Overall, three optimization criteria are considered in Phase 1: the curiosity-driven mode, referred to as sensor-entropy (SE), the discovery-driven mode, referred to as sensor-differential-entropy (SDE), and the entropy of the output nodes, referred to as actuator-entropy (AE).

Phase 1 uses a $(1 + 1)$-Evolution Strategy maximizing the chosen criterion for N generations, with N lower by an order of magnitude than used in [Delarboulas *et al.* 2010].

4.3.2 Phase 2: Building a data repository

In Phase 2, the best controllers with respect to the considered optimization objective are launched in the training arena, and their trajectories are recorded in the trajectory archive $\mathcal{K}$. For the sake of notational simplicity, it is assumed in the following that $\mathcal{K}$ involves a single trajectory of length T. The trajectory archive is used to initialize a data repository as follows.

Firstly, the sensor vectors (respectively the actuator vectors) are discretized. Let n_s (respectively n_a) denote the number of clusters obtained by ε-clustering in Phase 1. The clusters built by ε-clustering are not well-suited to Phase 2 and Phase 3 (more on this below). For this reason, P independent k-means algorithms with $k = n_s$ (respectively $k = n_a$) are launched in parallel on the sensor vectors (resp. the actuator vectors), and the best clustering in terms of distortion (Eq. 4.4) out of the P clusterings is retained.

Each (real-valued) trajectory $(s_0, a_0, s_1 \dots s_T)$ in the trajectory archive $\mathcal{K}$, with $s_t \in \mathbb{R}^s$ and $a_t \in \mathbb{R}^m$, is converted into a sequence of integers $(i_0, j_0, i_1, j_1 \dots i_T$ with i_t the index of the sensor cluster s_t belongs to, and likewise j_t the index of the actuator cluster a_t belongs to. The robot is said to execute action j in state i when $i_t = i$ and $j_t = j$.

To each pair i, j is associated the list $Z(i, j)$ of all instants t following the exe-

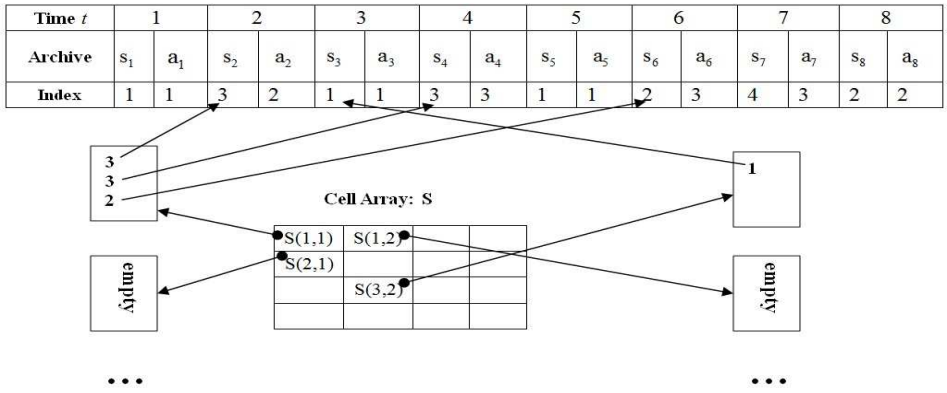


Figure 4.3: Computing the Q function from a 8-length trajectory (top), with $n_s = n_a = 4$. The 4×4 matrix S is built, where list $S(i,j)$ is used to compute entropy $Q(i,j)$ when not empty.

cution of action j in state i:

$$Z(i,j) = \{t, 0 \leq t \leq T, i_{t-1} = i \text{ and } j_{t-1} = j\} \subseteq \{1 \dots T\}$$

Let $S(i,j)$ denote the (multi-set) of state cluster indices for t in $Z(i,j)$ (note that a cluster index can appear several times in $S(i,j)$):

$$S(i,j) = \{i_t, t \in Z(i,j)\}$$

Let finally $Q(i,j)$ denote the entropy of $S(i,j)$. Denoting $n_{i,j,k}$ the number of times state cluster index k appears in $S(i,j)$ and $n_{i,j}$ the sum of $n_{i,j,k}$ for k ranging over the state cluster indices:

$$Q(i,j) = -\sum_k \frac{n_{i,j,k}}{n_{i,j}} \log \frac{n_{i,j,k}}{n_{i,j}}$$

It is clear that the higher $Q(i,j)$, the lesser predictable the next state of the robot upon selecting an action falling in the action cluster j in a state falling in the state cluster i. By slight abuse of notations, in the following we shall speak of state i (respectively action j) instead of state s (resp. action a) falling in the state cluster of index i (resp. action cluster of index j).

Tables $Z(i,j)$ and $S(i,j)$ are built and maintained online (Fig. 4.3) as First-In First-Out registers; they can be thought of as a transition model implemented as a look-up table. 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, with λ a user-specified parameter.

4.3.3 Phase 3. The Ev-ITER controller

The Ev-ITER controller is defined as a mixed strategy, hybridizing a pure random controller and a deterministic controller aimed at increasing the information stored in table Z.

The pure random mode, referred to as **babbling mode**, is triggered in two cases: when there is not enough information available in state i to compute statistically significant $Q(i,j)$, when the sum of $n_{i,j}$ with j ranging over the action cluster indices is less than a user-specified parameter λ'; and with a low η probability to prevent the degenerate behaviors incurred by a deterministic controller (below) depending on the specificities of the robot arena. The selected actuator index j^* is thus randomly selected in $\{1,\ldots,n_a\}$.

The other and main mode of the Ev-ITER controller referred to as **Ev-ITER mode** proceeds by deterministically selecting the action maximizing the following score function:

$$j^* = \text{ arg max } \{score(j|i) = (1-\alpha)Q(i,j) + \alpha\left(1 - \frac{n_{i,j,i}}{ni,j}\right), j = 1\ldots n_a\} \quad (4.9)$$

The rationale of the above score is to select the action resulting in a maximum uncertainty (maximum entropy) about the next state; a secondary criterion enforces the selection of an action such that it leads to a new state, since $\frac{n_{i,j,i}}{n_{i,j}}$ is the estimated probability that selecting action j in state i results in staying in state i. Hyper-parameter α controls the balance between the two terms: increasing the local information about the transition model in state i, and changing state.

In the two cases, letting j^* the selected actuator cluster index, the Ev-ITER controller runs an actuator vector $a_t \in \mathbb{R}^a$ which is uniformly selected in the actuator vector falling in the actuator cluster index j^*.

The pseudo-code of the Ev-ITER algorithm is displayed below (Alg. 5). In each time step, the $Z(i,j)$ and $S(i,j)$ lists are updated. Formally, upon selecting action j in state i at time t, the oldest elements in $Z(i,j)$ and $S(i,j)$ are removed if $n_{i,j} = \lambda$ and indices $t+1$ and i_{t+1} are respectively added to $Z(i,j)$ and $S(i,j)$.

4.3.4 Assessment of Ev-ITER controllers

As already said, while the robotic crawlers are trained in one robotic arena (the source environment), a main ambition of the Ev-ITER approach is to build controllers with good exploratory skills in new arenas, referred to as target environments, considered in Phase 3. Accordingly, two measures of performance of the exploratory Ev-ITER controller will be considered:

- The first measure of performance is the coverage of the target environment. Considering a fixed grid of the environment, the density of exploration is depicted by the cumulative density of visits to the grid cells, with $V(n)$ denoting the fraction of grid cells receiving at least $V(n)$ visits during Phase 3.
- The second measure of performance regards the accuracy of the transition model of the target environment, estimated from the $S.(i,j)$. More specifically as the target environment might also be a stochastic environment, the accuracy of the transition model is measured from the distance between the true stochastic transition model $p*(i,j,k)$, and the estimated one $\hat{p}(i,j,k) = \frac{n_{i,j,k}}{n_{i,j}}$, using a KL divergence:

$$\text{Accuracy } = KL(p||p^*) = \sum_{i=1}^{n_s}\sum_{j=1}^{n_a}\sum_{i=1}^{n_s} p(i,j,k)\log\frac{p(i,j,k)}{p^*(i,j,k)}$$

Algorithm 5 The Ev-ITER controller

Input: state i, registers $Z(i,j)$, parameters λ, η and α
Output: actuator vector a
if $\exists j \in 1 \ldots n_a$ *s.t.* $n_{i,j} < \lambda$ **then**
  $j^* = j$ (breaking ties at random)
else
  if With probability η **then**
    $j^* \sim U\{1 \ldots n_a\}$ (uniform selection of action index) ($p_1 = 5\%$)
  else
    $j^* =$ arg max $\{score(j|i) = (1-\alpha)Q(i,j) + \alpha\left(1 - \frac{n_{i,j,i}}{ni,j}\right), j = 1 \ldots n_a\}$ ($p_2 = 95\%$)
  end if
end if
return a uniformly selected in actuator cluster of index j^*

4.4 Summary and Discussion

As said in the preamble of this chapter, many algorithms aimed at building autonomous robotic controller ambition at getting the best of both worlds of Machine Learning and Evolutionary Robotics: see among others [Hurst *et al.* 2002, Williams & Browne 2012, Koutník *et al.* 2013, Parra *et al.* 2014, Wang *et al.* 2015]. The main specificities of the proposed Ev-ITER approach is twofold.

Firstly, the point is to be able to run online, on-board with no ground truth and no human intervention; in contrast, the cited algorithms involve some ground truth

information in order to compute the exploration indicators (e.g. when applied for simultaneous localization and mapping in [Williams & Browne 2012]).

Secondly, the deterministic Ev-ITER controller in Phase 3 can be run in target environments whereas the evolutionary training is done in a source environment, thus featuring some generality w.r.t. the robotic environments. The property of transferability from one environment to another one is particularly important to deal with hostile environments: the expected benefit is to minimize the exploration time needed to build a map of the target environment. Another expected benefit is to have the Phases 1 and 2 taking place in simulation, while Phase 3 takes place *in-situ*.

This approach presents some limitations and open questions, listed below and defining perspectives for further research:

- The most important algorithmic limitation in our opinion is to have the sensor and actuator clusters fixed once for all after Phase 2. They should be allowed to evolve and be refined along time, as in [Baranès & Oudeyer 2009, Oudeyer *et al.* 2012], all the more so when the target environment significantly differs from the source environment. The critical issue naturally is of doing so with a fixed memory budget.

- An open question is whether the criterion used in the deterministic Phase 3 should consider the transition entropy as in [Delarboulas *et al.* 2010], or rather the entropy reduction as in [Baranès & Oudeyer 2009, Oudeyer *et al.* 2012].

Chapter 5

Experimental Analysis

This chapter presents the experimental validation of the Ev-ITER algorithms described in the previous chapter. Firstly, sections 5.1 and 5.2 respectively discuss the goals of experiments and introduces the experimental settings and the four modes used in Ev-ITER. Section 5.3.2 reports on the performances of the four modes used in the evolutionary 1st-phase of Ev-ITER. The chapter last reports and discusses the experimental results of Ev-ITER comparatively to the baseline algorithms, where the Ev-ITER-1st phase respectively consider the entropy of the sensori-motor data (section 5.3.3) and the actuator data (section 5.3.4).

5.1 Goals of experiments

As said in the previous chapter, the presented work takes inspiration in both the evolutionary optimization of exploratory controllers [Delarboulas *et al.* 2010], and the use of the recorded data in order to define an exploratory controller taking inspiration from the intrinsic motivation scheme [Lopes *et al.* 2012]. Accordingly, the primary goal of the experiments is to assess whether the coupling of the two approaches, achieved by Ev-ITER, can improve on the performances of each method used as standalone. Several exploration indicators will be defined (section 5.2.4) and used for the quantitative comparison of all approaches, considering their diverse modes (Table 5.1).

A second goal is to assess the generality of the resulting controllers. Generality is the touchstone for machine learning approaches, meant as the hypothesis or model learned from given data (referred to as training set) must achieve comparable performances in expectation, and in practice, on new data (referred to as test set) generated from the same distribution as the training set. The generality criterion considered in this chapter (for which quantitative indicators are presented in section 5.2.4), differs from the usual generality assessment used in the machine learning litterature [Sutton & Barto 1998], in the following sense:

- A first arena (referred to as training arena or training environment) will be considered in the 1st-phase of Ev-ITER;

- A second arena (referred to as target or test arena or environment) will be considered in the 2nd phase of Ev-ITER.

This generality assessment procedure is original in the sense that ML and specifically reinforcement learning does never consider, to our best knowlwedge, the direct transfer of a controller from one environment to another one, since different Markov Decision Process (in particular considering different transition models or reward functions) correspond to different optimal controllers by construction. The litterature on transfer reinforcement learning focuses on how to *adapt* a controller learned in a given MDP, to another one [Konidaris *et al.* 2012, Zhan & Taylor 2015].
In the Ev-ITER case, the change of environment takes place between the first and the second phases. The first phase involves the evolutionary optimization of a controller in the training arena, and the building of an archive recording some trajectories of the best controllers in the training arena. The second phase uses this archive to determine the controller action selection in the test or target arena, and the archive is enriched with the resulting trajectory. The FIFO update mechanism of the archive gradually removes the data related to the training arena, and replaces it with the data related to the target, currently visited, arena.

Four Ev-ITER modes are considered, governing the optimization objective in the first phase of Ev-ITER and summarized in Table 5.1. We considered the two optimization objectives already investigated in [Delarboulas *et al.* 2010], respectively referred to as Curiosity and Discovery (chapter 4), where the fitness associated to a controller respectively measures the entropy of the trajectory (Eq. 4.5) or the differential entropy of the trajectory (Eq. 4.6).
In [Delarboulas *et al.* 2010], these objectives measure the entropy or differential entropy of the sensori-motor data; we additionally consider here the entropy or differential entropy applied to the only actuator data, defined as the output of the neural net (and yielding the motor actions through rounding).

	Mode	Optimization objective in Phase 1	Applied on
1	Ev-ITER-C	Entropy (Curiosity)	sensori-motor data
2	Ev-ITER-D	Differential Entropy (Discovery)	
3	Ev-ITER-Ca	Entropy (Curiosity)	actuator data
4	Ev-ITER-Da	Differential Entropy (Discovery)	

Table 5.1: The four Ev-ITER modes.

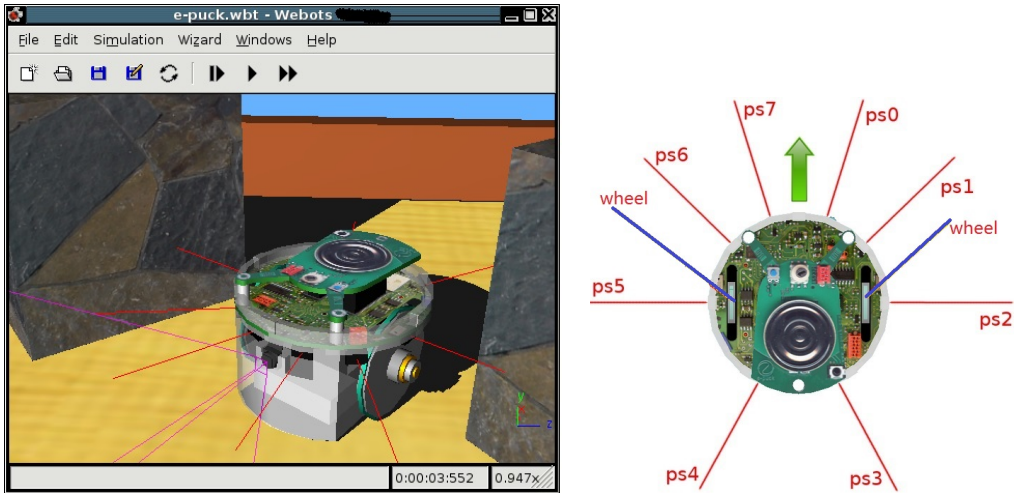
5.2 Experimental setting

This section firstly describes the robot agent (section 5.2.1) and the training and test environments (section 5.2.2). The baseline algorithms are then listed in section 5.2.3, and the performance indicators are reported in section 5.2.4.

5.2.1 The robot agent

All experiments consider a simulated robot, and are based on the Webots simulator emulating an E-puck robot[1] (Fig. 5.1(a)).

The E-puck diameter is 7.4 cm; its height is 4.5 cm. It is equipped with 2 motors and 8 infra-red proximity sensors, 6 in the front and 2 in the back. The placement of the sensors and wheels is illustrated on Fig. 5.1(b). Each proximity sensor returns a 12-bit resolution value (integer in $[0, 2^{12}]$), increasing with the proximity of an obstacle to the sensor. The movement of the E-puck robot is achieved by two stepper motors, respectively controlling the movement of the right and left wheels. The speed is given in a number of ticks/seconds where 1000 ticks correspond to a complete rotation of the wheel. The values are clamped between -1000 and 1000. For the sake of computational conveniency, these values are normalized in $[-1, 1]$.



(a) (b)

Figure 5.1: The robot agent. Left: Webots model of the e-Puck robot. Right: Top view of the E-puck robot. The red lines represent the directions of the infrared distance sensors, labelled with the distance sensor names in { ps0 ... ps7 }.

[1] http://www.cyberbotics.com/dvd/common/doc/webots/guide/section8.1.html.

5.2.2 The environments

Fig. 5.2 depicts the training environment, adapted from [Lehman & Stanley 2008, Delarboulas *et al.* 2010] and called *Hard arena* in the cited papers.

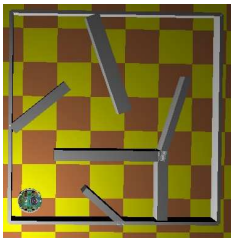


Figure 5.2: Training environment, adapted from [Lehman & Stanley 2008, Delarboulas *et al.* 2010]. The starting point is in the lower left corner.

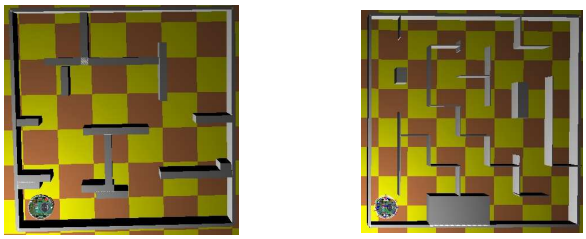
Two more complex arenas are considered (Fig. 5.3, left and right) in order to assess the generality property. These arenas, though similar in structure to the training arena, are designed to have many different corners instead of just straight walls, making the exploration task more difficult. Overall, the three arenas are ranked by increasing difficulty: i) the training arena is referred to as **easy arena**; ii) the **graph arena** (Fig. 5.3, left); iii) the **maze arena** (Fig. 5.3, right). The easy and graph arenas (respectively, the maze arena) are 0.6 m $\times$ 0.6 m (resp. 0.7 m $\times$ 0.7 m). For the sake of comparison, each arena is discretized in 100 $\times$ 100 squares.

To cross the arena in diagonal at full speed, assuming that there is no obstacles, the required number of time steps is 102 for the graph and easy arenas, and 122 for the maze arena.

In all experiments, the same starting point is considered, set to the lower left corner.

5.2.3 The baseline algorithms

The baseline algorithms are selected according to the original specifications, taken from the SYMBRION European project (European Integrated Project 216342, 2008-2013), requiring that the controller learning algorithm can run online, on-board, without requiring any ground truth or human intervention. These specifications, aimed at preventing the so-called reality gap (chapter 3), forbid for instance the use of the Novelty scheme [Lehman & Stanley 2008], which requires one to record all ending positions for all robot trajectories (as the robot fitness is set to its distance between its ending position and the previous ending positions). [Koutník *et al.* 2013, Koutník *et al.* 2014] also require the robot to know its actual location in order to



(a) graph arena (b) maze arena

Figure 5.3: Target arenas: Left: graph arena (0.6 m × 0.6 m); Right: maze arena (0.7 m × 0.7 m). The starting point is in the lower left corner.

compute the distance along the track axis measured from the starting line, and these approaches likewise do not satisfy the SYMBRION requirements. For the same reason, the approaches based on simultaneous localization and mapping (see e.g. [Williams & Browne 2012]) are not applicable either.

Finally, the baseline algorithms considered in the following include:

- The curiosity- and discovery driven approaches [Delarboulas *et al.* 2010], referred to in the following as evolutionary approaches;
- The intrinsic motivation [Lopes *et al.* 2012];
- A Brownian walk, implementing a most simple random walk [Fricke *et al.* 2013], where the action is uniformly selected in each time step in the action space.

5.2.4 The performance indicators

All reported results are averaged out of 15 independent runs. The indicators defined below are reported for a given duration of the 2nd phase, measured as number of epochs (where each epoch involves 2,000 time steps, section 5.2.5).

p_ℓ This indicator records the fraction of squares (out of the 10,000 squares of each arena) visited at least ℓ times during the 2nd phase, with $\ell = 1,\ 2,\ 5,\ 10$;

v_e This indicator indicates whether the controller does visit the farthest chambers from the starting point. This indicator is visually inspected by displaying the set of squares in each arena (colored in red when the number of visits in the 2nd phase is greater than 1 for each one out of 15 runs).

5.2.5 Algorithm parameters

All algorithm parameters are summarized in Table 5.2. The controller search space considered in the evolutionary approaches and in Ev-ITER is the space of multi-layer perceptrons with 8 inputs, 2 outputs, and 10 hidden neurons, amounting to a weight vector of dimension 112 (search space is $\mathbb{R}^{112}$). The 112 weights are initially randomly drawn following a normal distribution with mean 0 and variance 0.1.
For the Ev-ITER-1st phase and for the evolutionary approaches, an epoch or generation corresponds to launching a controller for 2,000 time steps (as said, it requires circa 100 time steps to cross the easy arena from the lower left corner to the upper high corner assuming no obstacles), starting from the lower left corner. The generated trajectory is assessed corresponding to the fitness mode (computing the entropy or the differential entropy, of the sensori-motor data or of the actuator data).

- The parameters involved in the 1st phase of Ev-ITER are same as in [Delarboulas *et al.* 2010], except for the ε parameter of ε-clustering as another simulator was considered.

- The evolutionary optimization is based on the (1+1)-Evolution Strategy using the $1/5^{th}$ rule (chapter 2), with isotropic Gaussian mutation $\mathcal{N}(0, \sigma I)$, with respectively $\vec{0}$ and I standing for the null vector in $\mathbb{R}^{112}$ and I standing for the identity matrix 112×112 and $\sigma = .2$;

- In each run, the evolution is reinitialized after 30 fitness evaluations with no improvement.

The Ev-ITER-1st phase and the pure evolutionary controllers only differ in the number of epochs, set to 2,000 for the evolutionary controllers, and to 200 for the Ev-ITER-1st phase. In Ev-ITER, the trajectories gathered up to the 200th epoch are recorded and used as initial information in the archive (tables $Z(i,j)$ and $S(i,j)$). The probability of uniform action selection (babbling mode) is set to $\eta = .05$. The uniform action selection is also triggered when the information on the current state is insufficient ($\sum_j |Z(i,j)| < \lambda' = 500$). Otherwise, the controller selects the action with highest weighted sum of i) entropy of the resulting state (weight $1-\alpha$), and ii) estimated probability of leading to another state (weight α), with parameter $\alpha = .7$.

The intrinsic motivation algorithm starts with an initially empty $Z(i,j)$ and $S(i,j)$, considering that $Q(i,j)$ is a proxy for the accuracy of the forward model in state i,j (Eq. 4.3).

Parameters	Symbol	**Value**
Number of time steps in each trajectory	T	2,000
Sensori-motor cluster radius	ε	1.8
Actuator cluster radius	ε'	0.6
Sensor cluster radius	ε''	1
Number of epochs in Evolutionary approaches		2,000
Number of epochs in Ev-ITER-1st phase		500
Buffering length of $Z(i,j)$	λ	60
Buffering length of $\sum_j \lvert Z(i,j)\rvert$	λ'	500
Weight of changing state term	α	0.7
Probability of random action selection	η	5%

Table 5.2: Parameters for experiments.

5.3 Experimental results

This section reports on the comparative validation of Ev-ITER. We first examine the results obtained by the Brownian move (section 5.3.1), before assessing the results of the evolutionary approaches, and examining how the actuator-based entropy or differential entropy behaves, compared to the original sensori-motor based entropy or differential entropy used in [Delarboulas *et al.* 2010].

5.3.1 The Brownian move baseline

As was expected, the Brownian move hardly visits the chambers which are the farthest away from the starting point (Fig 5.4): on the easy arena, on the graph arena and even more on the maze arena, it is visible that the robot is trapped in the first chambers; it never goes to further chambers due to the narrow width of the corridors and the number of angles.

5.3.2 Asessing the four evolutionary modes: (ensori-motor vs actuator -based, entropy vs differential entropy

This sub-section focuses on the relative performances of the four modes used in the evolutionary approaches (and in the 1st-phase of Ev-ITER), using same training and test arenas. For 2,000 epochs, where each epoch involves 2,000 time steps, controllers are evolved to maximize the entropy or differential entropy (respectively Curiosity vs Discovery in [Delarboulas *et al.* 2010]) measured from the sensori-motor vs actuator only data.

(a) Easy arena (b) Graph arena (c) Maze arena

Figure 5.4: The Brownian move: in red, locations visited after 2000 epochs of 2,000 time steps each for each run out of 15 independent runs.

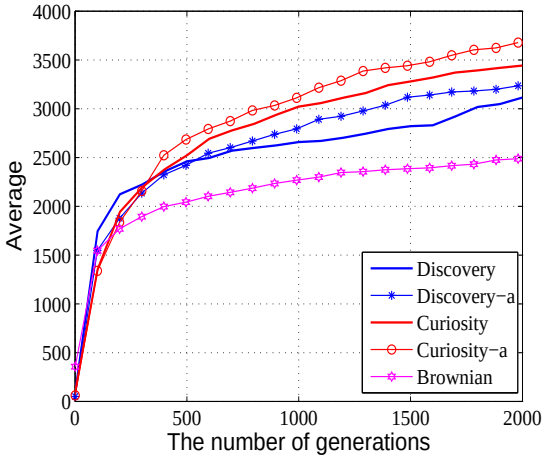
Fig. 5.5 reports the number of squares visited at least once over the three arenas, averaged out of 15 runs.

As said, the Brownian controller yields bad performances, reaching a plateau after the first 200 epochs. The differential entropy and entropy optimization objectives yield to significantly better results, specifically on the easy and maze arenas.

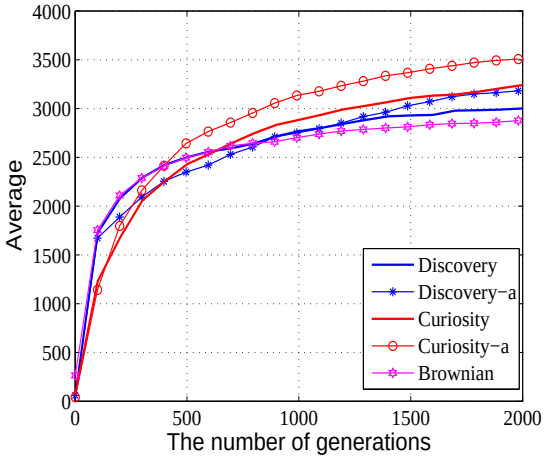
Interestingly, the results are significantly improved by considering actuator data instead of sensori-motor data, everything else being equal. A tentative interpretation for this result is the following. The actuator data is deterministically computed from the sensor data through the neural nets; therefore, the diversity of the sensor data is a necessary condition for the diversity of the actuator data. On the other hand, the diversity of the sensor data is not a sufficient condition for the diversity of the actuator data; for instance, if one sensor coordinate is not taken into account in the controller (e.g. the evolution compensating for sensor failures as in [Bongard *et al.* 2006]), its diversity makes no difference in the actual behavior of Maximizing the entropy of the actuator data might therefore contribute, more robustly than maximizing the entropy of the full sensori-motor data, to the behavioral diversity of the robot.

As in [Delarboulas *et al.* 2010], it is seen that the entropy optimization (curiosity) outperforms the differential entropy optimization (discovery), all the more so as more difficult arenas are considered. In Fig. 5.5, the difference increases between Curiosity and Curiosity-a, and from Discovery to Discovery-a from the top (easy arena) to the bottom (hard maze arena).

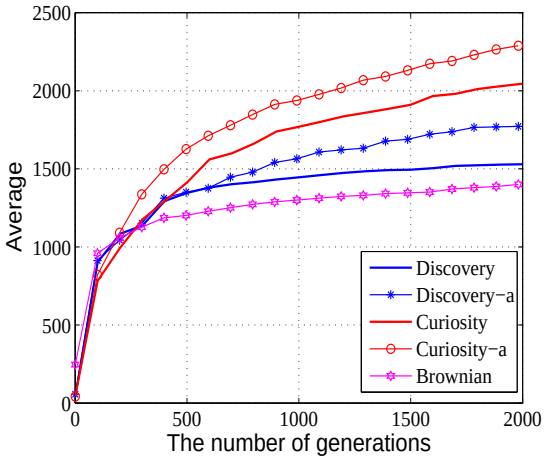
The difference is confirmed by the map of the visited squares (Fig. 5.6), showing that the differential entropy mode leads to less visiting the upper right chamber. Graph (Fig. 5.8: top, 3rd row) and maze (Fig. 5.10: top, 4th row) arenas show similar trends.



(a)



(b)



(c)

Figure 5.5: Comparative performances of the entropy and differential entropy, applied on sensori-motor or actuator data, after 2,000 epochs on the three arenas: (a) easy arena, (b) graph arena and (c) maze arena. The performance is the number of squares visited at least once, averaged out of 15 independent runs, comparatively to the Brownian controller.

Curiosity-a

Discovery-a

Curiosity

Discovery

Figure 5.6: Comparative performance of the optimization objectives, maximizing the entropy or the differential entropy (curiosity or discovery) of the sensori-motor or actuator data (-a) on the easy arena: squares visited 1 times or more after 2000 epochs over the 15 runs.

5.3.3 Comparative performances of Ev-ITER, sensori-motor modes

Let us present the experimental validation of Ev-ITER, where the 1st-Phase of Ev-ITER relies on optimizing the entropy or differential entropy of the sensori-motor data, comparatively to the baseline algorithms, Discovery, Curiosity and Intrinsic-Motivation.

In this section, the Ev-ITER-1st-phase is launched using the easy arena as training arena. The number of squares visited at least once during the Ev-ITER-2nd Phase in the target arenas, ranging over the easy arena (same as the training arena), the graph and the maze arenas, is visualized in Fig. 5.7 (a) easy; (b) graph; (c) maze. Likewise, the evolutionary approaches are launched on the easy arena; the generality of the evolved controllers is assessed by launching them in the target arenas.

A more detailed account of the performance indicators p_1 ($\ell = 1, 2, 5, 10$) is reported in Table 5.3, indicating the average and median number of squares visited 1, 2, 5 or 10 times, averaged on 2,000 epochs and 15 runs.

It is noted that both Ev-ITER-C and Ev-ITER-D outperform Intrinsic-Motivation, all the more so on the more complex arenas. Let us remind that Intrinsic-Motivation mostly differs from Ev-ITER as it starts with an empty archive. As the Ev-ITER archive includes (part of[2]) the trajectories of the 200 controllers generated during Ev-ITER-1stPhase, this difference could result in causing a simple delay of 200 generations in the performance curve of Intrinsic-Motivation. However, all performance curves rise abruptly for the easy and graph arenas, and to a lesser extent for the maze arena, and then reach a plateau. As seen from Fig. 5.7, The growth rate of Ev-ITER-C and Ev-ITER-D compared to Intrinsic-Motivation in the maze arena is higher than in the easy and graph arenas.

The fact that Ev-ITER starts with a diversified archive seems to prime a cumulative advantage phenomenon: it explores better the target arena, gathering more diverse observations, which in turn supports a better action selection.

A most surprising result is that Ev-ITER-C and Ev-ITER-D outperform Curiosity and Discovery even on the training, easy, arena, despite the fact that the optimization objective is meant to favor the exploration of the training arena. A tentative interpretation for this fact is twofold. On the one hand, entropy is but a proxy for the number of squares visited. On the other hand, the space to which belong the Ev-ITER controllers is much more complex than the neural net space. In particular, neural net controllers are bound to be continuous, and yield same actions

[2]Since only the last λ (s, a, s') events are retained in the archive, for s and a falling in a given sensor or motor cluster, (chapter 4, section 3).

in similar sensor contexts. Quite the contrary, the Ev-ITER controllers can select quite different actions for close sensor vectors, provided that these sensor vectors fall in different clusters.

A more expected result, Ev-ITER-C and Ev-ITER-D outperform their evolutionary counter-parts, Curiosity and Discovery, on other arenas than the training arena, all the more so as the arena is more complex: the gap between Ev-ITER and its evolutionary counterpart widens when passing from the graph to the maze arena.

Finally, the results show that entropy is slightly more efficient than differential entropy as optimization objective in what regards the exploratory performance: differential entropy slightly improves on the easy arena, but entropy catches up after 1,000 generations on the graph arena, and outperforms differential entropy from the start on the maze arena.

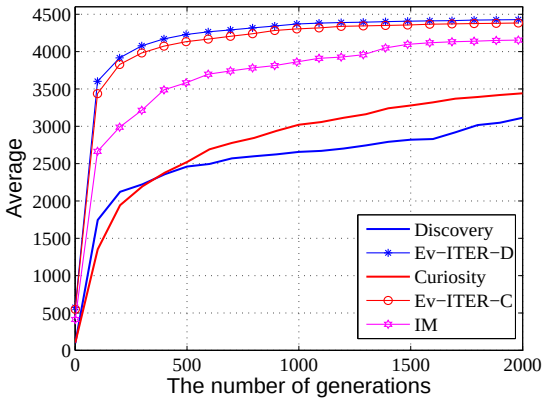
The good generality of Ev-ITER is visually assessed on Fig. 5.8, showing the squares actually visited at most once after 500 epochs (left column) and 2,000 epochs (2nd to rightmost column), on the easy arena (column 1 and 2), and the graph and maze arenas (respectively 3rd and 4th columns). While Curiosity outperforms Discovery, they are both lagging behind the other three approaches in all cases. On the easy and medium arenas, the performances of IM are visually a bit behind those of Ev-ITER-D and Ev-ITER-C for 500 epochs (complementary results omitted due to space restrictions), and they catch up for 2,000 epochs. On the maze arena finally, the performances of IM are behind those of Ev-ITER-D and Ev-ITER-C for both 500 and 2,000 epochs (see the middle corridors in the maze).

These results show the merits of the hybrid Ev-ITER approach in the considered settings. While, Ev-ITER-D and Ev-ITER-C significantly both improve on the evolutionary Discovery and Curiosity approaches, they also improve on the intrinsic motivation approach, as they are shown to explore more densely the regions far from the starting point.

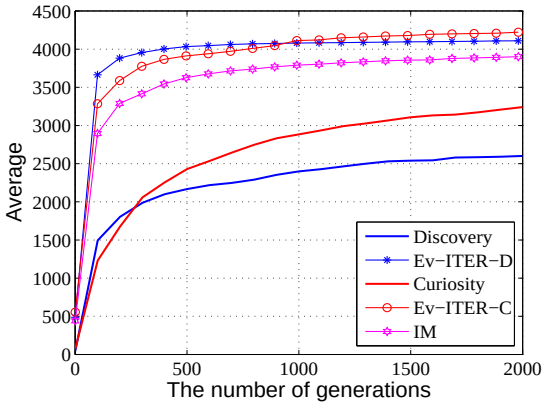
5.3.4 Actuator-entropy based validation of Ev-ITER

Let us finally present the experimental validation of Ev-ITER, where the 1st-Phase of Ev-ITER relies on optimizing the entropy or differential entropy of the actuator data, comparatively to the baseline algorithms, Discovery, Curiosity and Intrinsic-Motivation.

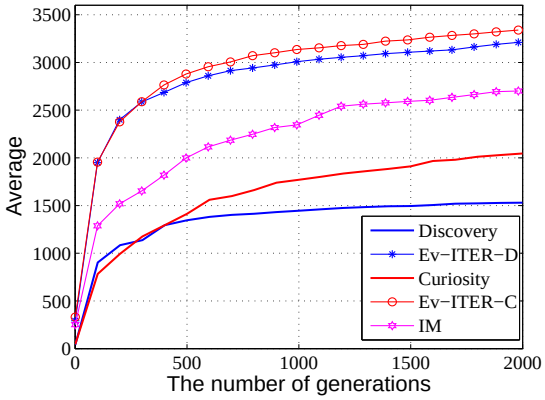
Likewise, the exploratory performance of all algorithms is comparatively displayed on the easy (Fig.5.9. a), graph (Fig.5.9. b) and maze (Fig.5.9. c) arenas, showing the number of squares visited at least once per run, and Table 5.4 reports a more detailed account of the performance indicators, indicating the average and median number of squares visited at least 1, 2, 5 or 10 times. These results confirm



(a)



(b)



(c)

Figure 5.7: Comparative performances of Discovery, Curiosity, IM, Ev-ITER-D and Ev-ITER-C in sensori-motor mode, on the easy arena, on the graph arena and on the maze arena. The performance is the number of squares visited at least once, averaged out of 15 independent runs. It is reminded that Curiosity and Discovery evolutionary approaches, as well as Ev-ITER-1st phase, are trained from the Easy arena.

Easy arena:500 **Easy arena:2000** **Graph arena:2000** **Maze arena:2000**

(a) Discovery (b) Discovery (c) Discovery (d) Discovery

(e) Curiosity (f) Curiosity (g) Curiosity (h) Curiosity

(i) IM (j) IM (k) IM (l) IM

(m) Ev-ITER-D (n) Ev-ITER-D (o) Ev-ITER-D (p) Ev-ITER-D

(q) Ev-ITER-C (r) Ev-ITER-C (s) Ev-ITER-C (t) Ev-ITER-C

Figure 5.8: Comparative performances of Discovery, Curiosity, IM, Ev-ITER-D and Ev-ITER-C in sensori-motor mode (from top to bottom row), on the easy arena (column 1, after 500 epochs; column 2 after 2,000 epochs), on the graph arena (column 3, after 2,000 epochs) and on the maze arena (column 4, after 2,000 epochs). The performance is the number of squares visited at least once, averaged out of 15 independent runs. Trajectories of Discovery (top row), Curiosity(2nd row), IM (3rd row), Ev-ITER-D (4th row) and Ev-ITER-C (bottom row) on the easy, graph and maze arenas, cumulative over 500 robots and 2,000 robots.

Algorithm	1 visit		2 visits		5 visits		10 visits	
	Medi	Aver(std.dev.)	Medi	Aver(std.dev.)	Medi	Aver(std.dev.)	Medi	Aver(std.dev.)
	Results on the easy arena							
Discovery	3252	3241 ± 643	2711	2848 ± 656	2270	2336 ± 592	2062	1968 ± 517
Curiosity	3489	3442 ± 667	3090	3084 ± 738	2420	2583 ± 786	2098	2212 ± 723
IM	4335	4156 ± 395	4270	3996 ± 599	3962	3651 ± 824	3512	3229 ± 868
Ev-ITER-D	4432	4427 ± 39	4395	4382 ± 59	4258	4260 ± 104	4095	4027 ± 273
Ev-ITER-C	4413	4384 ± 83	4374	4317 ± 143	4325	$4152 \pm 300.$	4212	3952 ± 464
	Results on the graph arena							
Discovery	2567	2601 ± 959	2125	2329 ± 609	1605	1951 ± 603	1331	1636 ± 916
Curiosity	3286	3241 ± 673	2901	2972 ± 764	2374	2588 ± 835	1926	2240 ± 848
IM	4022	3902 ± 283	3967	3806 ± 366	3901	3625 ± 502	3762	3414 ± 616
Ev-ITER-D	4116	4111 ± 25	4089	4081 ± 25	4038	4039 ± 33	4000	3979 ± 75
Ev-ITER-C	4383	4223 ± 450	4352	4100 ± 566	4284	3872 ± 737	4130	3651 ± 899
	Results on the maze arena							
Discovery	1217	1530 ± 640	1047	1344 ± 619	865	1136 ± 571	771	1001 ± 542
Curiosity	1998	2045 ± 568	1789	1789 ± 533	1493	1471 ± 409	1208	1260 ± 318
IM	2786	2706 ± 575	2599	2494 ± 610	2207	2165 ± 641	1845	1897 ± 626
Ev-ITER-D	3336	3212 ± 317	3123	3046 ± 399	2675	2778 ± 532	2274	2503 ± 622
Ev-ITER-C	3402	3341 ± 254	3225	3163 ± 305	2881	2844 ± 400	2528	2536 ± 480

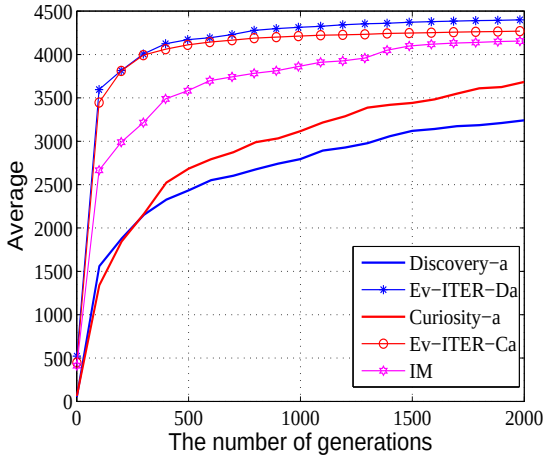
Table 5.3: Indicator p_ℓ in sensori-motor mode: number of square visited at least 1, 2, 5 and 10 times after 2,000 epochs in the easy, graph and maze arenas (median and average (std-deviation) out of 15 runs).

that Ev-ITER-Ca and Ev-ITER-Da improve on Intrinsic-Motivation, which itself outperform Curiosity-a and Discovery-a, on all arenas.

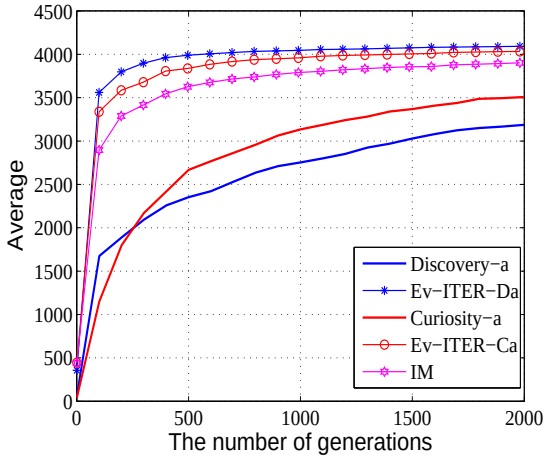
The generality property is also visually assessed on Fig. 5.10, showing the squares visited at least once after 500 epoch on the easy arena (left column) and 2,000 epochs (on the easy arena, 2nd column; graph arena, 3rd column; and maze arena, 4th column). It is seen that the sensori-motor and the actuator yield comparable results overall, with non-statistically significant differences.

5.3.5 Discussion and Perspectives

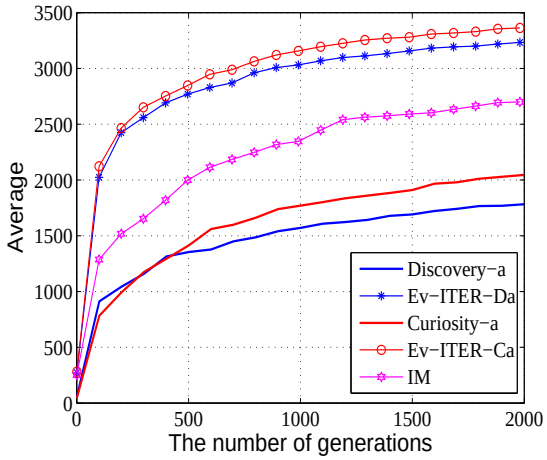
The goal of the presented approach, to provide a controller achieving good exploratory performances in an on-board, online fashion without requiring human intervention or ground truth, is successfully reached, with Ev-ITER matching the



(a)

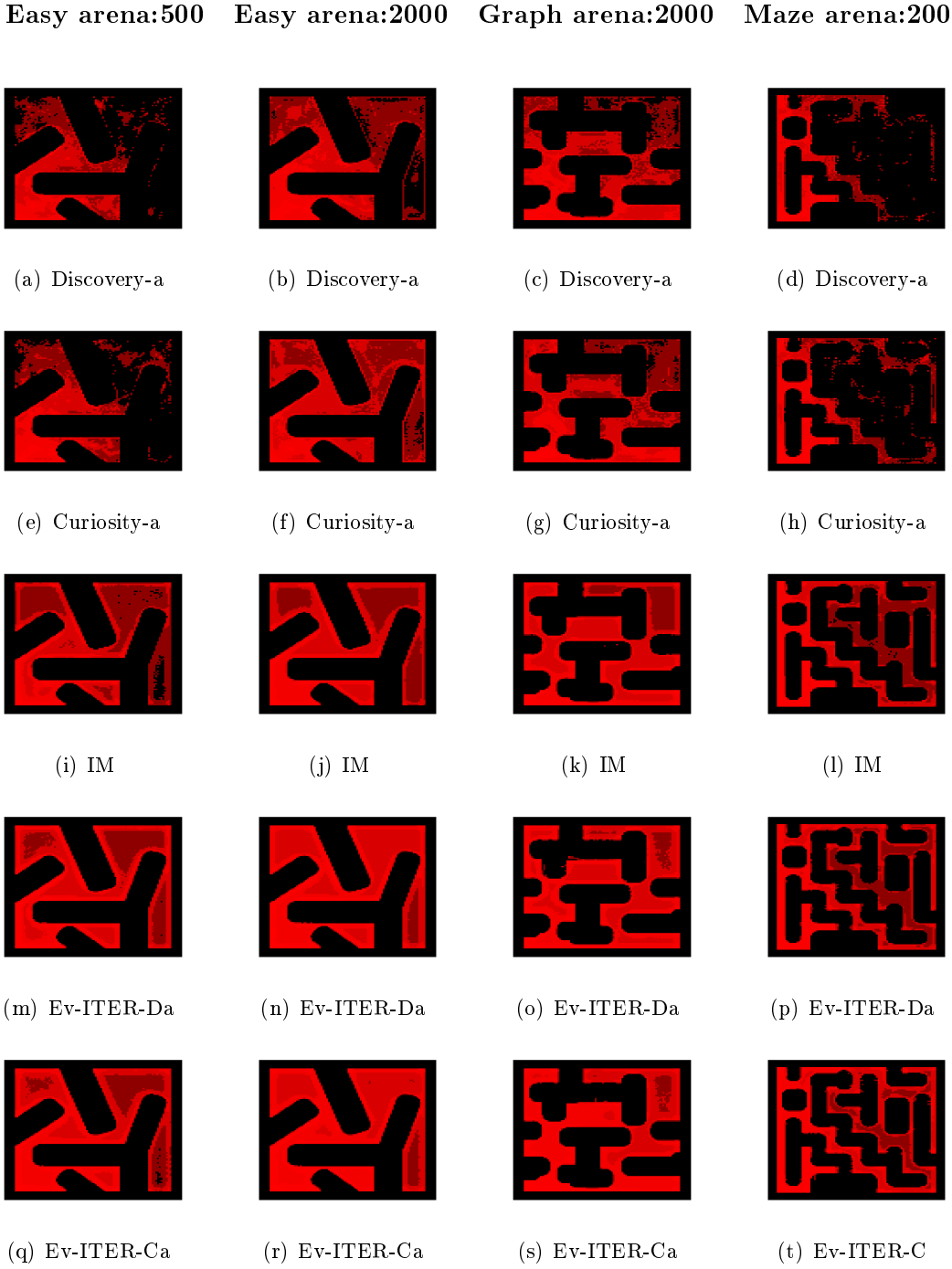


(b)



(c)

Figure 5.9: Comparative performances of Discovery, Curiosity, IM, Ev-ITER-D and Ev-ITER-C in actuator mode, under same conditions as in Fig. 5.7.



(a) Discovery-a (b) Discovery-a (c) Discovery-a (d) Discovery-a

(e) Curiosity-a (f) Curiosity-a (g) Curiosity-a (h) Curiosity-a

(i) IM (j) IM (k) IM (l) IM

(m) Ev-ITER-Da (n) Ev-ITER-Da (o) Ev-ITER-Da (p) Ev-ITER-Da

(q) Ev-ITER-Ca (r) Ev-ITER-Ca (s) Ev-ITER-Ca (t) Ev-ITER-C

Figure 5.10: Comparative performances of Discovery, Curiosity, IM, Ev-ITER-D and Ev-ITER-C in actuator mode, under same conditions as in Fig. 5.8.

Algorithm	1 visit		2 visits		5 visits		10 visits	
	Medi	Aver(std.dev.)	Medi	Aver(std.dev.)	Medi	Aver(std.dev.)	Medi	Aver(std.dev.)
	Results on the easy arena							
Discovery-a	3252	3241 ± 644	2711	2848 ± 656	2270	2336 ± 592	2062	1968 ± 517
Curiosity-a	3647	3683 ± 294	3198	3273 ± 383	2475	2621 ± 466	2036	2186 ± 407
IM	4335	4156 ± 396	4270	3996 ± 599	3962	3651 ± 824	3512	3229 ± 868
Ev-ITER-Da	4418	4401 ± 56	4381	4332 ± 116	4260	4148 ± 86)	4095	3938 ± 177
Ev-ITER-Ca	4397	4269 ± 160	4341	4204 ± 181	4171	4059 ± 299	3980	3862 ± 496
	Results on the graph arena							
Discovery-a	3454	3187 ± 623	3148	2915 ± 642	2617	2468 ± 631	2169	2070 ± 610
Curiosity-a	3646	3508 ± 418	3348	3260 ± 475	2866	2854 ± 521)	2441	2452 ± 526
IM	4022	3902 ± 284	3967	3806 ± 366	3901	3625 ± 503	3762	3414 ± 616
Ev-ITER-Da	4095	4093 ± 36	4068	4053 ± 59	4021	3967 ± 142	3968	3849 ± 239
Ev-ITER-Ca	4070	4035 ± 107	4036	3978 ± 151	3931	3862 ± 218	3726	3710 ± 294
	Results on the maze arena							
Discovery-a	1850	1783 ± 444	1610	1557 ± 377	1377	1310 ± 304	1225	1143 ± 265
Curiosity-a	1998	2045 ± 454	1789	1789 ± 433	1493	1471 ± 409	1228	1260 ± 318
IM	2786	2706 ± 575	2599	2494 ± 610	2207	2165 ± 642	1845	1897 ± 626
Ev-ITER-Da	3418	3236 ± 492	3245	3051 ± 583	2766	2706 ± 657	2406	2379 ± 669
Ev-ITER-Ca	3508	3365 ± 348	3403	3213 ± 406	3185	2961 ± 498	2875	2706 ± 555

Table 5.4: Indicator p_ℓ in actuator mode, under same conditions as in Table 5.3.

performance of the Intrinsic-Motivation and of the evolutionary robotic approaches complying with the same requirements.

It appears that Ev-ITER actually yields the best of both worlds. The data archive provided by the short preliminary evolutionary phase gives a significant advaantage to Ev-ITER compared to Intrinsic-Motivation. Furthermore, Intrinsic-Motivation does not catch up after gathering as much data as the one provided in the data repository, which suggests that the *quality of these data* provides a cumulative advantage to the exploration: the more data acquired, the better the exploration can be directed toward appropriate actions, thus priming a virtuous circle.

Compared to the evolutionary approaches on the other hand, Ev-ITER benefits from its action selection mechanism, allowing a much more flexible controller space than allowed by (low-dimensionality) neural nets.

The second main contribution of the proposed approach is to yield good performances even though the data repository is gathered on a different arena than the

arena actually explored. As discussed in section 5.1, this property of generality is original with respect to the state of the art, in reinforcement learning as well as in evolutionary robotics.

The approch suffers from several limitations:

- An important limitation of the approach lies in the clustering phase, which considers a fixed cluster radius ε. A main research perspective is to integrate more tightly the clustering phase within the evolutionary and exploratory processes, dynamically splitting the cells with highest transition entropy.
- A (modest) limitation of the approach is that a non-negligible fraction of time must be spent in acquiring the data archive (200 epochs, that is 1/10 of the 2,000 epochs involved in the evolutionary approaches). Complementary experiments showed that results were significantly degraded when reducing this time under 200 epochs.
- Overall, the key limitation of the presented results is that no experiments *in-situ* could be achieved. Porting these results on real-robots is the main priority of further experiments.

These results open several research perspectives, beyond the dynamic clustering aspects abovementioned.

- Firstly, it is yet unclear why and when the actuator data provides a better support than the sensori-motor data to the entropy optimization. An intermediate approach would be to consider the entropy of the data in the hidden layer of the neural nets, considering that the hidden layer provides an efficient representation of both the sensor data, and of the actuator data.
- Secondly, the limits of the generality property must be thoroughly assessed, considering more and more different arenas in order to understand when a target arena is sufficiently close from a training arena. Likewise, the generality property can be assessed by considering robots with (slightly) different sensori-motor equipments.

Chapter 6

Conclusion and Perspectives

As said, the Ev-ITER framework, which constitutes the main contribution of the presented PhD work, is at the cross-road of evolutionary robotics and reinforcement learning.

This combination of evolution and learning is original, to our best knowledge, in the following sense. In most hybrid approaches in the evolutionary and learning litterature, evolution is applied to the direct optimization of the solution, hypothesis or model, while learning is applied to focus and guide the evolutionary search, and/or to specialize or repair the evolutionary solution. In the proposed Ev-ITER scheme, evolution is merely applied to optimize the *data repository* provided to the learning algorithm, which will support good exploratory decisions; the learning algorithm autonomously proceeds on the basis of its data repository (though the strategy is mixed with a small probability of uniform exploration, to prevent the deterministic strategy from meeting endless loops), and maintains it using a simple FIFO mechanism.

In other words, the hybrid Ev-ITER framework suggests that an initial critical mass of information is required to feed reasoning in an appropriate way; and the empirical comparison with the Intrinsic Motivation framework [Lopes *et al.* 2012] (chapter 5) suggests that reasoning from scratch can hardly catch up. The exploratory controller, be it implemented through Ev-ITER or through Intrinsic Motivation, involves two ingredients: i) an action selection algorithm, based on its current information; and ii) this current information, compressing the past trajectories of the robot (and complying with its bounded memory resources by forgetting long past data) and defining a data repository. However, the result of the action selection algorithm (together with the robot environment) modifies the data repository itself.

The exploratory controller thereby defines a dynamic system, where the current information conditions the actions, which themselves modify the current information. The originality thus lies in considering both ingredients as a whole, using a rather simple action selection mechanism, and considering that this action selection mechanism *only requires to be seeded with appropriate information* to function appropriately, and to regenerate the data repository when the agent suddenly faces a new environment. The bulk of optimization thus focuses on the acquisition of

appropriate information, in the source environment. In other words, the Ev-ITER strategy can be viewed as yet another example of the Big Data motto: *Data beats algorithms*.

As said again (Chapter 5, last section), Ev-ITER fulfills some of the initial goals: it can (in principle) run on-board online, with bounded computational and memory resources, without requiring any ground truth or prior knowledge; the only ground truth is provided by the robot environment itself through the sensor information. The most appreciable empirical property is the generality property, as the Ev-ITER controller can apparently be transferred from one environment to another without compromising its exploratory efficiency[1], opening many interesting potential applications.

The main limitation of the present work is the lack of experimentations *in-situ*.

This work opens quite a few perspectives for further research.

- A first direction regards the automatic adjustment of the clusters along Ev-ITER-1st and 2nd phases, taking inspiration from [Lopes *et al.* 2012].
- A second direction is to extend the Ev-ITER mechanism to achieve other than exploratory behaviors. One possibility is to involve the user in the loop along an interactive optimization setting [Akrour *et al.* 2014].
- A related issue is how to organize the flow of information among the states. As noted by [Van Roy & Wen 2014], the main issue in reinforcement learning is that the exploration must be *planned* and cannot be achieved by greedy techniques: one must want to go in some states – although already well explored – because they might lead to other states which need additional exploration. In other words, some look-ahead is needed to achieve effective exploration. In the Ev-ITER setting, while Phase 2 implements a myopic and greedy exploration, it does so on data which have been gathered using a non-myopic criterion in Phase 1 (since Phase 1 aims at maximizing the global information gathered along a single trajectory). The fact that the data repository offers a global (approximate) perspective on the arena thus compensates to some extent the myopic strategy of the deterministic controller (together with the mixing with a η-uniform controller).
  The perspective of including some look-ahead in the score function (Eq. 4.9) thus seems a promising perspective of this work.

[1] Though complementary experiments, transporting the Ev-ITER controller from one environment to another along a regular or irregular schedule, are required to assess the limits of this generality property.

Bibliography

[Abbeel *et al.* 2007] Pieter Abbeel, Adam Coates, Morgan Quigley and Andrew Y Ng. *An application of reinforcement learning to aerobatic helicopter flight.* Advances in neural information processing systems, vol. 19, page 1, 2007. (Cited on page 22.)

[Acharyya 2008] Ranjan Acharyya. A new approach for blind source separation of convolutive sources: Wavelet based separation using shrinkage function. VDM, Verlag Dr. Müller, 2008. (Cited on page 27.)

[Akrour *et al.* 2011a] Riad Akrour, Marc Schoenauer and Michele Sebag. *Preference-based policy learning.* In Machine Learning and Knowledge Discovery in Databases, pages 12–27. Springer, 2011. (Cited on pages 36 and 37.)

[Akrour *et al.* 2011b] Riad Akrour, Marc Schoenauer and Michele Sebag. *Preference-based Reinforcement Learning.* In Choice Models and Preference Learning Workshop at NIPS, volume 11, 2011. (Cited on page 36.)

[Akrour *et al.* 2014] Riad Akrour, Marc Schoenauer, Michèle Sebag and Jean-Christophe Souplet. *Programming by feedback.* In International Conference on Machine Learning, 2014. (Cited on pages 2, 37 and 92.)

[Akrour 2014] Riad Akrour. *Robust Preference Learning-based Reinforcement Learning.* PhD thesis, Université Paris-Sud, 2014. (Cited on pages 3 and 33.)

[Arvin *et al.* 2014] Farshad Arvin, John Murray, Chun Zhang, Shigang Yue*et al.* *Colias: an autonomous micro robot for swarm robotic applications.* International Journal of Advanced Robotic Systems, vol. 11, no. 113, pages 1–10, 2014. (Cited on page 1.)

[Auer *et al.* 2002] Peter Auer, Nicolo Cesa-Bianchi and Paul Fischer. *Finite-time analysis of the multiarmed bandit problem.* Machine learning, vol. 47, no. 2-3, pages 235–256, 2002. (Cited on page 47.)

[Auer *et al.* 2011] Peter Auer, Shiau Hong Lim and Chris Watkins. *Models for Autonomously Motivated Exploration in Reinforcement Learning - (Extended Abstract).* In ALT'11, pages 14–17, 2011. (Cited on page 47.)

[Auger 2005] Anne Auger. *Convergence results for the (1, λ)-SA-ES using the theory of ϕ-irreducible Markov chains.* Theoretical Computer Science, vol. 334, no. 1, pages 35–69, 2005. (Cited on page 37.)

[Back *et al.* 2008] Thomas Back, M Emmerich and OM Shir. *Evolutionary algorithms for real world applications [Application Notes].* Computational Intelligence Magazine, IEEE, vol. 3, no. 1, pages 64–67, 2008. (Cited on page 7.)

[Baldassarre & Mirolli 2013] Gianluca Baldassarre and Marco Mirolli. Intrinsically motivated learning in natural and artificial systems. Springer, 2013. (Cited on pages 39, 41 and 43.)

[Bandaru *et al.* 2011] Sunith Bandaru, Cem Celal Tutum, Kalyanmoy Deb and Jesper Henri Hattel. *Higher-level innovization: A case study from friction stir welding process optimization.* In Evolutionary Computation (CEC), 2011 IEEE Congress on, pages 2782–2789. IEEE, 2011. (Cited on page 12.)

[Banzhaf *et al.* 1997] Wolfgang Banzhaf, Peter Nordin and Markus Olmer. *Generating adaptive behavior using function regression within genetic programming and a real robot.* In 2nd International Conference on Genetic Programming, Stanford, pages 14–23, 1997. (Cited on page 16.)

[Baranès & Oudeyer 2009] Adrien Baranès and P-Y Oudeyer. *R-IAC: Robust intrinsically motivated exploration and active learning.* Autonomous Mental Development, IEEE Transactions on, vol. 1, no. 3, pages 155–169, 2009. (Cited on pages xi, 3, 23, 25, 33, 39, 44, 45, 47, 53, 54, 55, 56 and 70.)

[Baranes & Oudeyer 2010] Adrien Baranes and P-Y Oudeyer. *Intrinsically motivated goal exploration for active motor learning in robots: A case study.* In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 1766–1773. IEEE, 2010. (Cited on pages 43 and 47.)

[Baranes & Oudeyer 2013] Adrien Baranes and Pierre-Yves Oudeyer. *Active learning of inverse models with intrinsically motivated goal exploration in robots.* Robotics and Autonomous Systems, vol. 61, no. 1, pages 49–73, 2013. (Cited on pages 40, 43 and 44.)

[Barto & Mahadevan 2003] Andrew G Barto and Sridhar Mahadevan. *Recent advances in hierarchical reinforcement learning.* Discrete Event Dynamic Systems, vol. 13, no. 4, pages 341–379, 2003. (Cited on page 30.)

[Barto *et al.* 2004] Andrew G Barto, Satinder Singh and Nuttapong Chentanez. *Intrinsically motivated learning of hierarchical collections of skills.* In Proc. 3rd Int. Conf. Development Learn, pages 112–119, 2004. (Cited on pages 40 and 42.)

[Bellman 1957] Richard Bellman. *Dynamic programming*, 1957. (Cited on page 30.)

[Berlyne 1960] Daniel E Berlyne. *Conflict, arousal, and curiosity.* 1960. (Cited on page 42.)

[Berlyne 1965] Daniel Ellis Berlyne. *Structure and direction in thinking.* 1965. (Cited on page 43.)

[Bertsekas & Tsitsiklis 1995] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming: an overview.* In Decision and Control, 1995., Proceedings of the 34th IEEE Conference on, volume 1, pages 560–564. IEEE, 1995. (Cited on page 28.)

[Bird *et al.* 2008] Jon Bird, Phil Husbands, Martin Perris, Bill Bigge and Paul Brown. *Implicit fitness functions for evolving a drawing robot.* In Applications of Evolutionary Computing, pages 473–478. Springer, 2008. (Cited on pages 17 and 18.)

[Bishop 1995] Christopher M Bishop. Neural networks for pattern recognition. Oxford university press, 1995. (Cited on page 26.)

[Bishop 2006] Christopher M Bishop. Pattern recognition and machine learning. springer, 2006. (Cited on page 27.)

[Blum 2007] Avrim Blum. *Machine learning theory.* Carnegie Melon Universit, School of Computer Science, 2007. (Cited on page 26.)

[Blynel 2000] Jesper Blynel. *Reinforcement Learning on Real Robots.* PhD thesis, Aarhus Universitet, Datalogisk Afdeling, 2000. (Cited on pages xi and 28.)

[Boeing *et al.* 2004] Adrian Boeing, Stephen Hanham and Thomas Braunl. *Evolving autonomous biped control from simulation to reality.* In Proceedings of the 2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand, pages 13–15, 2004. (Cited on pages 2 and 19.)

[Bolton & Hand 2002] Richard J Bolton and David J Hand. *Statistical fraud detection: A review.* Statistical science, pages 235–249, 2002. (Cited on page 26.)

[Bongard & Lipson 2004] J Bongard and Hod Lipson. *Once more unto the breach: Co-evolving a robot and its simulator.* In Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9), pages 57–62, 2004. (Cited on pages 21 and 32.)

[Bongard *et al.* 2006] Josh Bongard, Victor Zykov and Hod Lipson. *Resilient machines through continuous self-modeling.* Science, vol. 314, no. 5802, pages 1118–1121, 2006. (Cited on pages 13, 22 and 78.)

[Boser *et al.* 1992] Bernhard E Boser, Isabelle M Guyon and Vladimir N Vapnik. *A training algorithm for optimal margin classifiers.* In Proceedings of the fifth annual workshop on Computational learning theory, pages 144–152. ACM, 1992. (Cited on page 27.)

[Brambilla *et al.* 2013] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari and Marco Dorigo. *Swarm robotics: a review from the swarm engineering perspective.* Swarm Intelligence, vol. 7, no. 1, pages 1–41, 2013. (Cited on pages 1 and 54.)

[Bredeche & Montanier 2010] Nicolas Bredeche and Jean-Marc Montanier. *Environment-driven embodied evolution in a population of autonomous agents.* In Parallel Problem Solving from Nature, PPSN XI, pages 290–299. Springer, 2010. (Cited on page 13.)

[Bredeche *et al.* 2010] Nicolas Bredeche, Evert Haasdijk and AE Eiben. *On-line, on-board evolution of robot controllers.* In Artifical Evolution, pages 110–121. Springer, 2010. (Cited on page 12.)

[Bredeche *et al.* 2012] Nicolas Bredeche, Jean-Marc Montanier, Wenguo Liu and Alan FT Winfield. *Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents.* Mathematical and Computer Modelling of Dynamical Systems, vol. 18, no. 1, pages 101–129, 2012. (Cited on page 54.)

[Brooks 1995] Rodney A Brooks. *Intelligence without reason.* The artificial life route to artificial intelligence: Building embodied, situated agents, pages 25–81, 1995. (Cited on page 19.)

[Bruner 1991] Jerome Bruner. *The narrative construction of reality.* Critical inquiry, pages 1–21, 1991. (Cited on page 41.)

[Brys *et al.* 2014] Tim Brys, Anna Harutyunyan, Peter Vrancx, Matthew E Taylor, Daniel Kudenko and Ann Nowé. *Multi-objectivization of reinforcement learning problems by reward shaping.* In Neural Networks (IJCNN), 2014 International Joint Conference on, pages 2315–2322. IEEE, 2014. (Cited on page 33.)

[Busa-Fekete *et al.* 2013] Róbert Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng and Eyke Hüllermeier. *Preference-based evolutionary direct policy search.* In ICRA Workshop on Autonomous Learning, 2013. (Cited on pages 36 and 37.)

[Busoniu *et al.* 2010] Lucian Busoniu, Robert Babuska, Bart De Schutter and Damien Ernst. Reinforcement learning and dynamic programming using function approximators. CRC press, 2010. (Cited on page 30.)

[Capdepuy *et al.* 2007] Philippe Capdepuy, Daniel Polani and Chrystopher L Nehaniv. *Maximization of potential information flow as a universal utility for collective behaviour.* In Artificial Life, 2007. ALIFE'07. IEEE Symposium on, pages 207–213. IEEE, 2007. (Cited on page 41.)

[Celebi *et al.* 2013] M Emre Celebi, Hassan A Kingravi and Patricio A Vela. *A comparative study of efficient initialization methods for the k-means clustering algorithm.* Expert Systems with Applications, vol. 40, no. 1, pages 200–210, 2013. (Cited on page 27.)

[Chang *et al.* 2015] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III and John Langford. *Learning to search better than your teacher.* arXiv preprint arXiv:1502.02206, 2015. (Cited on page 30.)

[Chentanez *et al.* 2004] Nuttapong Chentanez, Andrew G Barto and Satinder P Singh. *Intrinsically motivated reinforcement learning.* In Advances in neural information processing systems, pages 1281–1288, 2004. (Cited on page 43.)

[Chernova & Veloso 2004] Sonia Chernova and Manuela Veloso. *An evolutionary approach to gait learning for four-legged robots.* In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 3, pages 2562–2567. IEEE, 2004. (Cited on page 17.)

[Cruz-Álvarez *et al.* 2013] Víctor Ricardo Cruz-Álvarez, Fernando Montes-Gonzalez, Efrén Mezura-Montes and José Santos. *Robotic behavior implementation using two different differential evolution variants.* In Advances in Artificial Intelligence, pages 216–226. Springer, 2013. (Cited on page 10.)

[Csikszentmihalyi & Csikzentmihaly 1991] Mihaly Csikszentmihalyi and Mihaly Csikzentmihaly. Flow: The psychology of optimal experience, volume 41. HarperPerennial New York, 1991. (Cited on page 37.)

[Csikszentmihalyi 1997] Mihaly Csikszentmihalyi. *Flow and the Psychology of Discovery and Invention.* HarperPerennial, New York, 1997. (Cited on pages 38 and 43.)

[Csikszentmihalyi 2000] Mihaly Csikszentmihalyi. Beyond boredom and anxiety. Jossey-Bass, 2000. (Cited on page 37.)

[Darwin 1859] Charles Darwin. *On the origins of species by means of natural selection.* London: Murray, 1859. (Cited on page 7.)

[Das & Suganthan 2011] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. *Differential evolution: A survey of the state-of-the-art.* Evolutionary Computation, IEEE Transactions on, vol. 15, no. 1, pages 4–31, 2011. (Cited on page 10.)

[Davis & Sampson 1986] John C Davis and Robert J Sampson. Statistics and data analysis in geology, volume 646. Wiley New York et al., 1986. (Cited on page 25.)

[Dayan & Balleine 2002] Peter Dayan and Bernard W Balleine. *Reward, motivation, and reinforcement learning.* Neuron, vol. 36, no. 2, pages 285–298, 2002. (Cited on page 39.)

[De Charms 1968] Richard De Charms. Personal causation: The internal affective determinants of behavior. Academic Press, New York, 1968. (Cited on pages 38 and 43.)

[De Croon *et al.* 2013] GCHE De Croon, LM O'connor, C Nicol and D Izzo. *Evolutionary robotics approach to odor source localization.* Neurocomputing, vol. 121, pages 481–497, 2013. (Cited on page 12.)

[de Margerie *et al.* 2007] E de Margerie, JB Mouret, S Doncieux and JA Meyer. *Artificial evolution of the morphology and kinematics in a flapping-wing mini-UAV.* Bioinspiration & biomimetics, vol. 2, no. 4, page 65, 2007. (Cited on page 21.)

[Deb & Srinivasan 2006] Kalyanmoy Deb and Aravind Srinivasan. *Innovization: Innovating design principles through optimization.* In Proceedings of the 8th annual conference on Genetic and evolutionary computation, pages 1629–1636. ACM, 2006. (Cited on page 12.)

[Deb & Srinivasan 2008] Kalyanmoy Deb and Aravind Srinivasan. *Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization.* In Multiobjective Problem Solving from Nature, pages 243–262. Springer, 2008. (Cited on pages 12 and 13.)

[Deb *et al.* 2001] Kalyanmoy Deb*et al.* Multi-objective optimization using evolutionary algorithms, volume 2012. John Wiley & Sons Chichester, 2001. (Cited on page 12.)

[Deci & Ryan 1985] Edward L. Deci and Richard M. Ryan. *Intrinsic Motivation and Self-Determination in Human Behavior*. In Plenum, New York, 1985. (Cited on page 38.)

[Delarboulas *et al.* 2010] Pierre Delarboulas, Marc Schoenauer and Michèle Sebag. *Open-ended evolutionary robotics: an information theoretic approach*. In Parallel Problem Solving from Nature, PPSN XI, pages 334–343. Springer, 2010. (Cited on pages xii, 2, 4, 23, 50, 53, 54, 56, 57, 60, 63, 66, 70, 71, 72, 74, 75, 76, 77 and 78.)

[Di Chiara 1999] Gaetano Di Chiara. *Drug addiction as dopamine-dependent associative learning disorder*. European journal of pharmacology, vol. 375, no. 1, pages 13–30, 1999. (Cited on page 39.)

[Di Paolo 2004] Ezequiel Di Paolo. *Crawling out of the simulation: Evolving real robot morphologies using cheap, reusable modules*. In Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Artificial Life, volume 9, page 94. MIT Press, 2004. (Cited on page 17.)

[Doncieux & Hamdaoui 2011] Stéphane Doncieux and Mohamed Hamdaoui. *Evolutionary Algorithms to Analyse and Design a Controller for a Flapping Wings Aircraft*. In New Horizons in Evolutionary Robotics, pages 67–83. Springer, 2011. (Cited on page 13.)

[Doncieux & Mouret 2010] Stephane Doncieux and J-B Mouret. *Behavioral diversity measures for Evolutionary Robotics*. In Evolutionary Computation (CEC), 2010 IEEE Congress on, pages 1–8. IEEE, 2010. (Cited on page 17.)

[Doncieux *et al.* 2011] Stéphane Doncieux, Nicolas Bredeche and Jean-Baptiste Mouret. New horizons in evolutionary robotics: Extended contributions from the 2009 evoderob workshop, volume 341. Springer, 2011. (Cited on page 12.)

[Duda *et al.* 2001] Richard O Duda, Peter E Hart and David G Stork. Pattern classification. John Wiley & Sons, 2001. (Cited on page 57.)

[Duda *et al.* 2012] Richard O Duda, Peter E Hart and David G Stork. Pattern classification. John Wiley & Sons, 2012. (Cited on pages 1 and 27.)

[Dunlop & Tamhane 2000] Dorothy D Dunlop and Ajit C Tamhane. Statistics and data analysis: from elementary to intermediate. Prentice Hall, 2000. (Cited on page 25.)

[Eiben & Smith 2003] Agoston E Eiben and James E Smith. Introduction to evolutionary computing. springer, 2003. (Cited on page 7.)

[Eiben *et al.* 2010a] AE Eiben, Evert Haasdijk, Nicolas Bredeche*et al.* *Embodied, on-line, on-board evolution for autonomous robotics.* Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution., vol. 7, pages 361–382, 2010. (Cited on page 22.)

[Eiben *et al.* 2010b] AE Eiben, Giorgos Karafotias and Evert Haasdijk. *Self-adaptive mutation in on-line, on-board evolutionary robotics.* In Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference on, pages 147–152. IEEE, 2010. (Cited on page 23.)

[Felch & Granger 2011] Andrew Felch and Richard Granger. *4 Sensor-rich robots driven by real-time brain circuit algorithms.* Neuromorphic and Brain-Based Robots, page 58, 2011. (Cited on page 31.)

[Festinger 1957] Leon Festinger. *A theory of cognitive dissonance.* 1957. (Cited on page 38.)

[Floreano & Mattiussi 2008] Dario Floreano and Claudio Mattiussi. Bio-inspired artificial intelligence: theories, methods, and technologies. MIT press, 2008. (Cited on page 12.)

[Floreano & Mondada 1998] Dario Floreano and Francesco Mondada. *Evolutionary neurocontrollers for autonomous mobile robots.* Neural networks, vol. 11, no. 7, pages 1461–1478, 1998. (Cited on pages 20 and 24.)

[Floreano & Urzelai 2001] Dario Floreano and Joseba Urzelai. *Evolution of plastic control networks.* Autonomous robots, vol. 11, no. 3, pages 311–317, 2001. (Cited on page 21.)

[Floreano *et al.* 1994] Dario Floreano, Francesco Mondada, Dario Floreano, Dario Floreano, Francesco Mondada and Francesco Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. ETH-Zürich, 1994. (Cited on page 18.)

[Fogel *et al.* 1966] L.J. Fogel, A.J. Owens and M.J. Walsh. *Artifical intelligence through simulated evolution.* 1966. (Cited on page 9.)

[Fogel 2006] David B Fogel. Evolutionary computation: toward a new philosophy of machine intelligence, volume 1. John Wiley & Sons, 2006. (Cited on page 7.)

[Fricke *et al.* 2013] G Matthew Fricke, François Asperti-Boursin, Joshua Hecker, Judy Cannon and Melanie Moses. *From microbiology to microcontrollers: Robot search patterns inspired by T cell movement.* In Advances in Artificial Life, ECAL, volume 12, pages 1009–1016, 2013. (Cited on page 75.)

[Friston *et al.* 2012] Karl Friston, Rick A Adams, Laurent Perrinet and Michael Breakspear. *Perceptions as hypotheses: saccades as experiments.* Frontiers in psychology, vol. 3, 2012. (Cited on page 39.)

[Fukuda *et al.* 1999] Toshio Fukuda, Kenichiro Mase and Yasuhisa Hasegawa. *Robot hand manipulation by evolutionary programming.* In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, volume 3, pages 2458–2463. IEEE, 1999. (Cited on page 9.)

[Fürnkranz & Hüllermeier 2010] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning. Springer, 2010. (Cited on page 36.)

[Fürnkranz *et al.* 2012] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng and Sang-Hyeun Park. *Preference-based reinforcement learning: a formal framework and a policy iteration algorithm.* Machine Learning, vol. 89, no. 1-2, pages 123–156, 2012. (Cited on page 36.)

[Geist *et al.* 2013] Matthieu Geist, Edouard Klein, Bilal Piot, Yann Guermeur, Olivier Pietquin*et al. Around inverse reinforcement learning and score-based classification.* In Reinforcement Learning and Decision Making Meetings, 2013. (Cited on page 36.)

[Goldberg & Holland 1988] David E Goldberg and John H Holland. *Genetic algorithms and machine learning.* Machine learning, vol. 3, no. 2, pages 95–99, 1988. (Cited on page 9.)

[Gomi & Ide 1998] Takashi Gomi and Koichi Ide. *Evolution of gaits of a legged robot.* In Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, volume 1, pages 159–164. IEEE, 1998. (Cited on page 16.)

[Gottlieb *et al.* 2013] Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes and Adrien Baranes. *Information-seeking, curiosity, and attention: computational and neural mechanisms.* Trends in cognitive sciences, vol. 17, no. 11, pages 585–593, 2013. (Cited on page 39.)

[Gregor *et al.* 2012] Michal Gregor, Juraj Spalek and Jan Capak. *Use of context blocks in genetic programming for evolution of robot morphology.* In ELEKTRO, 2012, pages 286–291. IEEE, 2012. (Cited on page 9.)

[Hansen & Ostermeier 1996] Nikolaus Hansen and Andreas Ostermeier. *Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation.* In Evolutionary Computation, 1996., Proceedings of IEEE International Conference on, pages 312–317. IEEE, 1996. (Cited on pages 11 and 37.)

[Hansen & Ostermeier 2001] Nikolaus Hansen and Andreas Ostermeier. *Completely derandomized self-adaptation in evolution strategies.* Evolutionary computation, vol. 9, no. 2, pages 159–195, 2001. (Cited on pages 11 and 37.)

[Harlow 1950] Harry F Harlow. *Learning and satiation of response in intrinsically motivated complex puzzle performance by monkeys.* Journal of Comparative and Physiological Psychology, vol. 43, no. 4, page 289, 1950. (Cited on page 38.)

[Hart & Grupen 2011] Stephen Hart and Roderic Grupen. *Learning generalizable control programs.* Autonomous Mental Development, IEEE Transactions on, vol. 3, no. 3, pages 216–231, 2011. (Cited on page 30.)

[Hartland & Bredeche 2006] Cédric Hartland and Nicolas Bredeche. *Evolutionary robotics, anticipation and the reality gap.* In Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on, pages 1640–1645. IEEE, 2006. (Cited on page 21.)

[Hartland 2009] Cédric Hartland. *A contribution to robust adaptive robotic control acquisition.* PhD thesis, Paris 11, 2009. (Cited on pages xi, 14 and 15.)

[Harvey *et al.* 1992] Inman Harvey, Philip Husbands, Dave Cliff*et al.* Issues in evolutionary robotics. School of Cognitive and Computing Sciences, University of Sussex, 1992. (Cited on page 19.)

[Hastie *et al.* 2009] Trevor Hastie, Robert Tibshirani and Jerome Friedman. Unsupervised learning. Springer, 2009. (Cited on page 27.)

[Hauert *et al.* 2009] Sabine Hauert, J-C Zufferey and Dario Floreano. *Reverse-engineering of artificially evolved controllers for swarms of robots.* In Evolutionary Computation, 2009. CEC'09. IEEE Congress on, pages 55–61. IEEE, 2009. (Cited on page 12.)

[Haugeland 1985] John Haugeland. *Artificial Intelligence: The Very Idea.* 1985. (Cited on page 26.)

[Heidrich-Meisner & Igel 2009] Verena Heidrich-Meisner and Christian Igel. *Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search.* In Proceedings of the 26th Annual International Conference on Machine Learning, pages 401–408. ACM, 2009. (Cited on pages 37 and 50.)

[Hemker *et al.* 2006] Thomas Hemker, Hajime Sakamoto, Maximilian Stelzer and Oskar von Stryk. *Hardware-in-the-loop optimization of the walking speed of a humanoid robot.* In Proc. of CLAWAR, 2006. (Cited on pages 20 and 22.)

[Hoffmann & Pfister 1996] Frank Hoffmann and Gerd Pfister. *Evolutionary learning of a fuzzy control rule base for an autonomous vehicle.* In Proceedings of the Fifth International Conference IPMU: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, pages 659–664, 1996. (Cited on page 16.)

[Hornby *et al.* 2000a] Gregory S Hornby, Seiichi Takamura, Osamu Hanagata, Masahiro Fujita and J Pollack. *Evolution of controllers from a high-level simulator to a high DOF robot.* In Evolvable Systems: from biology to hardware, pages 80–89. Springer, 2000. (Cited on pages 3, 17 and 23.)

[Hornby *et al.* 2000b] Gregory S Hornby, Seiichi Takamura, Jun Yokono, Osamu Hanagata, Takashi Yamamoto and Masahiro Fujita. *Evolving robust gaits with AIBO.* In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, volume 3, pages 3040–3045. IEEE, 2000. (Cited on page 18.)

[Horvitz 2000] JC Horvitz. *Mesolimbocortical and nigrostriatal dopamine responses to salient non-reward events.* Neuroscience, vol. 96, no. 4, pages 651–656, 2000. (Cited on page 39.)

[Huang & Weng 2002] Xiao Huang and John Weng. *Novelty and reinforcement learning in the value system of developmental robots.* 2002. (Cited on page 42.)

[Huang & Weng 2004] Xiao Huang and Juyang Weng. *Motivational system for human-robot interaction.* In Computer Vision in Human-Computer Interaction, pages 17–27. Springer, 2004. (Cited on page 42.)

[Hull 1943] Clark Leonard Hull. *Principles of behavior: An introduction to behavior theory.* 1943. (Cited on page 38.)

[Hunt 1965] John M Hunt. *Intrinsic motivation and its role in psychological development.* vol. 13, pages 189–282, 1965. (Cited on page 42.)

[Hurst *et al.* 2002] Jacob Hurst, Larry Bull and Chris Melhuish. *TCS learning classifier system controller on a real robot.* pages 588–597, 2002. (Cited on pages 53 and 69.)

[Hutter *et al.* 2015] F Hutter, B Kégl, R Caruana, I Guyon, H Larochelle and E Viegas. *Automatic Machine Learning (AutoML).* In ICML 2015 Workshop on Resource-Efficient Machine Learning, 32nd International Conference on Machine Learning, 2015. (Cited on page 62.)

[Igel *et al.* 2007] Christian Igel, Nikolaus Hansen and Stefan Roth. *Covariance matrix adaptation for multi-objective optimization.* Evolutionary computation, vol. 15, no. 1, pages 1–28, 2007. (Cited on page 11.)

[Jain 2010] Anil K Jain. *Data clustering: 50 years beyond K-means.* Pattern Recognition Letters, vol. 31, no. 8, pages 651–666, 2010. (Cited on page 27.)

[Jakobi *et al.* 1995] Nick Jakobi, Phil Husbands and Inman Harvey. *Noise and the reality gap: The use of simulation in evolutionary robotics.* In Advances in artificial life, pages 704–720. Springer, 1995. (Cited on pages 2, 18, 19 and 32.)

[Jakobi 1997] Nick Jakobi. *Evolutionary robotics and the radical envelope-of-noise hypothesis.* Adaptive behavior, vol. 6, no. 2, pages 325–368, 1997. (Cited on page 21.)

[Jakobi 1998] Nick Jakobi. *Running across the reality gap: Octopod locomotion evolved in a minimal simulation.* In Evolutionary Robotics, pages 39–58. Springer, 1998. (Cited on pages 16 and 21.)

[Kagan 1972] Jerome Kagan. *Motives and development.* Journal of personality and social psychology, vol. 22, no. 1, page 51, 1972. (Cited on page 38.)

[Kakade & Dayan 2002] Sham Kakade and Peter Dayan. *Dopamine: generalization and bonuses.* Neural Networks, vol. 15, no. 4, pages 549–559, 2002. (Cited on page 39.)

[Kala 2012] Rahul Kala. *Multi-robot path planning using co-evolutionary genetic programming.* Expert Systems with Applications, vol. 39, no. 3, pages 3817–3831, 2012. (Cited on page 9.)

[Kaplan & Oudeyer 2003] Frederic Kaplan and Pierre-Yves Oudeyer. *Motivational principles for visual know-how development.* 2003. (Cited on page 44.)

[Kaplan & Oudeyer 2007] Frederic Kaplan and Pierre-Yves Oudeyer. *In search of the neural circuits of intrinsic motivation.* Frontiers in neuroscience, vol. 1, no. 1, page 225, 2007. (Cited on page 38.)

[Karafotias *et al.* 2011] Giorgos Karafotias, Evert Haasdijk and Agoston Endre Eiben. *An algorithm for distributed on-line, on-board evolutionary robotics.* In Proceedings of the 13th annual conference on Genetic and evolutionary computation, pages 171–178. ACM, 2011. (Cited on page 22.)

[Kawai *et al.* 2001] Katsumi Kawai, Akio Ishiguro and Peter Eggenberger. *Incremental evolution of neurocontrollers with a diffusion-reaction mechanism of neuromodulators.* In IROS, pages 2384–2391, 2001. (Cited on page 17.)

[Kim *et al.* 2001] Yong-Jae Kim, Jong-Hwan Kim and Dong-Soo Kwon. *Evolutionary programming-based univector field navigation method for past mobile robots.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 31, no. 3, pages 450–458, 2001. (Cited on page 9.)

[Kim *et al.* 2015] Hyeoneun Kim, Woosang Lim, Kanghoon Lee, Yung-Kyun Noh and Kee-Eung Kim. *Reward Shaping for Model-Based Bayesian Reinforcement Learning.* In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015. (Cited on page 33.)

[Kleinberg 2003] Jon Kleinberg. *An impossibility theorem for clustering.* Advances in neural information processing systems, pages 463–470, 2003. (Cited on page 57.)

[Kober & Peters 2011] Jens Kober and Jan Peters. *Policy search for motor primitives in robotics.* Machine Learning, vol. 84, no. 1-2, pages 171–203, 2011. (Cited on page 32.)

[Kober & Peters 2012] Jens Kober and Jan Peters. *Reinforcement learning in robotics: A survey.* In Reinforcement Learning, pages 579–610. Springer, 2012. (Cited on pages xi, 3, 26, 30 and 31.)

[Kober *et al.* 2013] Jens Kober, J Andrew Bagnell and Jan Peters. *Reinforcement learning in robotics: A survey.* The International Journal of Robotics Research, vol. 32, no. 11, pages 1238–1274, 2013. (Cited on pages 27, 49 and 53.)

[Köker 2013] Raşit Köker. *A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization.* Information Sciences, vol. 222, pages 528–543, 2013. (Cited on page 9.)

[Konidaris *et al.* 2012] George Konidaris, Ilya Scheidwasser and Andrew G Barto. *Transfer in reinforcement learning via shared features.* The Journal of Machine Learning Research, vol. 13, no. 1, pages 1333–1371, 2012. (Cited on pages 55 and 72.)

[Koos *et al.* 2009] Sylvain Koos, J Mouret and Stéphane Doncieux. *Automatic system identification based on coevolution of models and tests.* In Evolutionary Computation, 2009. CEC'09. IEEE Congress on, pages 560–567. IEEE, 2009. (Cited on page 22.)

[Koos *et al.* 2013] Sylvain Koos, J-B Mouret and Stéphane Doncieux. *The transferability approach: Crossing the reality gap in evolutionary robotics.* Evolutionary Computation, IEEE Transactions on, vol. 17, no. 1, pages 122–145, 2013. (Cited on pages 2, 18, 19, 20 and 50.)

[Kormushev *et al.* 2013] Petar Kormushev, Sylvain Calinon and Darwin G Caldwell. *Reinforcement learning in robotics: Applications and real-world challenges.* Robotics, vol. 2, no. 3, pages 122–148, 2013. (Cited on pages 3, 27 and 30.)

[Koutník *et al.* 2013] Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber and Faustino Gomez. *Evolving large-scale neural networks for vision-based reinforcement learning.* In Proceedings of the 15th annual conference on Genetic and evolutionary computation, pages 1061–1068. ACM, 2013. (Cited on pages 4, 50, 53, 69 and 74.)

[Koutník *et al.* 2014] Jan Koutník, Jürgen Schmidhuber and Faustino Gomez. *Evolving deep unsupervised convolutional networks for vision-based reinforcement learning.* In Proceedings of the 2014 conference on Genetic and evolutionary computation, pages 541–548. ACM, 2014. (Cited on pages 4 and 74.)

[Koza 1992] John R Koza. Genetic programming: on the programming of computers by means of natural selection, volume 1. MIT press, 1992. (Cited on page 9.)

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems, pages 1097–1105, 2012. (Cited on page 27.)

[Kwok & Sheng 1994] DP Kwok and Fang Sheng. *Genetic algorithm and simulated annealing for optimal robot arm PID control.* In Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, pages 707–713. IEEE, 1994. (Cited on page 12.)

[Lagoudakis & Parr 2003a] Michail Lagoudakis and Ronald Parr. *Reinforcement learning as classification: Leveraging modern classifiers.* In ICML, volume 3, pages 424–431, 2003. (Cited on page 30.)

[Lagoudakis & Parr 2003b] Michail G Lagoudakis and Ronald Parr. *Approximate policy iteration using large-margin classifiers.* In Proceedings of the 18th international joint conference on Artificial intelligence, pages 1432–1434. Morgan Kaufmann Publishers Inc., 2003. (Cited on page 30.)

[Laskov *et al.* 2004] Pavel Laskov, Christin Schäfer, Igor Kotenko and K-R Müller. *Intrusion detection in unlabeled data with quarter-sphere support vector machines.* Praxis der Informationsverarbeitung und Kommunikation, vol. 27, no. 4, pages 228–236, 2004. (Cited on page 26.)

[LeCun *et al.* 1989] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard and Lawrence D Jackel. *Backpropagation applied to handwritten zip code recognition.* Neural computation, vol. 1, no. 4, pages 541–551, 1989. (Cited on page 27.)

[LeCun *et al.* 2004] Yann LeCun, Fu Jie Huang and Leon Bottou. *Learning methods for generic object recognition with invariance to pose and lighting.* In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–97. IEEE, 2004. (Cited on page 26.)

[Lehman & Stanley 2008] Joel Lehman and Kenneth O Stanley. *Exploiting Open-Endedness to Solve Problems Through the Search for Novelty.* In ALIFE, pages 329–336, 2008. (Cited on pages xii, 4, 15, 23, 48, 50, 55 and 74.)

[Lehman & Stanley 2011] Joel Lehman and Kenneth O Stanley. *Abandoning objectives: Evolution through the search for novelty alone.* Evolutionary computation, vol. 19, no. 2, pages 189–223, 2011. (Cited on pages 15 and 49.)

[Lehman *et al.* 2012] Joel Lehman, Sebastian Risi, David B D'Ambrosio and Kenneth O Stanley. *Rewarding Reactivity to Evolve Robust Controllers without Multiple Trials or Noise.* In Artificial Life, volume 13, pages 379–386, 2012. (Cited on page 50.)

[Lenaertz 2012] Mikael Lenaertz. Formal verification of flexibility inswarm robotics. Master's thesis, Université Libre de Bruxelles, Université D'Europe, 2012. (Cited on pages xi and 2.)

[Lim & Auer 2012] Shiau Hong Lim and Peter Auer. *Autonomous Exploration For Navigating In MDPs.* Journal of Machine Learning Research - Proceedings Track, pages 40–24, 2012. (Cited on page 47.)

[Lipson & Pollack 2000] Hod Lipson and Jordan B Pollack. *Automatic design and manufacture of robotic lifeforms.* Nature, vol. 406, no. 6799, pages 974–978, 2000. (Cited on pages 13 and 54.)

[Lipson *et al.* 2006] Hod Lipson, Josh C Bongard, Viktor Zykov and Evan Malone. *Evolutionary Robotics for Legged Machines: From Simulation to Physical Reality.* In IAS, pages 11–18, 2006. (Cited on pages 2, 3, 18 and 32.)

[Lisman & Grace 2005] John E Lisman and Anthony A Grace. *The hippocampal-VTA loop: controlling the entry of information into long-term memory.* Neuron, vol. 46, no. 5, pages 703–713, 2005. (Cited on page 39.)

[Lisowski *et al.* 2011] Michal Lisowski, Zhun Fan and Ole Ravn. *Differential evolution to enhance localization of mobile robots.* In Fuzzy Systems (FUZZ), 2011 IEEE International Conference on, pages 241–247. IEEE, 2011. (Cited on page 10.)

[Lopes *et al.* 2012] Manuel Lopes, Tobias Lang, Marc Toussaint and Pierre-Yves Oudeyer. *Exploration in model-based reinforcement learning by empirically estimating learning progress.* In Advances in Neural Information Processing Systems, pages 206–214, 2012. (Cited on pages 2, 44, 56, 71, 75, 91 and 92.)

[Manning & Schütze 1999] Christopher D Manning and Hinrich Schütze. Foundations of statistical natural language processing. MIT press, 1999. (Cited on page 26.)

[Marocco & Floreano 2002] Davide Marocco and Dario Floreano. *Active vision and feature selection in evolutionary behavioral systems.* From Animals to Animats, vol. 7, pages 247–255, 2002. (Cited on pages 2 and 23.)

[Martínez-Soto *et al.* 2014] Ricardo Martínez-Soto, Oscar Castillo and Juan R Castro. *Genetic Algorithm Optimization for Type-2 Non-singleton Fuzzy Logic Controllers.* In Recent Advances on Hybrid Approaches for Designing Intelligent Systems, pages 3–18. Springer, 2014. (Cited on page 9.)

[Matarić & Cliff 1996] Maja Matarić and Dave Cliff. *Challenges in evolving controllers for physical robots*. Robotics and autonomous systems, vol. 19, no. 1, pages 67–83, 1996. (Cited on page 18.)

[Matt Quinn *et al.* 2003] Lincoln Smith Matt Quinn, Giles Mayley and Phil Husbands. *Evolving teamwork and role-allocation with real robots*. Artificial Life 8, vol. 8, page 302, 2003. (Cited on page 17.)

[Michalewicz 1996] Zbigniew Michalewicz. Genetic algorithms+ data structures= evolution programs. springer, 1996. (Cited on page 11.)

[Michalski *et al.* 2013] Ryszard S Michalski, Jaime G Carbonell and Tom M Mitchell. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013. (Cited on page 27.)

[Michini & How 2012] Bernard Michini and Jonathan P How. *Bayesian nonparametric inverse reinforcement learning*. In Machine Learning and Knowledge Discovery in Databases, pages 148–163. Springer, 2012. (Cited on page 36.)

[Miglino *et al.* 1995] Orazio Miglino, Henrik Hautop Lund and Stefano Nolfi. *Evolving mobile robots in simulated and real environments*. Artificial life, vol. 2, no. 4, pages 417–434, 1995. (Cited on page 21.)

[Mitchell & Michell 1997] Tom M Mitchell and Thomas Michell. *Machine Learning (Mcgraw-Hill Series in Computer Science)*, 1997. (Cited on page 26.)

[Modayil *et al.* 2014] Joseph Modayil, Adam White and Richard S Sutton. *Multi-timescale nexting in a reinforcement learning robot*. Adaptive Behavior, vol. 22, no. 2, pages 146–160, 2014. (Cited on page 26.)

[Montanier & Bredeche 2011a] Jean-Marc Montanier and Nicolas Bredeche. *Embedded evolutionary robotics: The (1+ 1)-restart-online adaptation algorithm*. In New Horizons in Evolutionary Robotics, pages 155–169. Springer, 2011. (Cited on page 54.)

[Montanier & Bredeche 2011b] Jean-Marc Montanier and Nicolas Bredeche. *Emergence of altruism in open-ended evolution in a population of autonomous agents*. In Proceedings of the 13th annual conference companion on Genetic and evolutionary computation, pages 25–26. ACM, 2011. (Cited on page 60.)

[Montanier 2013] Jean-Marc Montanier. *Environment-driven Distributed Evolutionary Adaptation for Collective Robotic Systems*. PhD thesis, Université Paris Sud-Paris XI, 2013. (Cited on pages 14 and 17.)

[Montgomery 1954] Kay C Montgomery. *The role of the exploratory drive in learning.* Journal of Comparative and Physiological Psychology, vol. 47, no. 1, page 60, 1954. (Cited on page 38.)

[Moulin-Frier *et al.* 2013] Clément Moulin-Frier, Sao M Nguyen and Pierre-Yves Oudeyer. *Self-organization of early vocal development in infants and machines: the role of intrinsic motivation.* Frontiers in psychology, vol. 4, 2013. (Cited on page 40.)

[Mouret & Doncieux 2012] B Mouret J and Stéphane Doncieux. *Encouraging behavioral diversity in evolutionary robotics: An empirical study.* Evolutionary computation, vol. 20, no. 1, pages 91–133, 2012. (Cited on pages 50, 54 and 55.)

[Muelling *et al.* 2014] K Muelling, A Boularias, B Mohler, B Schölkopf and J Peters. *Learning strategies in table tennis using inverse reinforcement learning.* Biological cybernetics, pages 1–17, 2014. (Cited on page 34.)

[Murata & Onoda 2002] Hiroshi Murata and Takashi Onoda. *Estimation of power consumption for household electric appliances.* In Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on, volume 5, pages 2299–2303. IEEE, 2002. (Cited on page 26.)

[Nakamura *et al.* 2000] Hiroshi Nakamura, Akio Ishiguro and Y Uchilkawa. *Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks.* In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, volume 1, pages 158–165. IEEE, 2000. (Cited on page 17.)

[Nakanishi *et al.* 2008] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters and Stefan Schaal. *Operational space control: A theoretical and empirical comparison.* The International Journal of Robotics Research, vol. 27, no. 6, pages 737–757, 2008. (Cited on page 30.)

[Nelson *et al.* 2009] Andrew L Nelson, Gregory J Barlow and Lefteris Doitsidis. *Fitness functions in evolutionary robotics: A survey and analysis.* Robotics and Autonomous Systems, vol. 57, no. 4, pages 345–370, 2009. (Cited on pages xv, 14, 15 and 16.)

[Ng *et al.* 1999] Andrew Y Ng, Daishi Harada and Stuart Russell. *Policy invariance under reward transformations: Theory and application to reward shaping.* In ICML, volume 99, pages 278–287, 1999. (Cited on pages 33 and 35.)

[Ng *et al.* 2000] Andrew Y Ng, Stuart J Russell*et al.* *Algorithms for inverse reinforcement learning.* In Icml, pages 663–670, 2000. (Cited on pages 3, 33, 34 and 35.)

[Nodelman *et al.* 2012] Uri Nodelman, Christian R Shelton and Daphne Koller. *Expectation maximization and complex duration distributions for continuous time Bayesian networks.* arXiv preprint arXiv:1207.1402, 2012. (Cited on page 27.)

[Nolfi & Floreano 2000] Stefano Nolfi and Dario Floreano. Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. MIT press, 2000. (Cited on pages 1 and 2.)

[Nolfi & Floreano 2001] Stefano Nolfi and Dario Floreano. *Evolutionary robotics. The biology, intelligence, and technology of self-organizing machines.* Rapport technique, MIT press, 2001. (Cited on page 14.)

[Nolfi *et al.* 1994] Stefano Nolfi, Dario Floreano, Orazio Miglino and Francesco Mondada. *How to evolve autonomous robots: Different approaches in evolutionary robotics.* In Artificial life IV: Proceedings of the 4th International Workshop on Artificial Life, numéro LIS-CONF-1994-002, pages 190–197. MA: MIT Press, 1994. (Cited on pages 13 and 20.)

[Nordin & Banzhaf 1997] Peter Nordin and Wolfgang Banzhaf. *An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming.* Adaptive Behavior, vol. 5, no. 2, pages 107–140, 1997. (Cited on pages 2 and 23.)

[O'Dowd *et al.* 2011] PJ O'Dowd, AFT Winfield and M Studley. *The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours.* In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pages 4995–5000. IEEE, 2011. (Cited on page 54.)

[Ong *et al.* 2010] Yew-Soon Ong, Meng Hiot Lim and Xianshun Chen. *Memetic Computation–Past, Present & Future [Research Frontier].* Computational Intelligence Magazine, IEEE, vol. 5, no. 2, pages 24–31, 2010. (Cited on page 8.)

[Oudeyer & Kaplan 2007] Pierre-Yves Oudeyer and Frederic Kaplan. *What is intrinsic motivation? a typology of computational approaches.* Frontiers in neurorobotics, vol. 1, 2007. (Cited on pages 40, 41, 42, 43 and 44.)

[Oudeyer & Kaplan 2008] Pierre-Yves Oudeyer and Frederic Kaplan. *How can we define intrinsic motivation?* In proceedings of the 8th international conference on epigenetic robotics: modeling cognitive development in robotic systems, 2008. (Cited on pages 40 and 41.)

[Oudeyer *et al.* 2005] Pierre-Yves Oudeyer, Frédéric Kaplan, Verena V Hafner and Andrew Whyte. *The playground experiment: Task-independent development of a curious robot.* In Proceedings of the AAAI Spring Symposium on Developmental Robotics, pages 42–47. Stanford, California, 2005. (Cited on page 44.)

[Oudeyer *et al.* 2007] P-Y Oudeyer, Frédéric Kaplan and Verena Vanessa Hafner. *Intrinsic motivation systems for autonomous mental development.* Evolutionary Computation, IEEE Transactions on, vol. 11, no. 2, pages 265–286, 2007. (Cited on pages 23, 25, 39, 44, 54, 55 and 56.)

[Oudeyer *et al.* 2012] Pierre-Yves Oudeyer*et al. Interactive learning gives the tempo to an intrinsically motivated robot learner.* In IEEE-RAS International Conference on Humanoid Robots, 2012. (Cited on pages 3, 23, 25, 33, 53, 55, 56 and 70.)

[Oudeyer *et al.* 2013] Pierre-Yves Oudeyer, Adrien Baranes and Frédéric Kaplan. *Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints.* In Intrinsically motivated learning in natural and artificial systems, pages 303–365. Springer, 2013. (Cited on pages 39, 47 and 54.)

[Palmer *et al.* 2009] M Palmer, D Miller and T Blackwell. *An evolved neural controller for bipedal walking: Transitioning from simulator to hardware.* In Proc. of IROS 2009 Workshop on Exploring new horizons in Evolutionary Design of Robots, 2009. (Cited on page 20.)

[Panksepp 1998] Jaak Panksepp. Affective neuroscience: The foundations of human and animal emotions. Oxford university press, 1998. (Cited on page 39.)

[Parra *et al.* 2014] José Parra, Leonardo Trujillo and Patricia Melin. *Hybrid backpropagation training with evolutionary strategies.* Soft Computing, vol. 18, no. 8, pages 1603–1614, 2014. (Cited on pages 53 and 69.)

[Paugam-Moisy *et al.* 2006] Hélene Paugam-Moisy, Régis Martinez and Samy Bengio. *A supervised learning approach based on STDP and polychronization in spiking neuron networks.* Rapport technique, IDIAP, 2006. (Cited on page 27.)

[Pessin *et al.* 2010] Gustavo Pessin, FS Osório, Alberto Y Hata and Denis F Wolf. *Intelligent control and evolutionary strategies applied to multirobotic systems.* In Industrial Technology (ICIT), 2010 IEEE International Conference on, pages 1427–1432. IEEE, 2010. (Cited on page 12.)

[Pfeifer & Gomez 2005] Rolf Pfeifer and Gabriel Gomez. *Interacting with the real world: design principles for intelligent systems.* Artificial life and Robotics, vol. 9, no. 1, pages 1–6, 2005. (Cited on pages 1 and 25.)

[Pilarski *et al.* 2012] Patrick M Pilarski, Michael R Dawson, Thomas Degris, Jason P Carey and Richard S Sutton. *Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots.* In Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS & EMBS International Conference on, pages 296–302. IEEE, 2012. (Cited on page 26.)

[Pollack *et al.* 2000] Jordan B Pollack, Hod Lipson, Sevan Ficici, Pablo Funes, Greg Hornby and Richard A Watson. *Evolutionary techniques in physical robotics.* In Evolvable Systems: from biology to hardware, pages 175–186. Springer, 2000. (Cited on page 20.)

[Powell 2012] Warren B Powell. *AI, OR and control theory: a Rosetta Stone for stochastic optimization.* Princeton University, 2012. (Cited on page 30.)

[Prashanth & Andresen 2001] Sankaran Prashanth and Daniel Andresen. *Using implicit fitness functions for genetic algorithm-based agent scheduling.* In Proceedings of the 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001), Las Vegas. Citeseer, 2001. (Cited on page 18.)

[Price *et al.* 2006] Kenneth Price, Rainer M Storn and Jouni A Lampinen. Differential evolution: a practical approach to global optimization. Springer, 2006. (Cited on page 9.)

[Prieto *et al.* 2010] Abraham Prieto, Francisco Bellas, Jose A Becerra, Becerra Priego and Richard J Duro. *Self-organizing robot teams using asynchronous situated co-evolution.* In From Animals to Animats 11, pages 565–574. Springer, 2010. (Cited on page 18.)

[Puterman 2009] Martin L Puterman. Markov decision processes: discrete stochastic dynamic programming, volume 414. John Wiley & Sons, 2009. (Cited on page 28.)

[Quinlan 1993] John Ross Quinlan. C4. 5: programs for machine learning, volume 1. Morgan kaufmann, 1993. (Cited on page 27.)

[Ratliff *et al.* 2006] Nathan D Ratliff, J Andrew Bagnell and Martin A Zinkevich. *Maximum margin planning.* In Proceedings of the 23rd international conference on Machine learning, pages 729–736. ACM, 2006. (Cited on page 35.)

[Ratliff *et al.* 2007] Nathan Ratliff, David Bradley, J Andrew Bagnell and Joel Chestnutt. *Boosting structured prediction for imitation learning.* Robotics Institute, page 54, 2007. (Cited on page 33.)

[Rechenberg 1973] I Rechenberg. *Evolutionsstrategie–Optimierung technisher Systeme nach Prinzipien der biologischen Evolution.* 1973. (Cited on pages 7 and 10.)

[Rechenberg 1978] Ingo Rechenberg. Evolutionsstrategien. Springer, 1978. (Cited on page 10.)

[Redgrave & Gurney 2006] Peter Redgrave and Kevin Gurney. *The short-latency dopamine signal: a role in discovering novel actions?* Nature reviews neuroscience, vol. 7, no. 12, pages 967–975, 2006. (Cited on page 39.)

[Rice 2006] John Rice. Mathematical statistics and data analysis. Cengage Learning, 2006. (Cited on page 25.)

[Richiardi *et al.* 2013] Jonas Richiardi, Sophie Achard, Horst Bunke and Dimitri Van De Ville. *Machine learning with brain graphs: predictive modeling approaches for functional imaging in systems neuroscience.* Signal Processing Magazine, IEEE, vol. 30, no. 3, pages 58–70, 2013. (Cited on page 26.)

[Roy & McCallum 2001] Nicholas Roy and Andrew McCallum. *Toward optimal active learning through monte carlo estimation of error reduction.* ICML, Williamstown, 2001. (Cited on page 41.)

[Russell & Norvig 2010] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* 2010. (Cited on page 27.)

[Ryan & Deci 2000] Richard M Ryan and Edward L Deci. *Intrinsic and extrinsic motivations: Classic definitions and new directions.* Contemporary educational psychology, vol. 25, no. 1, pages 54–67, 2000. (Cited on pages 38, 39, 41 and 43.)

[Santucci *et al.* 2013] Vieri G Santucci, Gianluca Baldassarre and Marco Mirolli. *Which is the best intrinsic motivation signal for learning multiple skills?* Frontiers in neurorobotics, vol. 7, 2013. (Cited on page 43.)

[Saunders *et al.* 2011] Frank Saunders, Ethan Golden, Robert D White and Jason Rife. *Experimental verification of soft-robot gaits evolved using a lumped dynamic model.* Robotica, vol. 29, no. 06, pages 823–830, 2011. (Cited on page 20.)

[Saxena *et al.* 2008] Ashutosh Saxena, Justin Driemeyer and Andrew Y Ng. *Robotic grasping of novel objects using vision.* The International Journal of Robotics Research, vol. 27, no. 2, pages 157–173, 2008. (Cited on page 54.)

[Saxena *et al.* 2009] Ashutosh Saxena, Justin Driemeyer and Andrew Y Ng. *Learning 3-d object orientation from images.* In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 794–800. IEEE, 2009. (Cited on page 26.)

[Schaal & Atkeson 1994] Stefan Schaal and Christopher G Atkeson. *Robot juggling: implementation of memory-based learning.* Control Systems, IEEE, vol. 14, no. 1, pages 57–71, 1994. (Cited on page 47.)

[Schembri *et al.* 2007] Massimiliano Schembri, Marco Mirolli and Gianluca Baldassarre. *Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot.* In Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on, pages 282–287. IEEE, 2007. (Cited on page 43.)

[Schlesinger 2013] Matthew Schlesinger. *Investigating the origins of intrinsic motivation in human infants.* In Intrinsically motivated learning in natural and artificial systems, pages 367–392. Springer, 2013. (Cited on page 39.)

[Schmidhuber 1991] Jürgen Schmidhuber. *Curious model-building control systems.* In Proc. International Joint Conference on Neural Networks, Singapore, volume 2, pages 1458–1463. IEEE, 1991. (Cited on pages 25, 39, 43, 54 and 55.)

[Schmidhuber 2010] Jürgen Schmidhuber. *Formal theory of creativity, fun, and intrinsic motivation (1990–2010).* Autonomous Mental Development, IEEE Transactions on, vol. 2, no. 3, pages 230–247, 2010. (Cited on page 41.)

[Schölkopf & Smola 2002] Bernhard Schölkopf and Alexander J Smola. Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press, 2002. (Cited on page 27.)

[Schultz *et al.* 1996] Alan C Schultz, John J Grefenstette and William Adams. *Roboshepherd: Learning a complex behavior*. Robotics and Manufacturing: Recent Trends in Research and Applications, vol. 6, pages 763–768, 1996. (Cited on page 12.)

[Schultz 1998] Wolfram Schultz. *Predictive reward signal of dopamine neurons*. Journal of neurophysiology, vol. 80, no. 1, pages 1–27, 1998. (Cited on page 39.)

[Seo *et al.* 2010] Kisung Seo, Soohwan Hyun and Erik D Goodman. *Genetic programming-based automatic gait generation in joint space for a quadruped robot*. Advanced Robotics, vol. 24, no. 15, pages 2199–2214, 2010. (Cited on page 9.)

[Silverman 1986] Bernard W Silverman. Density estimation for statistics and data analysis, volume 26. CRC press, 1986. (Cited on page 25.)

[Sims 1994] Karl Sims. *Evolving virtual creatures*. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pages 15–22. ACM, 1994. (Cited on page 13.)

[Spears *et al.* 1993] William M Spears, Kenneth A De Jong, Thomas Bäck, David B Fogel and Hugo De Garis. *An overview of evolutionary computation*. In Machine Learning: ECML-93, pages 442–459. Springer, 1993. (Cited on page 10.)

[Sporns & Lungarella 2006] Olaf Sporns and Max Lungarella. *Evolving coordinated behavior by maximizing information structure*. In Artificial life X: proceedings of the tenth international conference on the simulation and synthesis of living systems, pages 323–329, 2006. (Cited on page 44.)

[Storn & Price 1997] Rainer Storn and Kenneth Price. *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces*. Journal of global optimization, vol. 11, no. 4, pages 341–359, 1997. (Cited on page 9.)

[Sutton & Barto 1998] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, volume 1. Cambridge Univ Press, 1998. (Cited on pages 1, 3, 27, 33, 40 and 71.)

[Sutton 1988] Richard S Sutton. *Learning to predict by the methods of temporal differences*. Machine learning, vol. 3, no. 1, pages 9–44, 1988. (Cited on pages 29 and 30.)

[Syed *et al.* 2008] Umar Syed, Michael Bowling and Robert E Schapire. *Apprenticeship learning using linear programming.* In Proceedings of the 25th international conference on Machine learning, pages 1032–1039. ACM, 2008. (Cited on page 35.)

[Taylor & Stone 2009] Matthew E Taylor and Peter Stone. *Transfer learning for reinforcement learning domains: A survey.* The Journal of Machine Learning Research, vol. 10, pages 1633–1685, 2009. (Cited on page 55.)

[Thompson *et al.* 1999] Adrian Thompson, Paul Layzell and Ricardo Zebulum. *Explorations in design space: Unconventional electronics design through artificial evolution.* Evolutionary Computation, IEEE Transactions on, vol. 3, no. 3, pages 167–196, 1999. (Cited on page 21.)

[Van der Spek 2014] IT Van der Spek. *Imitation learning for a robotic precision placement task.* PhD thesis, TU Delft, Delft University of Technology, 2014. (Cited on page 35.)

[Van Roy & Wen 2014] Benjamin Van Roy and Zheng Wen. *Generalization and exploration via randomized value functions.* arXiv preprint arXiv:1402.0635, 2014. (Cited on page 92.)

[Vapnik & Vapnik 1998] Vladimir Naumovich Vapnik and Vlamimir Vapnik. Statistical learning theory, volume 1. Wiley New York, 1998. (Cited on page 26.)

[Vapnik 1999] Vladimir N Vapnik. *An overview of statistical learning theory.* Neural Networks, IEEE Transactions on, vol. 10, no. 5, pages 988–999, 1999. (Cited on page 26.)

[Villarreal-Cervantes *et al.* 2010] Miguel G Villarreal-Cervantes, Carlos A Cruz-Villar, Jaime Alvarez-Gallegos and Edgar A Portilla-Flores. *Differential evolution techniques for the structure-control design of a five-bar parallel robot.* Engineering Optimization, vol. 42, no. 6, pages 535–565, 2010. (Cited on page 10.)

[Wang *et al.* 2015] W Wang, NH Reyes, ALC Barczak, T Susnjak and Peter Sincak. *Multi-Behaviour Robot Control using Genetic Network Programming with Fuzzy Reinforcement Learning.* In Robot Intelligence Technology and Applications 3, pages 151–158. Springer, 2015. (Cited on pages 53 and 69.)

[Warmuth *et al.* 2003] Manfred K Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta and Christian Lemmen. *Active learning with support vector machines in the drug discovery process.* Journal of Chemical

Information and Computer Sciences, vol. 43, no. 2, pages 667–673, 2003. (Cited on page 26.)

[Watson *et al.* 1999] Richard A Watson, SG Ficiei and Jordan B Pollack. *Embodied evolution: Embodying an evolutionary algorithm in a population of robots.* In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 1. IEEE, 1999. (Cited on pages 13 and 23.)

[Werbos 1987] Paul J Werbos. *Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research.* Systems, Man and Cybernetics, IEEE Transactions on, vol. 17, no. 1, pages 7–20, 1987. (Cited on page 28.)

[White 1959] Robert W White. *Motivation reconsidered: the concept of competence.* Psychological review, vol. 66, no. 5, page 297, 1959. (Cited on pages 38 and 43.)

[Williams & Browne 2012] Henry Williams and Will N Browne. *Integration of Learning Classifier Systems with simultaneous localisation and mapping for autonomous robotics.* pages 1–8, 2012. (Cited on pages 4, 53, 69, 70 and 75.)

[Wilson *et al.* 2012] Aaron Wilson, Alan Fern and Prasad Tadepalli. *A bayesian approach for policy learning from trajectory preference queries.* In Advances in Neural Information Processing Systems, pages 1133–1141, 2012. (Cited on page 36.)

[Wirth & Fürnkranz 2013a] Christian Wirth and Johannes Fürnkranz. *EPMC: Every Visit Preference Monte Carlo for Reinforcement Learning.* In Asian Conference on Machine Learning, pages 483–497, 2013. (Cited on page 36.)

[Wirth & Fürnkranz 2013b] Christian Wirth and Johannes Fürnkranz. *A policy iteration algorithm for learning from preference-based feedback.* In Advances in Intelligent Data Analysis XII, pages 427–437. Springer, 2013. (Cited on page 36.)

[Wirth & Fürnkranz 2013c] Christian Wirth and Johannes Fürnkranz. *Preference-Based Reinforcement Learning: A preliminary survey.* In Proceedings of the ECML/PKDD-13 Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards, 2013. (Cited on pages 3 and 33.)

[Wolff *et al.* 2008] Krister Wolff, David Sandberg and Mattias Wahde. *Evolutionary optimization of a bipedal gait in a physical robot.* In Evolutionary Computa-

tion, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, pages 440–445. IEEE, 2008. (Cited on page 18.)

[Yildirim *et al.* 2014] Sinan Yildirim, A Taylan Cemgil and Sumeetpal S Singh. *An Online Expectation-Maximisation Algorithm for Nonnegative Matrix Factorisation Models.* arXiv preprint arXiv:1401.2490, 2014. (Cited on page 27.)

[Zagal & Ruiz-Del-Solar 2007] Juan Cristóbal Zagal and Javier Ruiz-Del-Solar. *Combining simulation and reality in evolutionary robotics.* Journal of Intelligent and Robotic Systems, vol. 50, no. 1, pages 19–39, 2007. (Cited on page 22.)

[Zhan & Taylor 2015] Yusen Zhan and Matthew E Taylor. *Online Transfer Learning in Reinforcement Learning Domains.* arXiv preprint arXiv:1507.00436, 2015. (Cited on page 72.)

[Zhang & Sebag 2014] Guohua Zhang and Michèle Sebag. *Coupling Evolution and Information Theory for Autonomous Robotic Exploration.* In Parallel Problem Solving from Nature–PPSN XIII, pages 852–861. Springer, 2014. (Cited on page 53.)

[Zhang *et al.* 2011] Jun Zhang, Zhi-hui Zhan, Ying Lin, Ni Chen, Yue-jiao Gong, Jing-hui Zhong, Henry SH Chung, Yun Li and Yu-hui Shi. *Evolutionary computation meets machine learning: A survey.* Computational Intelligence Magazine, IEEE, vol. 6, no. 4, pages 68–75, 2011. (Cited on page 7.)

[Zhifei & Joo 2012] Shao Zhifei and Er Meng Joo. *A survey of inverse reinforcement learning techniques.* International Journal of Intelligent Computing and Cybernetics, vol. 5, no. 3, pages 293–311, 2012. (Cited on pages 34 and 36.)

[Zhou *et al.* 1996] Kemin Zhou, John Comstock Doyle, Keith Glover*et al.* Robust and optimal control, volume 40. Prentice hall New Jersey, 1996. (Cited on page 1.)

[Zhou *et al.* 2011] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan and Qingfu Zhang. *Multiobjective evolutionary algorithms: A survey of the state of the art.* Swarm and Evolutionary Computation, vol. 1, no. 1, pages 32–49, 2011. (Cited on page 12.)

[Ziebart *et al.* 2008] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell and Anind K Dey. *Maximum Entropy Inverse Reinforcement Learning.* In AAAI, pages 1433–1438, 2008. (Cited on page 36.)

[Zufferey *et al.* 2002] Jean-Christophe Zufferey, Dario Floreano, Matthijs Van Leeuwen and Tancredi Merenda. *Evolving vision-based flying robots.* In Biologically Motivated Computer Vision, pages 592–600. Springer, 2002. (Cited on page 17.)

[Zykov *et al.* 2004] Viktor Zykov, Josh Bongard and Hod Lipson. *Evolving dynamic gaits on a physical robot.* In Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO, volume 4, 2004. (Cited on pages 17 and 20.)