Knowledge Extraction from Learning Traces in Continuous Domains

Stephane Doncieux[†,‡]
[†]Sorbonne Universités, UPMC Univ Paris 06, UMR 722, ISIR, F-75005, Paris, France
[‡]CNRS, UMR 7222, ISIR, F-75005, Paris, France
stephane.doncieux@upmc.fr

Abstract

A method is introduced to extract and transfer knowledge between a source and a target task in continuous domains and for direct policy search algorithms. The principle is (1) to use a direct policy search on the source task, (2) extract knowledge from the learning traces and (3) transfer this knowledge with a reward shaping approach. The knowledge extraction process consists in analyzing the learning traces, i.e. the behaviors explored while learning on the source task, to identify the behavioral features specific to successful solutions. Each behavioral feature is then attributed a value corresponding to the average reward obtained by the individuals exhibiting it. These values are used to shape rewards while learning on a target task. The approach is tested on a simulated ball collecting task in a continuous arena. The behavior of an individual is analyzed with the help of the generated knowledge bases.

Introduction

Providing an agent the ability to learn remains a challenge although powerful learning algorithms have been developed. One of the reasons is that exploiting adapted representations as well as tips and tricks specific to a task to solve is mandatory for successful learning in many cases (Clark and Thornton, 1997). The knowledge an experimenter can include in the learning process comes from its experience, in particular from past attempts to solve a similar problem. *Can a robotic agent discover on its own what is important with respect to its morphology, environment and task? How can it transfer this knowledge from one task to another?* These questions are at the core of transfer learning (Taylor and Stone, 2009). They will be considered here in continuous domains and with no a priori defined states nor actions. The agent is given sensors and effectors and can move in a continuous environment to fulfill a task described by a reward function. The goal is to bootstrap a life-long learning process with as few a priori knowledge as possible about the tasks the robot is supposed to fulfill. The proposed method consists in (1) a task-agnostic direct policy search (2) an analysis of learning traces to extract knowledge and (3) the validation of acquired knowledge in a transfer learning setup.

The task-agnostic direct policy search relies on neuroevolution, i.e. on the simultaneous synthesis of neural network structures and optimization of their parameters with evolutionary algorithms (Floreano et al., 2008). It is task-agnostic as it requires only to define sensors and effectors as well as similarity measures to compare behaviors. No knowledge is required about how to solve the task. A direct encoding inspired from NEAT (Stanley and Miikkulainen, 2002) is used to explore robot behaviors with a state-of-the-art multi-objective evolutionary algorithm (Deb et al., 2002) driven by the reward defining the task, i.e. what to do and not how to do it, as well as a task agnostic behavioral diversity (Mouret and Doncieux, 2012). This method allows to generate robot controllers maximizing the reward in a non trivial source task.

The extraction of knowledge is the main contribution of this work. It relies on the following principle: *what is important is what is specific to successful behaviors*. The evolutionary algorithms used to learn the robot controller are population based search algorithms. Many different solutions are then explored in parallel, some being successful, and most failing to get any reward. After a selection process, new individuals are generated thanks to a mutation operator that makes random and, on average, small modifications to a parent individual[1]. Any individual has then a parent that has a relatively close behavior as they are separated by a single mutation. The lineage of an individual represents the set of all its parents, starting from the randomly generated individuals. The lineage of a best-of-run individual contains then failing individuals (typically the very first randomly generated individual), successful individuals (the best-of-run) and a set of individuals which have intermediate controllers and then intermediate behaviors in between the two. Our hypothesis is that the corresponding set of behaviors contains enough information to highlight the most relevant robot behavioral features. In these experiments, we used the raw sensori-motor values experienced by the corresponding controllers as behavioral features. A behavioral feature is the matrix of sensorimotor values during a given time length. These values are thus recorded and associated to the average reward of individuals exhibiting them. This will represent the knowledge base to be transferred from the

[1]Due to the permutation problem when evolving neural networks (Radcliffe, 1993), the crossover operator is seldom used. It is not used here.

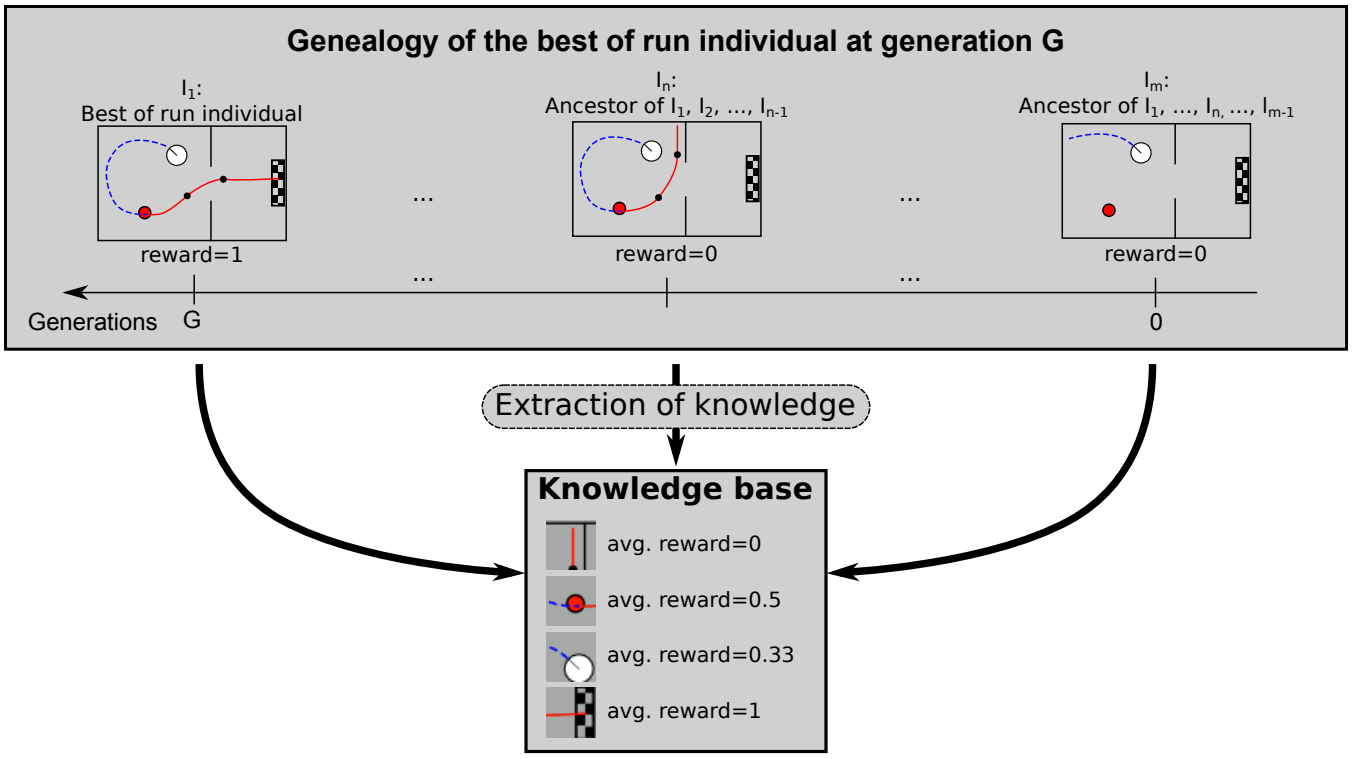


Figure 1: Overview of the proposed approach. The snapshots in the database correspond to robot sensor and effector values during a given time window.

source task to the target task. It will also provide insights on the most important features of robot behaviors.

The transfer of knowledge follows the principles of reward shaping (Dorigo and Colombetti, 1994; Mataric, 1994). When learning on the target task, the sensori-motor values experienced by an individual are compared to those of the knowledge base and the maximum average reward thus extracted is used as a separate objective to be maximized in a multi-objective setup. The rationale of this approach is to reward individuals that exhibit the same behavioral features as those exhibited by successful individuals.

The approach has been applied in simulation to a ball collecting task in a continuous environment.

Related work

Can we extract knowledge while learning to solve a source task and use it to more efficiently solve a target task? This question is at the core of transfer learning that has drawn a lot of attention in the machine learning community in general (Pan and Yang, 2010) and in reinforcement learning in particular (Taylor and Stone, 2009). The proposed transfer learning method relies on a shaping reward. It requires to keep the same agent space, i.e. the same sensors and effectors. This work is thus close to (Konidaris and Barto, 2006), except that there is no problem-space, as the learning algorithm explores directly in the space of policies. Madden and Howley proposed to transfer knowledge between environments with an increasing difficulty for discrete MDP (Madden and Howley, 2004). The goal of this work is similar, but in continuous domains instead of discrete ones.

Solving problems of increasing complexity while transferring knowledge acquired during each problem resolution process has been extensively used in Evolutionary Robotics where these approaches are called *incremental evolution*, *behavioral complexification* or *scaffolding* (Doncieux and Mouret, 2014). Transferred knowledge is typically the robot controller, i.e. the system that computes motors values out of sensor values (Kodjabachian and Meyer, 1997; Mouret and Doncieux, 2008; Bongard, 2011). In the proposed approach, the transferred knowledge is a database of observed behavioral features together with an associated value describing its relevance with respect to the task. It is thus independent from the formalism used to control the robot. It anyway implies to learn again, instead of relying on previously acquired controllers, but it has the advantage of providing insights on robot behaviors and could be used in a cognitive bootstrapping perspective (Windridge and Kittler, 2010; Mugan and Kuipers, 2012).

Method

The method consists in alternating between different phases: (1) learning a new behavior with evolutionary algorithms, e.g. with neuro-evolution, (2) analysis of learning traces with the proposed approach (Figure 1) and (3) exploitation of acquired knowledge in a reward shaping approach. The second step aims at discovering what is specific to the most

efficient individuals. (Doncieux, 2013) proposed to analyze the behavior of all generated controllers. It is proposed here to analyze only the lineage of one of the best individuals. The complexity is thus largely reduced as only a small subset of generated individuals needs to be analyzed.

Knowledge building process

The knowledge base KB contains behavioral features b_i, together with the average value of reward r_i obtained by the individuals that have exhibited b_i:

$$KB = \{(b_1, r_1), \ldots, (b_{KB_s}, r_{KB_s}) | \forall i, j, i \neq j \Rightarrow b_i \neq b_j\}$$

with KB_s the size of the knowledge base.

The definition of the behavioral features used in the experiments reported here is given in the next section.

The knowledge base building process requires a set of behaviors to analyze. This set is used to identify the behavioral features and to associate them with a good estimate of their value with respect to the reward provided to the system. In (Doncieux, 2013), the knowledge base was built out of the set of all the behaviors generated during the evolutionary search process. It is proposed here to build KB out of the analysis of the behavior of individuals that are in the lineage of a best-of-run. Once a run is terminated, the best-of-run is then extracted and the algorithm described in Algorithm 1 is executed. Much less behaviors need to be analyzed, resulting in a much faster knowledge building process than in (Doncieux, 2013).

For each behavioral feature the number of individuals that have exhibited it and the cumulated sum of rewards they have obtained are recorded. Behavioral feature update consists then in adding the fitness to the cumulated reward and increasing the corresponding number by one.

Algorithm 1 Algorithm of the knowledge extraction process.

```
I ←best-of-run individual
KB ← ∅
while I ≠ NULL do
    Generate I's behavior and extract:
    ... f(I) ∈ ℝ, the fitness of I
    ... B(I), the set of behavioral features exhibited by I
    for all b ∈ B(I) do
        if b ∉ KB then
            KB ← add(KB, b)
        end if
        KB ← update_reward(KB, b, f(I))
    end for
    I ← ancestor(I), NULL if no ancestor
end while
```

Knowledge exploitation

The exploitation of the database is done through a reward shaping approach: when a new individual is generated, its behavioral features are compared to that of the database. The maximum value of the set of behavioral features thus obtained is considered as a separate objective to maximize. If no behavioral feature belongs to the database, the reward shaping objective gets an arbitrary negative value (rewards are always positive). The reward shaping objective RS is thus defined over a knowledge base KB as follows:

$$RS_{KB}(x) = max\left(\{r_i | (b_i, r_i) \in KB \wedge b_i \in \mathcal{B}(x)\}\right)$$

where $\mathcal{B}(x)$ is the set of behavioral features exhibited by the agent controlled by x.

Experimental setup

Robot, environment and goal-oriented fitness

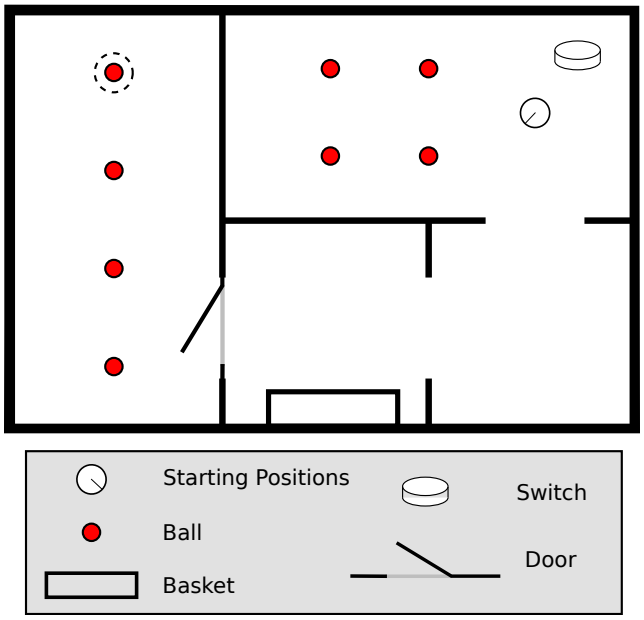


Figure 2: Simulated arena. The robot has to put balls into the basket. In the source task used for knowledge building, all balls are present. In the target task used for testing the knowledge base, only the ball surrounded by a dashed circle is present, and it appears only when the switch is activated. In this new setup, the door is removed. Once this ball has been collected, a new one can appear if the robot activates the switch again.

The experimental setup is the same as in (Doncieux, 2013). A simulated robotic agent has to collect balls in a continuous arena and put them into a basket. The robot controller drives the left and right wheel motors as well as the ball collecting system, that has two modes: collecting balls or releasing balls. The robot can get only one ball at a time. It receives one fitness point each time it puts a ball into the basket. To this end, it must release the ball while in front of the basket and touching it. In every cases, a released ball disappears from the environment. A switch that the robot can activate gives access to another room containing other balls. The environment used for knowledge building is shown in

Figure 2. It contains 8 balls and different rooms. Robot sensors can't see through walls, so the robot needs to explore the environment to get fitness points. Robot evaluation is done on one setup only and for 4000 time steps[2].

The robot has the following sensors:

- 3 distance sensors (continuous value)
- 2 ball detection sensors (front left and front right, binary value)
- 2 basket detection sensors (front left and front right, binary value)
- 2 switch detection sensors (front left and front right, binary value)
- 2 bumpers (front left and front right, binary value)
- 1 sensor to detect if the robot is carrying a ball (binary value)

The maximum robot displacement per time step is $4u$ (unit of length), the map size is $400 \times 600u^2$ and the robot radius is $20u$.

Behavioral features

The behavior of each individual is split into multiple, fixed size and non overlapping time windows. The sensorimotor flow during these time windows is recorded and stored in $m \times T$ matrices, where $m = nb_{sensors} + nb_{effectors}$ and T is the time window length (in the following, $T = 20$ time steps). These matrices are the behavioral features that will be considered to describe robot behavior.

Searching for a behavioral feature in the knowledge base consists in first looking for the closest matrix in the knowledge base. If the distance to the closest matrix is below a threshold, then the behavioral feature is considered to be part of the knowledge base, otherwise, it is considered not to be part of it. To accelerate the search, the FLANN library has been used for nearest neighbor search (Muja and Lowe, 2009)[3].

Experiments on real robots and with more realistic sensors should consider using more sophisticated behavioral features, like representations generated by networks trained with deep learning algorithms (Bengio, 2011).

Robot controller

The robot is controlled by a neural network whose structure and parameters are explored by an evolutionary algorithm with an approach based on NEAT (Stanley and Miikkulainen, 2002), but without using crossover and innovation numbers. Starting from a feed-forward neural network without any hidden neuron and connecting all inputs to all outputs, mutations can add or remove connections as well as nodes. Mutations can also change connection weights. Neuron activation function is a sigmoid function. Corresponding parameters are given in appendix.

Dynamic behavioral diversity

Besides the goal-oriented objective that counts the number of balls put into the basket, a dynamic behavioral diversity objective was used (Doncieux and Mouret, 2013). This objective revealed to significantly improve performance with respect to behavioral diversity proposed in (Mouret and Doncieux, 2012). The results obtained here confirm these results. The objective BD to maximize measures the distance between the considered individual x and the rest of the population in the behavioral space. It is thus computed as follows:

$$BD(x) = \frac{1}{N} \sum_{y \in P} d_k(x, y)$$

with P the population for the current generation, N its size and x, y individuals. $d_k(x, y)$ is a behavioral similarity measure. Dynamic behavioral diversity is similar to behavioral diversity (Mouret and Doncieux, 2012), except that the behavioral similarity measure is randomly changed at a given generation period p_g. M different behavioral similarity measures are available $(d_1, d_2, \ldots, d_M)$, and at every p_g generation a new value of k is randomly chosen between 1 and M (in this work, $p_g = 50$ generations).

Four different behavioral similarity measures were used:

- adhoc (Ollion and Doncieux, 2011): the vector of ball positions at the end of an evaluation is the behavior descriptor and the behavioral similarity measure is the Euclidian distance between these vectors;
- hamming (Mouret and Doncieux, 2012): the behavioral similarity measure is the Hamming distance between vectors of discretized sensorimotor values (60000 bits vector = 4000 $\times$ (12 sensors + 3 effectors));
- trajectory (Ollion and Doncieux, 2011): the behavioral similarity measure is the edit distance between the discretized trajectories of the robots (Trujillo et al., 2011);
- entropy (Delarboulas et al., 2010): the entropy is computed for each sensor and effector. For binary sensors, i.e. bumpers, balls, basket and switch sensors, the entropy has been computed as follows:

$$E = -\sum_{i=0,1} p_i \frac{ln(p_i)}{ln(2)}$$

with p_i the probability that the corresponding sensor takes the value i during the evaluation. For sensors and effectors with a continuous value, the range of possible values has been divided in ten different subranges and the entropy has been computed as follows:

$$E = -\sum_{i=1}^{10} p_i \frac{ln(p_i)}{ln(10)}$$

with p_i the probability that the sensor or effector value belongs to the i-th subrange. The behavior descriptor is then the vector of the entropies of each sensors and effectors. The corresponding distance is an Euclidean distance between those vectors;

[2] in (Doncieux, 2013), each robot was evaluated in 3 different conditions. This modification aims at saving computational time.

[3] http://www.cs.ubc.ca/research/flann/

List of treatments

All treatments rely on the multi-objective evolutionary algorithm NSGA-II (Deb et al., 2002). The only difference between treatments are the objectives optimized during the evolutionary process. Following the classification of (Doncieux and Mouret, 2014), the number of balls is a goal-oriented objective, i.e. the only one important at the end of a run. This is the objective that will be taken into account for computing average rewards in the knowledge base. Dynamic behavioral diversity and reward shaping objectives are process helpers whose value at the end of a run is not important to decide which individual is the best-of-run. They are only aimed at making the learning process more efficient.

A single treatment has been considered for the source task in which two objectives are to be maximized:

1. number of balls;
2. BD: dynamic behavioral diversity.

On the target task, three objectives can be used:

1. number of balls;
2. BD: dynamic behavioral diversity;
3. RS_{KB}: reward shaping objective.

Several runs have been launched on the source task. Because of the stochastic properties of Evolutionary Algorithms, the corresponding best-of-run individuals are not the same and actually have different fitnesses. Knowledge bases have been extracted from the lineage of the best-of-run individuals of four different runs that ended up with different numbers of collected balls. The treatments that have been considered on the target task are thus the following:

- KB 1:
  - objectives 1, 2 and 3;
  - KB1 generated from a best-of-run with a fitness of 7 (balls collected out of a maximum of 8), contains 23734 behavioral features.
- KB 2:
  - objectives 1, 2 and 3;
  - KB2 generated from a best-of-run with a fitness of 6, contains 21237 behavioral features.
- KB 3:
  - objectives 1, 2 and 3;
  - KB3 generated from a best-of-run with a fitness of 5, contains 23818 behavioral features.
- KB 4:
  - objectives 1, 2 and 3;
  - KB4 generated from a best-of-run with a fitness of 3, contains 31421 behavioral features.
- KB1 all:
  - objectives 1, 2 and 3;
  - KB generated from the same experiment used to generate KB1, but while considering all generated behaviors, as in (Doncieux, 2013) instead of the lineage of the best of run individual. It contains 60848 behavioral features.
- No transfer (control exp.): objectives 1 and 2.
- KB1 random (control exp.): objectives 1, 2 and 3 with KB1 in which average rewards have been replaced by random values.

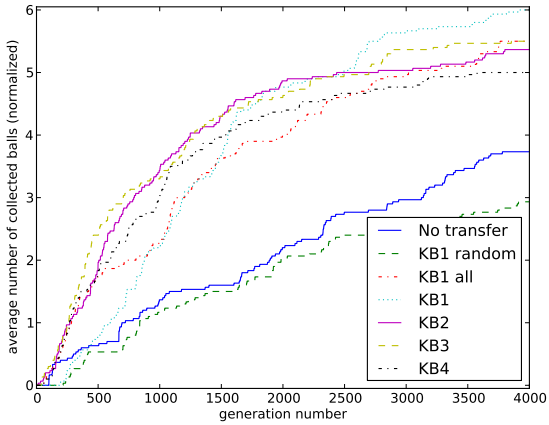


Figure 3: Average number of collected balls of best-of-run individuals along generations for each treatment (average over 30 runs).

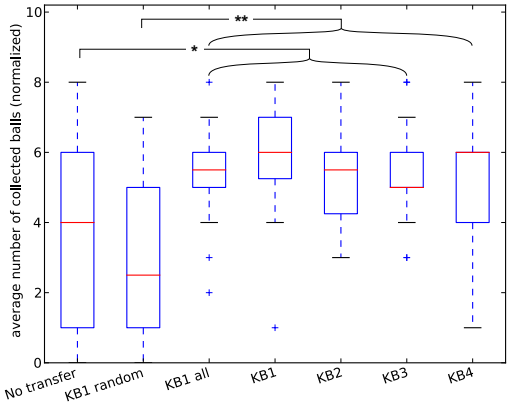
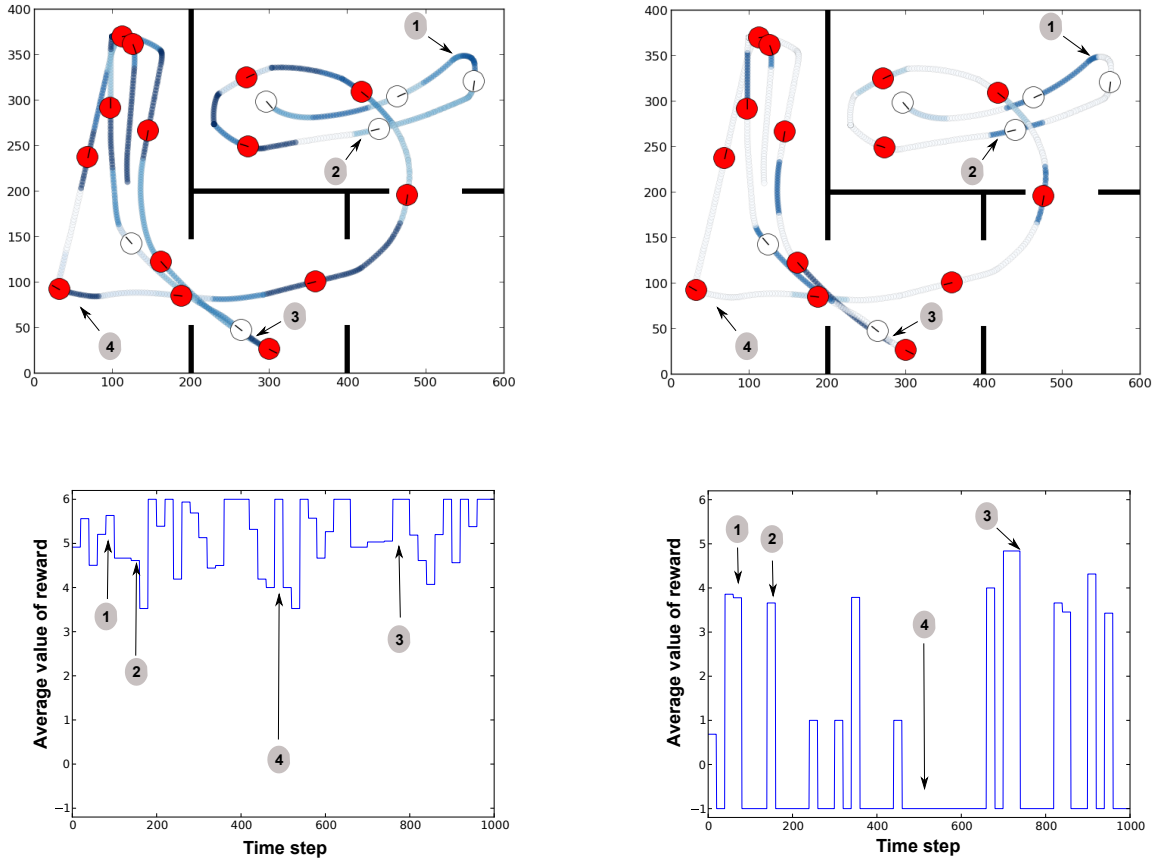


Figure 4: Boxplots of the maximum number of collected balls for each treatment at generation 4000 (30 runs). '*' means statistically different with a $p - value < 10^{-2}$ (Mann-Whitney test), '**' means $p - value < 10^{-3}$, details of $p - values$ are given in appendix.

The difference between KB1 and KB1 all lies in the number of individuals used to generate the knowledge base. For KB1 all, the behavior of all individuals generated during the evolutionary run is added to the knowledge base, i.e. 800,000 individuals (4000 generations $\times$ 200 individuals generated per generation). For KB1, only the individuals in the lineage of the best-of-run are used. In this experiment, it is less than 600. The knowledge building process relying on



(a) (b)

Figure 5: Trajectory of the best-of-run individual used to generate KB3 colored by the values of the corresponding behavioral features. (a) values from KB3, (b) values from KB2. Top: trajectory, white means low values and blue high values, the robot is pictured every 50 time steps in white if the robot does not carry a ball and in red if it carries a ball. Bottom: value with respect to time steps (-1 means that the value has not been found in the knowledge base). See text for comments on numbered parts of the trajectories.

the lineage is then more than 1,000 faster to the one used in (Doncieux, 2013).

Parameters of the algorithm are given in appendix. The source code of the experiments is available on `http://pages.isir.upmc.fr/evorob_db/`.

Results and analysis

Performance on the source and target task

The experiments on the source task had an average value of 4.25 balls collected (max = 8) after 4000 generations.

On the target task, the knowledge base revealed to significantly improve the learning speed with an asymptotic improvement, at least over the considered budget (Figure 3). This observation is true no matter where the knowledge base comes from and no matter how it is computed (all individuals or over best-of-run lineage only). The difference is significant after around 1000 generations and even before for some treatments (see Figure 4 for the performance at generation 4000). The control experiment with a randomized average reward value is not statistically different from the experiment that did not use a knowledge base. It means that the third objective per se can't explain this difference of performance and thus that the values associated to behavioral features actually carry some useful information with respect to the task. It thus validates, at least on this experimental setup, our knowledge extraction approach.

The lack of difference between the different knowledge bases is surprising as one would expect that the knowledge base associated to a more fit individual would lead to a higher performance as this base should help identify more useful information about the robot, its environment and the task. Anyway, the shaping objective that exploits the knowledge base, aims at bootstrapping the learning process by rewarding stepping stones towards efficient behaviors. Few intermediate behaviors may actually be enough to bootstrap learning. All knowledge bases have been generated out of

individuals able to collect several balls and thus to move around in the environment, pick up a ball, keep it, look for the basket and go towards it and then release the ball. Even if the corresponding behaviors are not all optimal, the knowledge base built from them can recognize and encourage their generation on the target task.

What is recognized as important?

The knowledge base associates a value to behavioral features. Plotting trajectories colored by the value of the corresponding behavioral feature indicates then what is considered to be important for one particular knowledge base.

Figure 5 shows an example of trajectory for the source task, colored by the values from two different databases. When colored by the knowledge base built out of the same individual, i.e. KB3, all behavioral features have been found in the knowledge base (by construction) and most of them have a high value. This is expected as the knowledge in KB3 is associated to the behavior of this particular individual. This plot thus highlights the main features of this behavior with respect to close behaviors (in terms of genotypic distance and thus, to some extent in terms of behavioral distance) that were less successful. Some of them make sense from a human observer point of view: 1. the robot activates the switch, 2. the robot picks up a ball, 3. the robot goes towards the basket to drop the ball. Some others are more difficult to interpret, like 4. the robot collides with the wall after going though the door. This behavior was probably necessary for the navigation strategy of the robot. This particular feature was for instance not in KB2, whereas 1., 2. and 3. also belong to KB2 and are associated to high values..

Combining the values coming from different knowledge bases should lead to the identification of the features that are salient for the set of behaviors out of which these knowledge bases were built. To this end, it is proposed to associate to each behavioral feature the product of the values coming from each knowledge base (with 0 when the value has not been found in the database). Figure 6 shows the result of such a combination for KB1, KB2 and KB3[4]. Much less behavioral features have a non null value and they are easy to interpret: 1. activating the switch, 2. going towards a ball to pick it up, 3. entering the second room (discovering the basket), 4. approaching the basket while carrying the ball, 5. going back to find new balls after having released a ball.

Conclusion

The proposed approach allows to extract, from evolutionary traces, the relative importance of behavioral features with respect to robot features, robot environment and fitness function. The relevance of this knowledge has been validated in a transfer learning experiment in which a more difficult version of the task has been considered: experiments exploiting the knowledge base did converge faster and to more efficient solutions. A knowledge base with a randomized value did not generate the same results, thus confirming the significance of the values attached to behavioral features.

[4]KB4 has been neglected as the corresponding behavior did not use the switch.

The observation of the values of behavioral features along a trajectory shows, for one particular behavior, what is considered to be important in the knowledge base. As the knowledge base is built from the lineage of a best-of-run, it is somehow specialized in recognizing what is important in one particular behavior. It highlights behavioral features that are easy to understand from a human observer point of view, but also features that are specific to the strategy used by this best-of-run individual to solve the problem. Interestingly, combining the values from different knowledge bases highlights the features that are shared between all the corresponding strategies and that are thus the more general and unavoidable to get the reward.

This approach could be used in a cognitive bootstrapping setup to face the curse of dimensionality and focus the acquisition of new states and actions on the most promising ones. While the database is still large, future work should try to reduce its size, for instance by keeping only the behavioral features with the highest values or those that make a consensus between different knowledge bases.

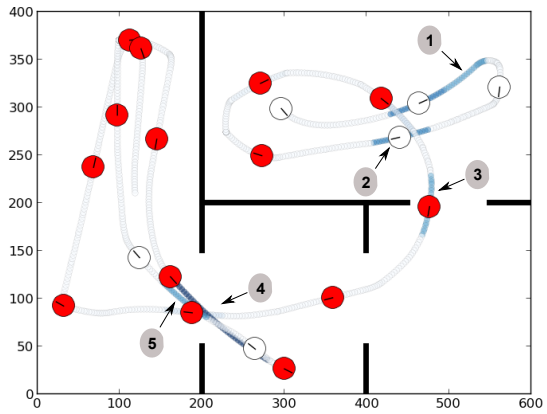


Figure 6: Trajectory of the best-of-run individual used to generate KB3 colored by the product of the values of the corresponding behavioral features in KB1, KB2 and KB3. See text for comments on numbered parts of the trajectory.

Acknowledgement

This work is supported by the ANR CreAdapt project (ANR-12-JS03-0009).

References

Bengio, Y. (2011). Deep Learning of Representations for Unsupervised and Transfer Learning. *Journal of Machine Learning Research-Workshop and Confenrence Proceedings*, 7:1–20.

Bongard, J. C. (2011). Morphological and environmental scaffolding synergize when evolving robot controllers. In *Proc. of the international conference on Genetic and Evolutionary Computation Conference (GECCO'11)*, pages 179–186.

Clark, A. and Thornton, C. (1997). Trading spaces: computation, representation, and the limits of uninformed learning. *Behavioral and Brain Sciences*, pages 57–90.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Delarboulas, P., Schoenauer, M., and Sebag, M. (2010). Open-ended evolutionary robotics: an information theoretic approach. *Parallel Problem Solving from Nature*, (216342).

Doncieux, S. (2013). Transfer Learning for Direct Policy Search : A Reward Shaping Approach. In *Proceedings of the IEEE ICDL-EpiRob conference*.

Doncieux, S. and Mouret, J.-B. (2013). Behavioral diversity with multiple behavioral distances. *2013 IEEE Congress on Evolutionary Computation*, 2013:1427–1434.

Doncieux, S. and Mouret, J.-B. (2014). Beyond Black-Box Optimization: a Review of Selective Pressures for Evolutionary Robotics. *Evolutionary Intelligence*, pages 1–23.

Dorigo, M. and Colombetti, M. (1994). Robot shaping : developing autonomous agents through learning. *Artificial intelligence*, 71:321–370.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.

Kodjabachian, J. and Meyer, J. A. (1997). Evolution and development of Neural Networks Controlling Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects. *IEEE Transactions on Neural Networks*, 9:796–812.

Konidaris, G. D. and Barto, A. (2006). Autonomous shaping: Knowledge transfer in reinforcement learning. In *ICML '06 Proceedings of the 23rd international conference on Machine learning*, pages 489—-496. ACM.

Madden, M. and Howley, T. (2004). Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, (1986):375–398.

Mataric, M. (1994). Reward Functions for Accelerated Learning. In *ICML '94 Proceedings of the 11th international conference on Machine learning*, pages 181—-189. Morgan Kaufman.

Mouret, J.-B. and Doncieux, S. (2008). Incremental Evolution of Animats' Behaviors as a Multi-objective Optimization. In *From Animals to Animats 10*, volume 5040, pages 210–219. Springer.

Mouret, J.-B. and Doncieux, S. (2012). Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study. *Evolutionary computation*, 20(1):91–133.

Mugan, J. and Kuipers, B. (2012). Autonomous Learning of High-Level States and Actions in Continuous Environments. *IEEE Transactions on Autonomous Mental Development*, 4(1):70–86.

Muja, M. and Lowe, D. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, pages 331–340.

Ollion, C. and Doncieux, S. (2011). Why and How to Measure Exploration in Behavioral Space. In *GECCO '11: Proceedings of the 13 th annual conference on Genetic and Evolutionary Computation*, pages 267–294.

Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Radcliffe, N. J. (1993). Genetic set recombination and its application to neural network topology optimisation. *Neural computing and applications*, 1(1):67–90.

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

Taylor, M. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685.

Trujillo, L., Olague, G., Lutton, E., Fernández de Vega, F., Dozal, L., and Clemente, E. (2011). Speciation in Behavioral Space for Evolutionary Robotics. *Journal of Intelligent & Robotic Systems*, 64(3):323–351.

Windridge, D. and Kittler, J. (2010). Perception-action learning as an epistemologically-consistent model for self-updating cognitive representation. *Advances in experimental medicine and biology*, 657:95–134.

Appendix

Statistical significance of the results on the target task (Mann-Whitney test) at generation 4000:

	No transf.	KB1 r.	KB1 all	KB1	KB2	KB3	KB4
No tr.	1	0.1248	<10e-2	<10e-3	<10e-2	<0.01	0.026
KB1 r.	0.1248	1	<10e-4	<10e-6	<10e-4	<10e-4	<10e-3
KB1 all	<10e-2	<10e-4	1	0.043	0.34	0.44	0.29
KB1	<10e-3	<10e-6	0.043	1	0.020	0.039	0.016
KB2	<10e-2	<10e-4	0.34	0.020	1	0.40	0.41
KB3	<10e-2	<10e-4	0.44	0.039	0.40	1	0.33
KB4	0.026	<10e-3	0.29	0.016	0.41	0.33	1

Parameters of the knowledge extraction process:

- Time window length, $T = 20$
- Threshold for matrix distance, 1.83

Parameters common to all the treatments:

- MOEA: NSGA-II (pop. size : 200)
- DNN (direct encoding):
  - number of neurons$\in [10, 30]$
  - number of connections $\in [50, 250]$
  - prob. of changing weight/bias: 0.1
  - prob. of adding/deleting a conn.: 0.15/0.05
  - prob. of changing a conn.: 0.03
  - prob. of adding/deleting a neuron: 0.05/0.05
  - activation function for neurons:
    $y_i = \varphi\left(\sum_j w_{ij} x_j\right)$ where $\varphi(x) = \frac{1}{1+\exp(b-x)}$