# Transfer learning for direct policy search: a reward shaping approach

Stéphane Doncieux

# Transfer Learning for Direct Policy Search: A Reward Shaping Approach

Stephane Doncieux
ISIR, Université Pierre et Marie Curie-Paris 6,
CNRS UMR 7222
4 place Jussieu, F-75252, Paris Cedex 05, France
Email: doncieux@isir.upmc.fr

***Abstract*—In the perspective of life long learning, a robot may face different, but related situations. Being able to exploit the knowledge acquired during a first learning phase may be critical in order to solve more complex tasks. This is the transfer learning problem. This problem is addressed here in the case of direct policy search algorithms. No discrete states, nor actions are defined a priori. A policy is described by a controller that computes orders to be sent to the motors out of sensor values. Both motor and sensor values can be continuous. The proposed approach relies on population based direct policy search algorithms, i.e. evolutionary algorithms. It exploits the numerous behaviors that are generated during the search. When learning on the source task, a knowledge base is built. The knowledge base aims at identifying the most salient behaviors segments with regards to the considered task. Afterwards, the knowledge base is exploited on a target task, with a reward shaping approach: besides its reward on the task, a policy is credited with a reward computed from the knowledge base. The rationale behind this approach is to automatically detect the stepping stones, i.e. the behavior segments that have lead to a reward in the source task before the policy is efficient enough to get the reward on the target task. The approach is tested in simulation with a neuroevolution approach and on ball collecting tasks.**

## I. Introduction

Exploiting the knowledge acquired one day to make the robot more efficient another day is of critical importance for life long learning [18]. Thanks to its experience, the robot may learn faster or solve a task that was out of reach before. This is the *transfer learning* approach [13], [19]. Whereas this approach has drawn much attention in the machine learning community, it remains an open question in the robotics field.

Different approaches to transfer learning have been proposed based on the reinforcement learning paradigm [16]. The value function evaluated in the source task can be a starting point to learn the target task[17]. Another alternative consists in developing an algorithm that chooses between the old policy and the current policy under construction[3]. The past policy is followed with a probability $\psi$ while the new policy is exploited with an $\epsilon$-greedy strategy. Knowledge can also be inserted directly in the definition of the reward function. This leads to the reward shaping approach[6], [14], [4]. Whereas this approach has been developed initially to include expert knowledge in order to facilitate learning, it has also been used in a context of transfer learning in reinforcement learning, thus using past experience to automatically build the shaping function. Previous learning episodes can be used to train a function estimator that predicts the value function, i.e. the expected reward, out of a state in an agent centered space[5]. Other work has even proposed to automatically identify the relevant features to take into account in the shaping function[15].

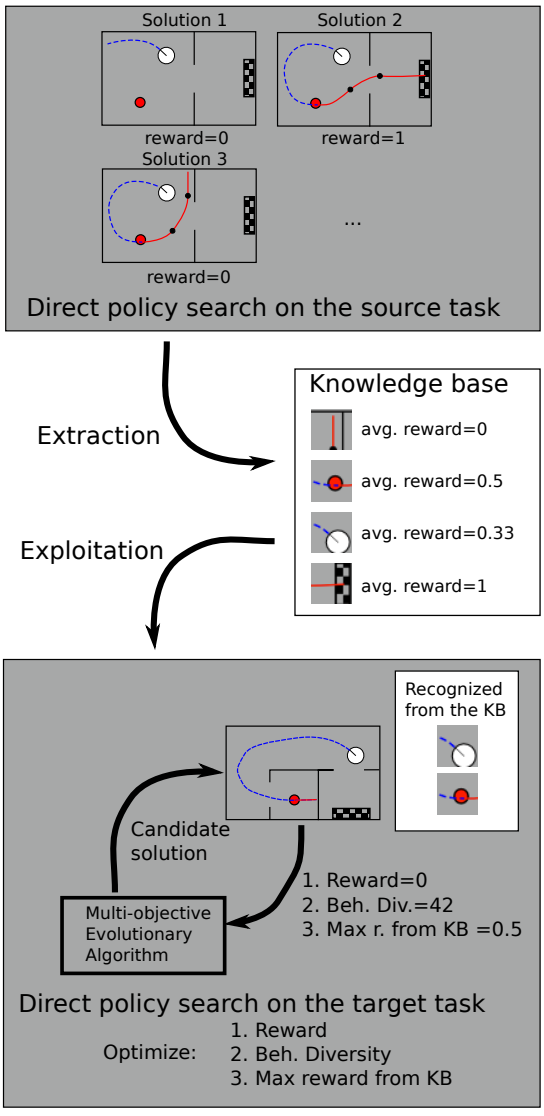


Fig. 1. Overview of the proposed approach.

All the previously mentioned work rely on the definition of discrete states and actions. This work focuses on continuous

states and actions. The rationale of the proposed approach is the following (figure 1):

1) learn to solve the source task from scratch;
2) extract knowledge from this learning episode by identifying *salient* behavior segments;
3) use this knowledge to learn how to solve similar tasks while rewarding the appearance of those behavior segments;

The proposed approach consists segmenting the behaviors into fixed length segments. The saliency of a behavior is the average reward of the policies that exhibit it: salient behaviors will be those that are mostly exhibited by efficient controllers. To have an accurate estimation of elementary behavior saliency, many different policies need to be explored, both efficient and inefficient. Evolutionary algorithms are used for the direct policy search. Their population based feature makes them explore a large number of policies in parallel, thus providing a large set of policies from which to build the knowledge base. A reward shaping approach [6], [14], [4] is used to exploit the knowledge acquired on the source task. A reward is computed out of the knowledge base and taken into account in the learning process for the target task.

Sequential tasks have been considered with a direct policy search based on neuro-evolution.

## II. Method

The proposed approach has two main steps: (1) a knowledge base building step and (2) a knowledge exploitation step (figure 1). Step (1) is performed on a source task and step (2) is performed on a related target task. The approach is meant to be used with learning algorithms that explore many different policies, e.g. evolutionary algorithms.

### A. Behavior representation

The behavior of a policy can be represented by the sequence of sensor and effector values, i.e. by a $(n_s + n_e) \times T$ matrix, if $n_s$ is the number of sensors, $n_e$ the number of effectors and $T$ the number of time steps. To extract knowledge from this matrix, it is first decomposed into segments with a predefined width $w$. The sequence of these behavior segments describes the behavior.

### B. Knowledge base building process

The knowledge base is made up with behavior segments associated with the average value of the rewards of all the policies exhibiting this behavior segment[1].

During the learning step on the source task, at a given period, every policy $\rho$ explored in a generation $g$ of the evolutionary algorithm, is involved in the knowledge base building process. A period of 1 will build the most accurate knowledge base, while a higher period will save computational time[2]. Each behavior segment $bs$ of $\rho$ is compared to each behavior segment in the knowledge base. If the knowledge base contains a behavior segment $bs'$ closer to $bs$ than the threshold $\delta$, the cumulated reward of $bs'$ is increased by the fitness of $\rho$ and its number incremented by one. Otherwise, $bs$ is added to the knowledge base with a cumulated reward equal to the reward of $\rho$ and a number of 1. The distance used to compare behavior segments is an euclidean distance.

At the end of the learning process, the knowledge base contains then the set of the behavior segments encountered with their cumulated reward and number of appearance, i.e. with their average reward on all the explored policies. This average reward is an estimation of the contribution of the behavior segments to the source task resolution.

[1]As the average value is computed incrementally, the knowledge base contains behavior segments together with their cumulated reward and the number of policies in which they have been found.

[2]In the experiments reported here, a period of 200 has been used.

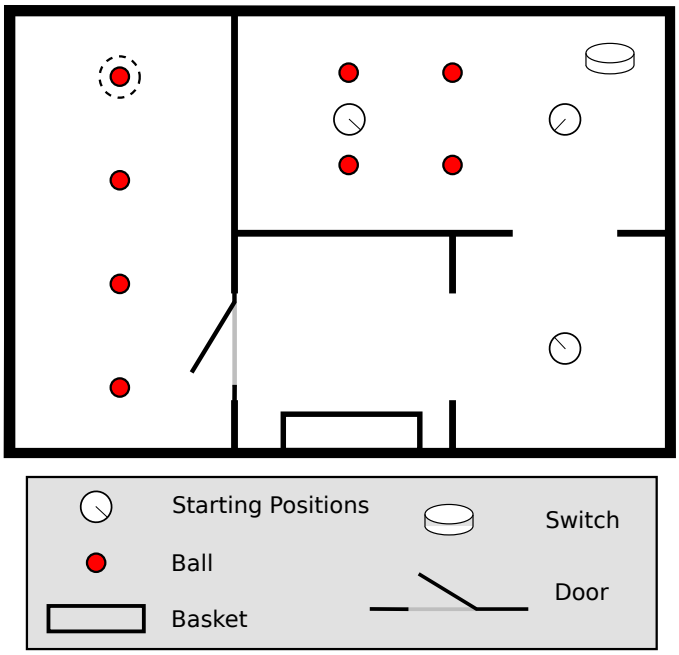


Fig. 2. Overview of the arena and of the robot for the source task. The goal of the experiment is to place as many balls as possible into the basket. A robot controller is evaluated with three different initial positions. To reach the four balls in the left room, the robot has to open the door by first pressing the switch button. The target task uses the same arena, but with no more than one ball at a time (the one circled with a dashed line). In this setup, the door is always opened and the switch makes a ball appear (at the beginning, no ball are in the arena).

### C. Exploitation of the knowledge base

On the target task, the knowledge base is used to estimate if a particular policy contains "salient" behaviors. Each behavior segment of the policy is sought into the knowledge base. The maximum reward thus obtained $r_{max}$ is taken into account in the learning process.

Two approaches have been tested to take this value into account. The first consists in aggregating it with the reward obtained by an policy, for instance with a sum. The second approach consists in considering $r_{max}$ as a separate objective to be maximized by the evolutionary learning process.

## III. Experimental setups

The considered tasks are ball collecting tasks inspired from [12] (figure 2). A two-wheeled simulated robot has to pickup

balls and put them into a basket. The reward of a policy is the number of balls put into the basket while the robot executes it. The robot has two wheels and twelve sensors: three wall distance sensors, two bumpers, two ball detection sensors, two switch sensors, two basket detection sensors and one "carrying ball" sensor. It has three effectors: left and right motors as well as an "action" effector. A nearby ball is collected or the switch is activated if the value of the "action" effector value is above 0.5 (the robot can carry only one ball at a time). If it is below 0.5, a carried ball is released. A released ball disappears from the arena. If, at that time, the robot was in front of the basket and touching it, then the ball is considered to be collected and one reward point is given to the robot.

### A. Source task

The source task is that described in [12]. The arena contains four balls and a switch button that allows the robot to open the door of a room containing four more balls. The robot starts from three different initial positions as shown on figure 2. The initial positions are always the same so that individuals are evaluated in the same conditions.

### B. Target task

The target task relies on the same arena than the source task, but no ball is present in the arena at the beginning of an evaluation. Activating the switch make them appear, one at a time, at the top left ball position (see figure 2), i.e. at the position that is the most distant to the switch. The door is always opened in this setup.

In this task, no reward is obtained before the robot learns to go towards the switch, activate it, navigate in the arena to find the ball, pick it up, go towards the basket, touch it and release the ball. This sequence allows the robot to get a reward of one. It has to repeat it to get other reward points. The sequence to get a reward point is then longer than for the source task, making it more difficult to solve.

### C. Neuro-evolution

The robot is controlled by a neural network with twelve inputs and three outputs. Both structure and parameters of the neural network are generated by an evolutionary algorithm, using the DNN encoding as described in [10]. A learning process starts with a perceptron with no hidden neuron that directly links input neurons to output neurons. Connection weights are chosen randomly. No crossover is used and mutations can add or remove neurons and connections and change connection weights. The parameters used for these experiments are listed in the appendix.

NSGA-II, a state-of-the-art multi-objective evolutionary algorithm, is used for the direct policy search [1]. Several objectives are optimized at the same time: the number of collected balls and some other helper objectives (see text below). The best-of-run policy is the one that has collected the highest number of balls. The other objectives are used to enhance the search, but are neglected when observing the solutions generated in a run.

### D. Treatments

In the source task, the objectives to optimize are the reward, i.e. the number of collected balls and a behavioral diversity objective, used to foster exploration [10]. This objective measures the distance, in the space of behaviors, of a policy towards the other policies in the population at a given generation of the evolutionary algorithm. This objective revealed to increase the efficiency of the direct policy search algorithm in multiple contexts [10], [2], [8], [7]. The behavior distance used to compute the behavioral diversity objective is an edit distance on the discretized robot trajectory, as described in [12].

The knowledge base is used with the two proposed approaches on the target task, leading to the following treatments:

- *sum*: the maximum average reward associated to the behavior segments of a policy out of the knowledge base is normalized and added to the reward of the policy on the task[3]. Two objectives are maximized:
  1) reward on the task (number of collected balls) + max behavior segment reward from the knowledge base
  2) behavioral diversity
- *mo*: multi-objectivization approach, the maximum average reward associated to the behavior segments of a policy out of the knowledge base is added as a new objective. Three objectives are maximized:
  1) reward on the task (number of collected balls)
  2) behavioral diversity
  3) maximum behavior segment reward from the knowledge base

For control experiments, a new knowledge base has been created. It contains the same behavior segments than the knowledge base used before, but associated with random average rewards, drawn from a uniform distribution in the interval $[0, 1]$. The control treatments are the following:

- *sum rand*: the *sum* treatment, but with the randomized knowledge base
- *mo* rand: the *mo* treatment, but with the randomized knowledge base
- *no transfer*: the same treatment than for the source task, i.e. the following objectives to maximize:
  1) reward on the task (number of collected balls)
  2) behavioral diversity

### E. Parameter settings

The main parameter of the proposed method is the $\delta$ threshold parameter. It has been set to $1.83$, that is the mean distance between the 20 closest behaviors during 4000 time steps of a successful behavior generated on the source task. This has been empirically chosen to limit the size of the knowledge base while gathering behavior segments reasonably similar.

Segments of width 20 time steps have been considered. To save computation time, the considered segments do not

[3] after normalization, a policy that has collected $n$ balls has always a lower fitness than a policy that has collected $n + 1$ balls.

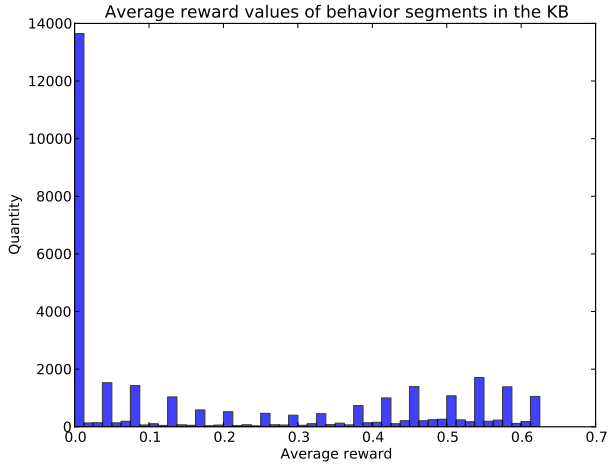


Fig. 3. Histogram of the number of behavior segments in the knowledge base with a particular average reward. The knowledge base comes from a single learning experiment on the source task.

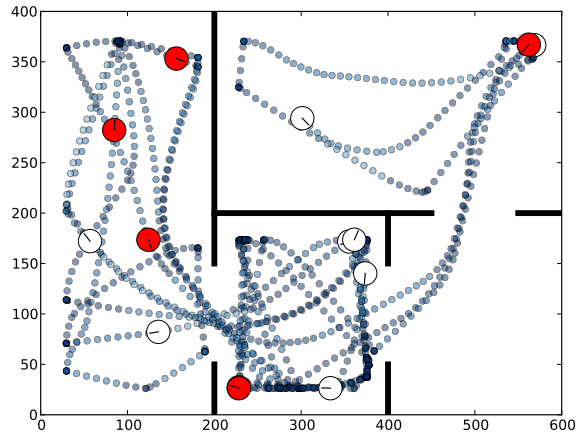


Fig. 4. Trajectory of the best-of-run policy on the source task. Small circles represent the trajectory. Their color represents the average reward of the corresponding behavior segment in the knowledge base: white if the average reward is 0, dark blue for the highest value. The robot orientation is periodically represented in white when the robot does not carry a ball and in red when it carries a ball.

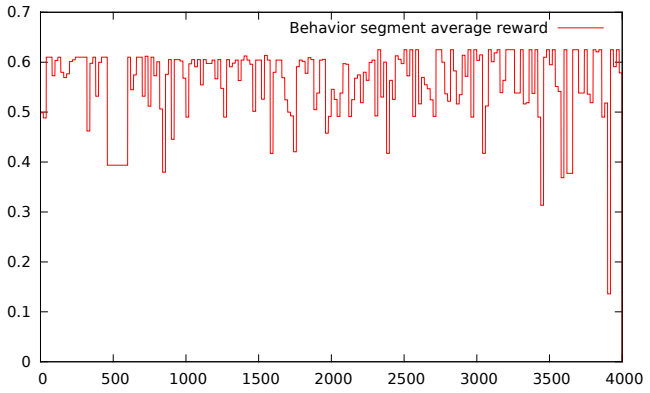


Fig. 5. Behavior segment average reward of the best-of-run policy on the source task. X-axis: time steps, Y-axis: average reward.

overlap. The evaluation of a policy lasts 12000 time steps (4000 time steps for each initial position). The arena size is $600 \times 400$ units and the robot can move up to 4 units per time step. A maximum number of 4000 time steps have been considered for the knowledge base building process or exploitation: only the behavior of the policy when starting from the first initial position has been taken into account.

To speedup the search of a behavior segment into the knowledge base, a KD-tree structure is used as implemented in the FLANN library [11]. It allows a fast approximate search for nearest neighbors[4].

The experiments have been implemented in the $SFERES_{v2}$ software framework [9], which is a framework for evolutionary algorithms and evolutionary robotics experiments[5]. The source code of the experiment is available from http://pages.isir.upmc.fr/evorob_db.

## IV. Results

### A. Knowledge base content

The knowledge base generated after 4000 generations by the best run out of 30 different runs has been selected. A reward of 0.625 has been reached, i.e. 15 balls out of 24 are collected by the best policy. The knowledge base contains 32580 behavior segments. The average reward of the behavior segments range from 0 to 0.625, i.e. the maximum reward gathered in this run. Figure 3 shows the repartition of behavior segments average rewards. Almost 14000 behavior segments are associated to a 0 reward. It corresponds to behavior segments that have been mostly found in policies that did not get any reward. Around 1000 behavior segments have an average reward of 0.625. As the reward is an integer, the average rewards follows a comb distribution.

### B. Analysis of a best-of-run policy on the source task

The most efficient behavior found in this experiment can be analyzed through the rewards of its behavior segments in the knowledge base. It leads to the trajectory plotted on figure 4. The corresponding behavior segments have an average reward that mostly range from 0.4 to 0.625 (figure 5). It means that the behavior segments are mostly found in policies that get a significant amount of reward. The highest values are reached after time step 2000, when the robot starts collecting the balls in the left room.

### C. Impact of the knowledge base while learning to solve the target task

The number of successful runs along generations is shown on figure 6. A run is considered as successful if the best-of-run policy collects more than half of the available balls.

[4]The source code of the library used for these experiments is available from http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN.

[5]The software is open-source and can be downloaded from http://sferes2.isir.upmc.fr/.

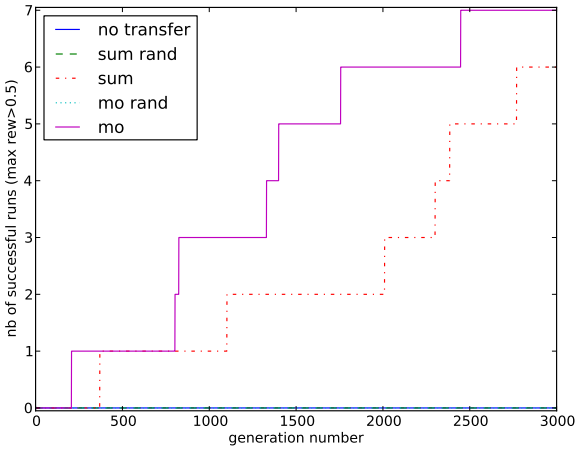


Fig. 6. Number of successful runs for each treatment on the target task out of 20 independent runs. A run is considered as successful if the best policy on the task has a reward of 0.5, i.e. half of the balls are collected.

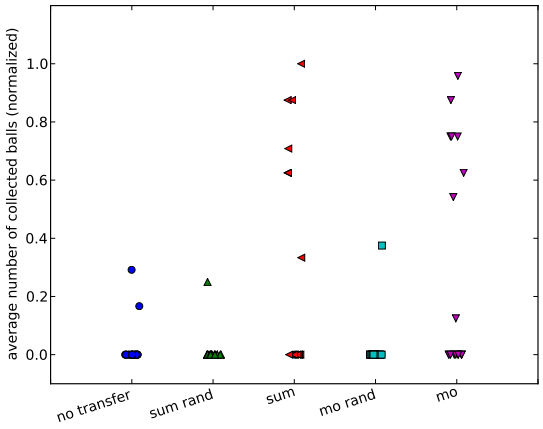


Fig. 7. Performance of the different treatments on the target task. Results at generation 3000 on 20 runs. Each symbol represents the reward of the best policy of a run. Small random variations have been added to the X-axis in order to make the cloud of points easier to look at.

Both *sum* and *mo* treatments exhibit a significantly better performance (see appendix for the results of a Mann-Whitney test at generation 3000). Efficient solutions for both tasks are thus generated when the knowledge base is taken into account. The average reward associated to each behavior segment in the knowledge base seems then significant and useful as the performance of the runs exploiting the randomized version of the knowledge base is very low. Only a few runs of the control experiment did succeed in collecting some balls (maximum of 0.4), while the *sum* and *mo* treatments did succeed in collecting up to all available balls. It should be noticed that the reward generated with *sum* and *mo* treatments shows a high variability (figure 7): some runs have bootstrapped, and some have not been able to collect a single ball. The success rate at generation 3000 for *sum* (resp. *mo*) is 35% (resp 30%). It is 0% for all control experiments.

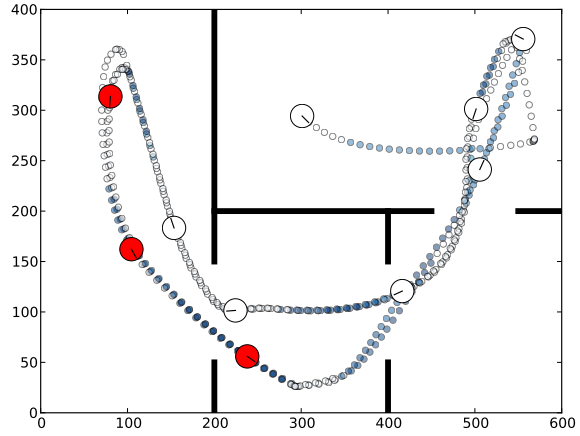

Fig. 8. Trajectory of the best-of-run policy on the target task. Small circles represent the trajectory. Their color represents the average reward of the corresponding behavior segment in the knowledge base: white if the average reward is 0, dark blue for the highest value. The robot orientation is periodically represented in white when the robot does not carry a ball and in red when it carries a ball. Only a part of the trajectory is shown for clarity.

### *D. Analysis of a best-of-run policy on the target task*

A part of the trajectory of a best-of-run policy is shown on figure 8. The robot goes from the switch to the ball and stops at the basket to release the ball on its way back to the switch. It repeats this cycle up to the end of the evaluation, thus collecting all the available balls, no matter its initial position. Only a part of the trajectory corresponds to a non null reward in the knowledge base (blue small circles on figure 8). It actually corresponds to some of the critical parts of the behavior:

1) the robot has activated the switch and starts going towards the ball;
2) it crosses the basket room to go to the ball room;
3) it robot approaches the ball;
4) it approaches the basket while carrying the ball;
5) it goes back to the switch after having released the ball.

## V. Discussion

The knowledge base used in a reward shaping approach has allowed to solve the target task while no policy generated without it did succeed. The results show anyway a large variability. It can be hypothesized that this variability is due to an incomplete or inaccurate knowledge base. The reward of each behavior segments of a best-of-run policy is high all along its trajectory (figure 5) while it may be expected that only some parts of it are actually salient and critical to solve the task. It means that the knowledge base may consider as salient behaviors that are actually not important to get the reward. While learning on the target task, the reward of such behavior segments may be misleading and may thus explain the low performance of some runs. Higher performances may then be expected with a knowledge base containing better approximations of behavior segments contribution. The knowledge base should for instance be updated in every generation while solving the source task, instead of every 200 generations

as it was done here. Likewise, the knowledge base from different runs can be merged to make the estimation of a behavior segment saliency more accurate. Such a knowledge base is actually interesting per se as it allows to identify what parts of the policies are the most significant. This knowledge may be used to better understand what happens, but also to progressively build a repertoire of discrete actions, if the part of a policy responsible for the behavior segment can be isolated. It would open the way towards the use of reinforcement learning algorithms with actions that are automatically built and thus significant from the point of view of the robot.

The question of the relation between the source and target task has not been addressed here. The knowledge base takes into account the reward on the task and tries to credit each behavior segment with its contribution. It can then be seen as a credit assignment approach. The source and target tasks must have the same reward function for the transfer to be of help.

## VI. Conclusion

This work introduces a transfer learning approach in robotics that deals with continuous actions and states. It relies on the population based feature of evolutionary algorithms to build a knowledge base in which the contribution of each observed behavior segment is progressively evaluated. The knowledge base is exploited when solving a target task with a reward shaping approach. The reward to be optimized by the direct policy search algorithm includes the maximum reward of the behavior segments out of the knowledge base, thus identifying salient behavior segments before they lead to a reward on the task. This approach allowed to solve a complex sequential task while exploiting the knowledge acquired on a simpler version of it. The analysis of the solution on the target task reveals that the salient behaviors actually make sense from a human expert point of view, thus showing that the knowledge base has captured at least some of the important features of the task.

## Acknowledgment

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

## References

[1] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the PPSN VI Conference*, pages 849–858, Paris, France, 2000. Springer. LNCS No. 1917.

[2] S. Doncieux and J.-B. Mouret. Behavioral diversity measures for Evolutionary Robotics. In *IEEE Congress on Evolutionary Computation, 2010 (CEC 2010)*, pages 1303–1310, 2010.

[3] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06*, New York, New York, USA, 2006. ACM Press.

[4] M. Grzes. *Improving exploration in reinforcement learning through domain knowledge and parameter analysis*. PhD thesis, Univ. of York, 2010.

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

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

[7] J.-B. Mouret and S. Doncieux. Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In *IEEE Congress on Evolutionary Computation, 2009 (CEC 2009)*, pages 1161–1168, 2009.

[8] J.-B. Mouret and S. Doncieux. Using Behavioral Exploration Objectives to Solve Deceptive Problems in Neuro-evolution. In *GECCO'09: Proceedings of the 11th annual conference on Genetic and evolutionary computation*, pages 627–634. ACM, 2009.

[9] J.-B. Mouret and S. Doncieux. Sferesv2: Evolvin' in the Multi-Core World. In *IEEE Congress on Evolutionary Computation, 2010 (CEC 2010)*, pages 4079—-4086, 2010.

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

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

[12] C. Ollion and S. Doncieux. 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, 2011.

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

[14] J. Randlov and P. Alstrom. Learning to Drive a Bicycle using Reinforcement Learning and Shaping. *ICML '98 Proceedings of the 15th international conference on Machine learning*, pages 463—-471, 1998.

[15] M. Snel and S. Whiteson. Multi-task evolutionary shaping without pre-specified representations. *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, pages 1031—-1038, 2010.

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

[17] M. E. Taylor, P. Stone, and Y. Liu. Value Functions for RL-Based Behavior Transfer: A Comparative Study. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

[18] S. Thrun and T. M. Mitchell. Lifelong Robot Learning. *Robotics and Autonomous Systems*, 15:25–46, 1995.

[19] L. Torrey and J. Shavlik. Transfer Learning. In *Handbook of Research on Machine Learning Applications*, pages 1–22. 2009.

## Appendix

Statistical significance of the results on the target task (Mann-Whitney test) at generation 3000

| | no transfer | sum rand | sum | mo rand | mo |
|---|---|---|---|---|---|
| no transfer | 1 | 0.3026 | 0.01088 | 0.3423 | 0.005414 |
| sum rand | 0.3026 | 1 | 0.005338 | 0.4845 | 0.002507 |
| sum | 0.01088 | 0.005338 | 1 | 0.007448 | 0.4091 |
| mo rand | 0.3423 | 0.4845 | 0.007448 | 1 | 0.003204 |
| mo | 0.005414 | 0.002507 | 0.4091 | 0.003204 | 1 |

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 $\varphi(x) = \frac{1}{1+\exp(b-x)}$