Autonomous Shaping via Coevolutionary Selection of Training Experience

Marcin Szubert and Krzysztof Krawiec

Institute of Computing Science, Poznan University of Technology, Poznań, Poland
{mszubert,kkrawiec}@cs.put.poznan.pl

Abstract. To acquire expert skills in a sequential decision making domain that is too vast to be explored thoroughly, an intelligent agent has to be capable of inducing crucial knowledge from the most representative parts of it. One way to shape the learning process and guide the learner in the right direction is effective selection of such parts that provide the best training experience. To realize this concept, we propose a shaping method that orchestrates the training by iteratively exposing the learner to subproblems generated autonomously from the original problem. The main novelty of the proposed approach consists in equalling the learning process with the search in subproblem space and in employing a coevolutionary algorithm to perform this search. Each individual in the population encodes a sequence of subproblems that is evaluated by confronting the learner trained on it with other learners shaped in this way by particular individuals. When applied to the game of Othello, temporal difference learning on the best found subproblem sequence yields substantially better players than learning on the entire problem at once.

Keywords: reinforcement learning, coevolutionary algorithms, shaping.

1 Introduction

Many real-world problems concern sequential decision making where every single decision changes the state of the environment and results in a *reward*. The main difficulties with handling such problems arise from the fact that rewards can be delayed in time. As a result, acting greedily is not always the best strategy and, even more importantly, it is hard to determine which actions should be credited with future rewards. Training an autonomous agent to maximize the cumulative payoff in this kind of problems is formalized as reinforcement learning (RL) [1]. This machine learning paradigm encapsulates the nature-related concept of *trial-and-error* search for optimal behavior, guided by the *interactions* between a learner and an unknown environment.

Past research shows that the most difficult RL problems are those with a long sequence of unrewarded decisions leading to a single payoff at the end. Typical examples of such scenarios are board games, where the only explicit reward is the final game outcome. One way to aid the learning process in this case is to use the idea of *shaping,* borrowed from behavioral psychology [2]. It assumes that a

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 215–224, 2012.

learner is trained on a series of easier problems before approaching the original one. The main difficulty with shaping is that it requires very careful selection of training problems that should possibly approximate the desired behavior [3]. Such expert-driven shaping involves substantial amount of domain knowledge, and can introduce unnecessary biases into the learning process. In this context, learning from scratch remains an unbiased and thus attractive alternative.

In this paper we propose a method for autonomous shaping without giving up the above *tabula rasa* attitude. We employ competitive coevolution [4] to identify appropriate training experience for an agent that learns a game playing strategy. This leads to mapping the original problem of optimizing an agent's policy into a *dual* problem of finding the best input for the policy learning algorithm, while preserving the ultimate goal of learning — maximization of an adopted quality measure. The critical question one needs to answer to implement this form of shaping is: where can we get the simpler training problems from? In the case of games, *endgames* are the most obvious form of subproblems, as they naturally include the final rewards, which are essential to do any learning at all. Assuming that the training experience is gathered dynamically, starting from a given initial game state, our idea is to change this initial state in such way that the following interactions allow for faster and more general learning. More specifically, we consider *sequences of endgames*, represent them as *shaping vectors*, and search for the shaping vector that provides the best possible learning gradient.

We expect that learning from the pre-selected experience will converge faster and improve the final performance of the trained agents. Additionally, the dual problem definition can bring even more benefits. Firstly, the selected set of subproblems is a valuable source of knowledge about the problem structure. Indeed, shaping vector can be considered as an analog to the concept of *underlying objectives* of the problem [5], which here can be interpreted as the crucial set of skills needed for successfully operating in the given environment. Secondly, diversification of learning experience is a natural answer to the *exploration-exploitation* trade-off. Performing random moves to explore the environment (for instance, according to the so-called ε-greedy action selection scheme) could no longer be needed if the shaping vector is diverse enough.

2 Shaping by Initial State Selection

We consider sequential decision problems, which are defined by a state space S, a set of possible actions A, a default initial state $s_o \in S$, and a subset of terminal states. Additionally, the environment specifies a reward function $r : S \times A \to \mathbb{R}$ and a transition function $f : S \times A \to S$, which can be non-deterministic. The objective is to automate the process of learning agents that solve such problems, i.e., maximize the expected reward. An agent's behavior is determined by its policy $\pi : S \to A$, $\pi \in \Pi$ that for each state chooses an action leading to one of the subsequent states. The set of states traversed by an agent in a single episode is a directed path from s_0 to one of the terminal states in the transition graph that spans S. Such paths form samples of experience that can be used for improving the policy.

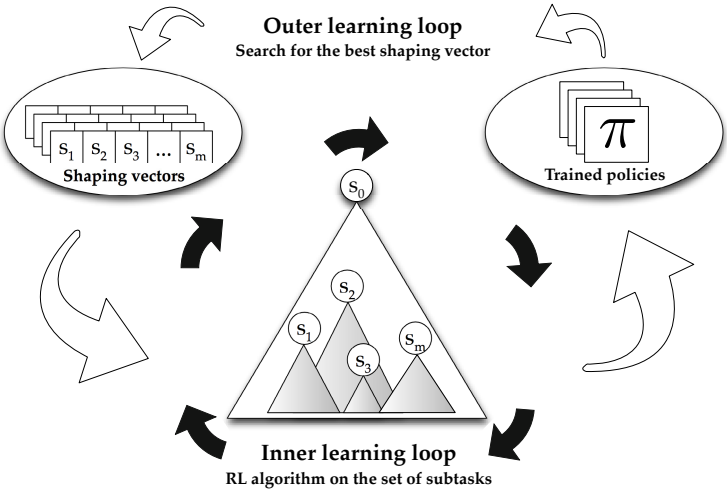


Fig. 1. The inner and outer loops of the shaping by initial state selection

We assume that an incremental learning algorithm $T : \Pi \times S \to \Pi$ is given that, provided with a current policy π_k and an initial state $s \in S$, produces an improved policy π_{k+1}. In the online variant considered here, learning occurs during exploration of the state graph: a training episode $T(s, \pi_k)$ consists in a simulation of agent's traversal through S, starting in s, with a single learning step taking place after each state transition.

It is usually assumed, particularly in the domain of board games, that the training process starts from the default initial state, i.e., T is always applied to s_0. This seems obvious, as, in the end, we want to learn a policy capable of solving the *entire* problem (e.g. playing the full game). However, for many problems the number of states that can be reached in the initial steps of problem solving is low, and grows exponentially with subsequent steps. As a result, a learner that starts from s_0 is doomed to overexplore the initial stages of problem solving while underexploring the final ones.

The main tenet of the proposed approach is that training a policy on a well-assorted, properly diversified and representative set of subproblems can be more beneficial than confronting it with the entire problem. We implement the concept of a set of subproblems by defining *shaping vector*, which is simply a vector $\mathbf{s}$ of m states $s_i \in S$, $i = \{1, \ldots, m\}$, where, in accordance with the sequential nature of considered problems, every $s_i \neq s_0$ identifies a subproblem of problem s_0 (assuming the transition graph is acyclic). A shaping vector can represent the training experience from potentially different areas of the state graph.

We orchestrate the learning process by iteratively applying the learning algorithm to consecutive elements of shaping vector: $\pi_i \leftarrow T(s_i, \pi_{i-1})$, where π_0 is an initial policy created in some arbitrary way. In this way, the experience gathered in π_i while solving subproblem s_i can be preserved when learning from subsequent subproblems. This inner learning loop (see Fig. 1) can iterate over the

elements of experience multiple times, if needed. Ultimately, the obtained strategy $\pi_{\mathbf{s}}$ is expected to embody the knowledge derived form the set of subproblems embedded in the shaping vector $\mathbf{s}$.

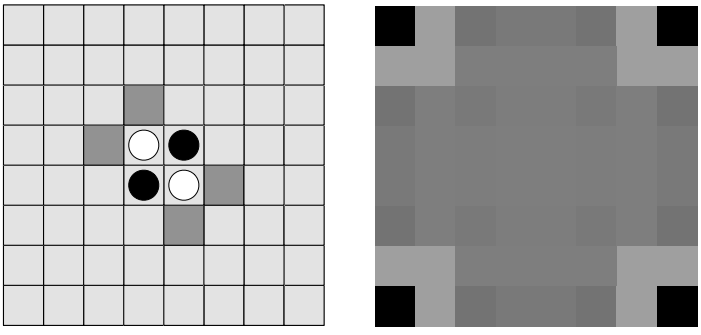
The choice of subproblems to form the training experience is essential for the performance of the trained policy. For instance, the particular assortment of subproblems can make it impossible for the learner to visit certain states in S during learning. In absence of objective guidelines that would help making this choice, we delegate this task to evolutionary algorithm, which maintains a population of individuals, each defining a training experience. Shaping vector $\mathbf{s}$ forms then the *genotype* of an individual, while its *phenotype* is the policy $\pi_{\mathbf{s}}$ trained using the above learning procedure. Evaluation of the phenotype formed in this way consists in running $\pi_{\mathbf{s}}$ on the entire problem, starting from s_0, possibly multiple times if indeterminism is involved. The fitness of individual can be then defined as, e.g., the average reward obtained by $\pi_{\mathbf{s}}$. In this way, the evolutionary process becomes responsible for searching for the useful training experiences, forming the outer loop of the proposed approach (Fig. 1).

In this paper we apply the above recipe for autonomous shaping to *competitive environments*, in which solving sequential decision problems boils down to playing games, and subproblems correspond to endgames. Rather than maximizing the expected reward on a single problem in a static, single-agent environment, we want to maximize the expected game outcome when playing against *any* opponent, i.e., another agent that interferes at the decision making process. This objective can be naturally implemented using coevolutionary algorithms, in which the fitness of an individual depends on the outcomes of its interactions with the other individuals in the population. Technically, we implement single-population competitive coevolution [4]: in the evaluation phase, the strategies $\pi_{\mathbf{s}}$ derived from particular individuals play a round-robin tournament against each other, and the total score received determines individual's fitness.

Independently of the choice of the algorithm performing the outer learning loop, the proposed approach can be then considered dual with respect to traditional methods of policy learning. Rather than aiming at acquisition of maximum knowledge from the original problem by, e.g., tuning the parameters of the training algorithm, the focus of the method is on shaping, i.e., exposing the learner to the 'right' training experience represented by selected subproblems. In short, *what* to learn becomes here more important than *how* to learn. In this context, the choice of the actual training algorithm T is of secondary importance: its parameters, if any, remain fixed during the entire training process, and it only serves as a means to assess the usefulness of particular set of initial states.

3 Experimental Setup

In the following we apply the proposed approach of autonomous shaping in its coevolutionary variant to the problem of learning to play the board game of Othello (Fig. 2a). The experiments have been conducted using our coevolutionary algorithms library cECJ [6] built upon the Evolutionary Computation in Java framework. For each considered setup, evolutionary runs have been repeated 20 times.



(a) Othello initial board state (b) Heuristic WPC weights

Fig. 2. Othello board and its coloring according to heuristic player weights (darker color — greater weight)

Learner Architecture. One of the main issues to consider when learning game-playing strategy is the architecture of the learner, which is mainly determined by a strategy representation. Of many possible ways in which the strategies $\pi \in \Pi$ could be represented, we chose the simple weighted piece counter (WPC). WPC assigns a weight w_i to each board location i and uses scalar product to calculate the utility f of a board state $\mathbf{b}$:

$$f(\mathbf{b}) = \sum_{i=1}^{8\times 8} w_i b_i, \tag{1}$$

where b_i is +1, -1, or 0 if, respectively, location i is occupied by a black piece, white piece, or empty. The players interpret the values of f in a complementary manner: the black player prefers moves leading to states with larger values, while smaller values are favored by the white player. Alternatively, WPC may be viewed as an artificial neural network comprising a single linear neuron with inputs connected to board locations. The standard heuristic player represented as a WPC is illustrated in the Fig. 2b. We use it also as an opponent in our experiments to measure the post-training performance of agents.

The Inner Learning Algorithm. We used the basic temporal difference method TD(0) as the learning algorithm T that improves a game-playing strategy on the basis of the training experience represented as a shaping vector (cf. Section 2). The initial strategy π_0 has all weights zeroed (see Eq. (1)). Given a state s_i from the shaping vector $\mathbf{s}$, invoking $T(s_i, \pi_k)$ consists in a single training pass of self-play TD(0) with s_i as an initial state and π_k determining the initial values of WPC weights. Game outcomes determine the rewards. The weights of WPC are modified after every move by a gradient-descent temporal difference update rule [7] with the learning rate parameter set to $\alpha = 0.01$. Each element of the shaping vector was used as an initial state for 100 learning episodes to increase the amount of experience gathered in the corresponding part of the game

tree. TD(0) was previously applied for Othello [8], proving capable of producing very good players in short training times.

The Outer Learning Algorithm. The outer learning algorithm, which targets on optimizing the training experience used by the inner learning phase, is framed as coevolutionary learning. The initial population comprises 50 shaping vectors, each composed of $m = 50$ states selected randomly from games played between two random players. The subsequent generations are bred by crossover followed by mutation. The former operator is uniform and homologous, so an offspring inherits $m/2$ randomly selected states from the first parent and the rest from the second one, and the order of states is preserved. Mutation is applied to the offspring with probability 0.05 per state and consists in replacing a state with a newly generated random state. The genotype-phenotype mapping is realized by the inner learning loop, and the evaluation consists of playing a population-wide round-robin tournament between strategies created in this way. The players score $3, 1$, or 0 points for winning, drawing, and losing, respectively. The total score earned in the tournament becomes individual's fitness, which is then subject to tournament selection of size 5. Thus, we evaluate shaping vectors by judging the performance of players created with their guidance.

4 The Results

The complete process of learning a game strategy using autonomous shaping involves two phases. First, the method proposed in Section 2 attempts to evolve the best shaping vector for the given learning algorithm. In the second phase, this vector is employed to train a strategy, which becomes the final outcome of the overall training process. All players in our experiments are deterministic, as well as the game of Othello itself. Thus, in order to estimate the score of a given trained player against the WPC-heuristic (Fig. 2b), we forced both players to make random moves with probability $\varepsilon = 0.1$. This provides richer repertoire of players' behaviors and makes the resulting estimates more continuous and robust.

Phase 1: Search for the Best Shaping Vector. The objective progress of this procedure was monitored by assessing the quality of the fittest player, i.e., the player that appeared the best among all the players trained with particular shaping vectors. We call this player the *best-of-generation learner*.

Figure 3 illustrates the performance of the best-of-generation learners, averaged over 20 coevolutionary runs. For reference, we plot also best-of-generation players found by standard coevolutionary search performed directly in the space of WPC strategies (for more details see [8]). Clearly, coevolution of training experience outperforms the direct approach. The level of play it attains is very similar to the best strategies obtained using CTDL, a hybrid of coevolution and TDL proposed in our previous work [8].

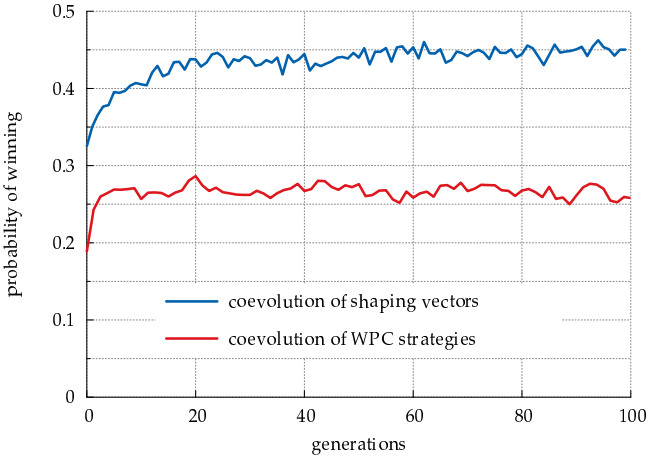


Fig. 3. Comparison of the average performance of the *best-of-generation learners* shaped by coevolved training experience and the *best-of-generation players* coevolved directly, against the WPC-heuristic opponent

Phase 2: Training the Strategy Using the Best Shaping Vector Found. In this phase, the best shaping vector found in phase 1 is mapped to a strategy using the genotype-phenotype mapping described in Section 2. We take a deeper look at this inner learning process realized by TD(0) algorithm.

Figure 4 visualizes the learning from the training experience embodied by the best shaping vector. Every thin blue curve depicts the mean performance of a strategy trained using the best shaping vector found in one of 20 evolutionary runs. The horizontal axis corresponds to the inner learning loop shown in Fig. 1 (as opposed to Fig. 3, where it marked the iterations of the outer loop). Each learning episode corresponds to an application of the training algorithm (TD(0)) to a single initial state, $T(s_i, \pi_k)$, so the horizontal axis is simply the k axis.

The thick red line shown in Fig. 4 depicts the behavior of the the standard TDL learning process, starting always from s_0 (illustrated in Fig. 2a), which gathers experience by ε-greedy action selection scheme (with ε equal to 0.1). Standard TDL clearly stalls much earlier than the shaping approach, and attains substantially worse performance at the end of training.

Performance against the WPC-heuristic says only a little about the overall objective quality of a strategy, because in practice we typically aim at producing versatile and robust players, capable of winning against a wide range of opponents. Thus, we gauged also the *relative performance* against other players trained using different methods. To this aim, we confront the teams of best-of-run strategies obtained from 20 runs with the team of players that have been trained on full games, using TD(0) randomized self-play starting from the default initial state s_0. Table 1 presents the outcomes of that duel, with the shaping-trained strategies sorted descendingly with respect to their outcome. The teams of strategies produced using the proposed approach are clearly superior. Even the worst

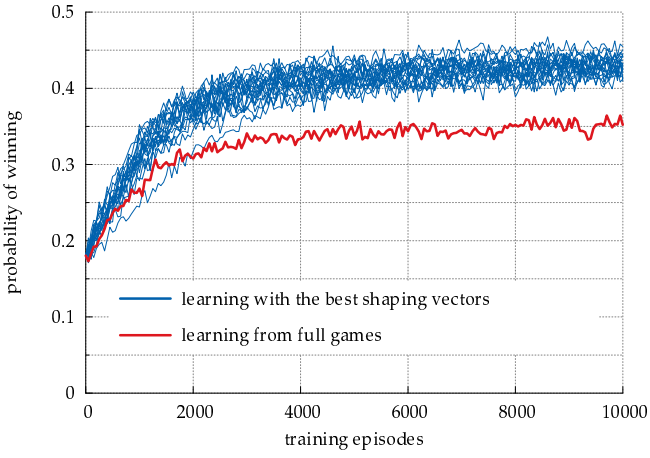


Fig. 4. The average performance of the learners trained with the best shaping vectors (blue, one plot per vector) vs. the average performance of the learners trained from full games (default experience, red), as a function of the number of TD(0) training episodes

Table 1. The outcomes of matches played between the teams of players obtained using the shaping approach with a team of strategies trained using randomized self-play TD(0)

Run #	*Wins*	*Draws*	*Losses*	*Points*	*% pts.*
1	508	20	272	1544	64.33
2	497	36	267	1527	63.63
3	497	23	280	1514	63.08
4	495	22	283	1507	62.79
5	492	20	288	1496	62.33
6	488	28	284	1492	62.17
7	486	26	288	1484	61.83
8	482	32	286	1478	61.58
9	478	35	287	1469	61.21
10	476	27	297	1455	60.63

Run #	*Wins*	*Draws*	*Losses*	*Points*	*% pts.*
11	476	22	302	1450	60.42
12	472	33	295	1449	60.38
13	475	23	302	1448	60.33
14	476	15	309	1443	60.13
15	472	26	302	1442	60.09
16	474	17	309	1439	59.96
17	465	28	307	1423	59.29
18	464	24	312	1416	59.00
19	453	37	310	1396	58.17
20	457	24	319	1395	58.13

of them wins substantially more games than it loses. Also, the performance of particular teams varies only slightly, with most of them scoring between 59 and 62% of all available points. This clearly suggests that the search for initial state vectors, though intermediated by the nontrivial genotype-phenotype mapping, repeatedly leads to producing stable and well-performing players.

5 Discussion and Related Work

The problem of selecting the training experience for a reinforcement learning agent has been addressed by several authors. Mihalkova and Mooney [9] propose a method for improving the reinforcement learning by allowing the learner to

relocate, i.e., to be placed in a requested state of the environment. An agent may benefit from this possibility by omitting the already known regions of the environment (when the agent is "bored" and is not learning anything new) or escaping from the parts of the state space that are unlikely to be visited again using the optimal policy (when the agent made a wrong exploratory move and fell "in trouble"). Relocation destinations are chosen according to an uncertainty measure reflecting agent's confidence about the best action in a given state. This approach differs from ours in being inherently *active* — it is the learner who makes decisions about when and where to relocate within an online training process. In this context, our method resembles more the selective sampling used in traditional supervised active learning [10].

A complementary *passive* approach is taken by Rachelson *et al.* [11] who introduce the meta-algorithm of Optimal Sample Selection (OSS). Given a batch-mode RL policy inference algorithm, a policy evaluation method, and a generative model of the environment, OSS attempts to identify a set of one-step transitions which, when supplied to the policy learning algorithm, lead to an optimal behavior with respect to the evaluation measure. In this method the learning proceeds independently from the selection of training experience — the learner cannot affect the way the experience is gathered. Also, the policy learning algorithm is assumed to work in an offline batch manner, i.e., it exploits a fixed, prepared in advance set of training examples (sample of transitions), without a need of dynamically interacting with the environment. Our approach abstracts from the character of the policy learning algorithm and is more coarse-grained — instead of selecting single transitions, we find entire states that implicitly identify many useful paths through the environment. This allows us to represent the training experience in a more compact, illustrative, and generic way.

Finally, changing the initial state can be seen as a slight modification of the learning task itself. What we finally want to achieve is then the transfer of knowledge from a set of adjusted task to the original problem. We expect that it can improve the learning process in the same way as the transfer learning [12], including initial performance, time of learning and final performance. In a similar spirit, Konidaris and Barto [13] investigate knowledge transfer across a sequence of tasks and employ an autonomous shaping approach by augmenting reward functions – they use the knowledge about predicted rewards from one task to shape the reward function of the other one.

6 Summary

In learning game-playing strategies, it is typically the role of a *trainer* to guide the learner through the paths of the game tree from which it can learn the most. This study revolved around the observation that such guidance can take on different forms. Naturally, the trainer is embodied by an opponent, e.g., an expert player or the learner itself [14]. In the proposed method, the opponent strategy, though varying with time, is stationary in being produced by a fixed learning algorithm, while the role of guidance is delegated to a set of initial

states that limit the exploration to the corresponding partial game trees. We are not looking for an ideal trainer here, but for an ideal training experience. Eventually, the goal is to shape the learning process so that it produces proficient learners prepared to perform well in every, potentially unseen before, region of environment. This goal has been attained in this study for the game of Othello: rephrasing a learning task in a way that enables autonomous shaping led to better performing and more versatile players. Applicability of this approach to other interactive and non-interactive domains is to be verified in future research.

Acknowledgment. This work has been supported by grant no. N N519 441939.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press (1998)
2. Skinner, B.: The behavior of organisms: An experimental analysis. Appleton-Century (1938)
3. Randløv, J., Alstrøm, P.: Learning to drive a bicycle using reinforcement learning and shaping. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 463–471. Morgan Kaufmann, San Francisco (1998)
4. Popovici, E., Bucci, A., Wiegand, R.P., de Jong, E.D.: Coevolutionary principles. In: Handbook of Natural Computing. Springer, Berlin (2010)
5. Jaśkowski, W., Krawiec, K.: Formal analysis, hardness and algorithms for extracting internal structure of test-based problems. Evolutionary Computation 19(4), 639–671 (2011)
6. Szubert, M.: cECJ — Coevolutionary Computation in Java (2010), `http://www.cs.put.poznan.pl/mszubert/projects/cecj.html`
7. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning 3, 9–44 (1988)
8. Szubert, M., Jaśkowski, W., Krawiec, K.: Coevolutionary Temporal Difference Learning for Othello. In: 2009 IEEE Symposium on Computational Intelligence and Games, pp. 104–111 (2009)
9. Mihalkova, L., Mooney, R.: Using active relocation to aid reinforcement learning. In: Proceedings of the 19th International FLAIRS Conference, pp. 580–585 (2006)
10. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning 15(2), 201–221 (1994)
11. Rachelson, E., Schnitzler, F., Wehenkel, L., Ernst, D.: Optimal sample selection for batch-mode reinforcement learning. In: Proceedings of the 3rd International Conference on Agents and Artificial Intelligence, ICAART 2011 (2011)
12. Torrey, L., Shavlik, J.: Transfer Learning. In: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, pp. 242–264. IGI Global (2009)
13. Konidaris, G., Barto, A.: Autonomous shaping: Knowledge transfer in reinforcement learning. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 489–496. ACM (2006)
14. Epstein, S.: Toward an ideal trainer. Machine Learning 15(3), 251–277 (1994)