Neurocomputing 263 (2017) 48–59

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Multi-objectivization and ensembles of shapings in reinforcement learning

Tim Brys[a,*], Anna Harutyunyan[a], Peter Vrancx[a], Ann Nowé[a], Matthew E. Taylor[b]

[a] Vrije Universiteit Brussel, 1050 Brussels, Belgium
[b] Washington State University, Pullman, WA 99164, United States

ARTICLE INFO

Article history:
Received 9 February 2016
Revised 30 May 2016
Accepted 1 February 2017
Available online 16 June 2017

Keywords:
Reinforcement learning
Multi-objectivization
Ensemble techniques
Reward shaping

ABSTRACT

Ensemble techniques are a powerful approach to creating better decision makers in machine learning. Multiple decision makers are trained to solve a given task, grouped in an ensemble, and their decisions are aggregated. The ensemble derives its power from the diversity of its components, as the assumption is that they make mistakes on different inputs, and that the majority is more likely to be correct than any individual component. Diversity usually comes from the different algorithms employed by the decision makers, or the different inputs used to train the decision makers.

We advocate a third way to achieve this diversity, called diversity of evaluation, using the principle of *multi-objectivization*. This is the process of taking a single-objective problem and transforming it into a multi-objective problem in order to solve the original problem faster and/or better. This is either done through decomposition of the original objective, or the addition of extra objectives, typically based on some (heuristic) domain knowledge. This process basically creates a diverse set of feedback signals for what is underneath still a single-objective problem. In the context of ensemble techniques, these various ways to evaluate a (solution to a) problem allow different components of the ensemble to look at the problem in different ways, generating the necessary diversity for the ensemble.

In this paper, we argue for the combination of multi-objectivization and ensemble techniques as a powerful tool to boost solving performance in reinforcement learning. We inject various pieces of heuristic information through reward shaping, creating several distinct enriched reward signals, which can strategically be combined using ensemble techniques to reduce sample complexity. We provide theoretical guarantees and demonstrate the potential of the approach with a range of experiments.

1. Introduction

When making decisions, especially decisions with a potentially high impact, it is considered good practice to solicit the advice of one or more third parties [1]. The idea is that when many parties agree, one can be more confident about the accuracy of the proposed decision. Conversely, if there is disagreement, the decision maker can at least take into account the fact that it is possible he does not know the best response, and take precautionary measures. This strategy is not only frequently used by humans, but is also implemented in automated decision makers. In the latter case, such approaches are most commonly referred to as ensemble techniques [2]. A set of decision makers is combined into an ensemble (French for 'together'), and their advice or proposed decisions are aggregated into an actual decision using a specific strategy, such as majority voting. Ensembles derive their power from the diversity that is present in the set of their decision makers [3], which allows the composite decision maker to outperform any single one of its components (clearly it would be pointless to combine many identical predictors). This diversity can originate from many sources: the decision makers can use different decision making algorithms, they could have been trained on different inputs, be differently parameterized, etc. The reasoning is that due to their diversity, the different decision makers will make their mistakes on different inputs, and thus that the strategic combination of their advice will lead to a reduction of mistakes.

In this paper, we argue for an as yet little researched way of generating this diversity. Instead of diversifying algorithm parameters or the training inputs fed into the algorithms, we create a diverse set of utility functions based on the original one. This process allows for the exploitation of structure found in the original utility function or the inclusion of domain knowledge, and yields

* Corresponding author.
E-mail address: timbrys@vub.ac.be (T. Brys).

http://dx.doi.org/10.1016/j.neucom.2017.02.096
0925-2312/

several distinct ways to evaluate decisions for the ensemble to use. Although the idea of using diversity of evaluation in an ensemble could have general applicability, we develop it in the context of reinforcement learning, which is specifically amenable to the idea due to the distinct way an agent evaluates its behavior (the accumulation of stochastic, delayed rewards over time, as opposed to an instantaneous deterministic objective function evaluation).

We first briefly discuss ensemble techniques and multi-objectivization, and then formally introduce reinforcement learning. Following that, we look how the former ideas can be combined and integrated into reinforcement learning. We discuss and prove some theoretical properties of this approach, and experimentally show for two reinforcement learning problems that a significant boost in performance can be obtained.

2. Ensemble techniques

As discussed before, a successful ensemble [2] is built by providing two components: (1) a set of *diverse* decision makers, and (2) a strategy for the combination of their outputs such that the number of mistakes made is minimized. The choice of how to diversify the decision makers is crucial, as it is desirable that different decision makers are experts on different (overlapping) parts of the input space,[1] or put another way, that different decision makers make their mistakes on different inputs. We discuss two common ways of creating this required diversity: diversity of *input* and *algorithm*.

A lot of research on ensemble techniques has focused on classification and regression problems [2]. Most successful techniques in those domains generate diversity by feeding different decision makers with different (possibly overlapping) subsets of the training data, i.e., *diversity of input*. Bagging [4] is the most naive approach in this category, as it trains different classifiers on randomly sampled subsets of the training data, aggregating their output by majority voting. Boosting [5] takes this idea one step further, by intelligently selecting the subset of training data attributed to each classifier. It trains a first classifier with a random subset of the training data. Then, a second classifier is trained on data only half of which the first classifier can classify correctly. A third classifier is then trained on instances which the first two disagree on. This process can be applied recursively to build a strong ensemble. AdaBoost [6] is an extremely popular and successful generalization of boosting, improving on boosting by including a more refined probabilistic training data sampling procedure, and weighted majority voting, based on classifiers' success during the training phase.

Diversity of algorithm refers to the actual algorithms used by decision makers being different, either inherently, or through different parameterization. Hansen and Salamon [7] note that different neural network instances, trained on the full set of training data (in stark contrast to the 'diversity of input' approaches), will still be diverse due to different initializations of their weights and different sequencing of the training data. These networks end up in different local optima of the weights-space and therefore have different accuracy in different parts of the input space. In differential evolution literature, Mallipeddi et al. [8] use ensembles of mutation strategies and parameters, positing that "different optimization problems require different mutation strategies with different parameter values" and "different mutation strategies with different parameter values may be better during different stages of the evolution." Their ensembles compared favorably with the state-of-the art in differential evolution. Wiering and van Hasselt [9] combine a variety of reinforcement learning algorithms with significantly different properties, showing how their ensemble requires less learning experiences to achieve equal or better performance than the single best algorithm.

These two ways to generate ensemble diversity have been researched extensively in different research areas and have yielded several powerful techniques. In this paper, we argue for a third approach based on *diversity of evaluation*. In the following section, we describe *multi-objectivization*, the process we will use to achieve this diversity.

3. Multi-objectivization

Multi-objectivization [10] is the process of creating a variety of utility functions for a task, starting from a single one.[2] Formally, multi-objectivization takes single-objective problem p with utility function $u : X \to \mathbb{R}$, and outputs multi-objective problem $\hat{p}$ with utility function $\mathbf{u} = \{u_1, u_2, \ldots, u_n\} : X \to \mathbb{R}^n$, with $n > 1$. The idea is that $\hat{p}$ should be constructed in such a way that it is easier to solve than p. There are basically two approaches to create a diverse set of utility functions or 'objectives',[3] starting from a single-objective problem: either through *decomposition* of the original single objective, or through *addition* of extra objectives.

A prime example of multi-objectivization through *decomposition* is found in decision tree literature, where instead of using the total classification error to guide tree generation, tree performance is measured using the per-class misclassification, and trees are optimized using a multi-objective algorithm, which has been shown to lead to better trees in some cases [11]. Another good example is the transformation of constrained optimization problems[4] into unconstrained ones, with the original objective being decomposed into an unconstrained objective, and several other objectives encoding the constraints [12–14]. For example, Coello [12] gives empirical evidence that this form of constraint relaxation can outperform more standard penalty stratagems in Evolutionary Algorithms. This idea was also studied in reinforcement learning, where Raicevic assigned different reward signals to the sub-tasks resulting from a task decomposition [15].

Multi-objectivization through the *addition* of objectives typically involves the incorporation of some heuristic information or expert knowledge on the problem. Jensen [16] was one of the first to use what he calls 'helper' objectives next to the primary one. He investigated the job-shop scheduling and traveling salesman problems and found that additional objectives based on, respectively, time and distance-related intuitions help solve these problems faster. In genetic programming, where the goal is to evolve an accurate program to solve specific tasks, Bleuler et al. [17] and de Jong et al. [18] found that adopting program size as a second objective resulted in smaller *and* more accurate programs being evolved. This helper objective is a straightforward implementation of Occam's razor, and solves the common genetic programming problem of 'bloat', i.e., the tendency to evolve large overly-complex programs. The same principle of encoding minimization of model complexity for better generalization as an extra objective was succesfully used in decision tree research by Kim [19].

In most of these examples, multi-objectivization is beneficial because the true utility function is not available; only

[1] Input is being used here as referring to such concepts as input features in classification, state in reinforcement learning, candidate solution in evolutionary computation, etc.

[2] In this paper, we are only concerned with problems that are originally single-objective. Actual multi-objective problems may also benefit from the approach described here. But, since optimality is defined differently in multi-objective problems, it is unclear whether applying multi-objectivization will have unwanted effects.

[3] Both utility function and objective will be used interchangeably throughout this paper, and refer to the same concept.

[4] Constrained optimization problems are problems where one needs to optimize a given objective function with respect to a set of variables, given constraints on the values of these variables.

approximations are. Consider the case of learning a decision tree to classify some data. We can not evaluate the true utility (accuracy) of a given decision tree, because that would involve testing it on all possibly relevant data. Therefore, one uses an approximation of this true utility, by measuring the tree's accuracy on the available training data, a subset of all possible data. Thus, it can be helpful to use heuristics, such as minimizing tree size, as further guidance that may improve the approximation of the true utility (or rather, create an order over decision trees using multiple objectives that is more like the true order over decision trees).

In temporal difference reinforcement learning, the true utility of an action is the expected, discounted, accumulated reward to be gained by taking an action and then following some behavior. Since this utility is never directly given, but must be approximated based on observed rewards, reinforcement learning could benefit from the principle of multi-objectivization.

4. Reinforcement learning

In a control setting, Reinforcement learning (RL) [20] is a paradigm that allows an agent to optimize its behavior while operating in a given environment. The agent is rewarded or punished for the behavior it exhibits, and its aim is to maximize the accumulated reward over time, which by definition amounts to solving the task. More formally, the environment is defined as a Markov Decision Process (MDP) $\langle S, A, T, \gamma, R\rangle$. $S = \{s_1, s_2, \ldots\}$ is the set of states the environment can be in, and $A = \{a_1, a_2, \ldots\}$ is the set of actions the learning agent can execute. Executing action a when the environment is in state s makes it transition to state s' with probability $T(s'|s, a)$, yielding $R(s, a, s')$ as reward for that transition. Finally, γ, the discounting factor, defines how important future rewards are. The goal is to learn a policy π that maps states to actions in such a way that the expected discounted cumulative reward J^π, is maximized.

$$J^\pi \equiv E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]$$

Expectations are taken over start states s_0, rewards R, transition function T and policy π, with t representing timesteps.

Reinforcement learning algorithms either directly search the policy space to find a policy that maximizes the return, or estimate the expected returns and derive a policy from those. The learning algorithms used in this paper are of the second type, and more specifically temporal-difference (TD) learning algorithms. These estimate state (V) or state-action (Q) value functions that represent the return expected for following a given behavior policy. Algorithms such as Q-learning [21] incrementally update these estimates $\hat{Q}$ based on the rewards observed while the agent is interacting with the environment:

$$\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha\delta$$

with δ the temporal-difference error:

$$\delta = R(s, a, s') + \gamma \max_{a'} \hat{Q}(s', a') - Q(s, a)$$

If each action is executed in each state an infinite number of times on an infinite run and α is decayed appropriately, Q-learning is guaranteed to converge to the optimal values Q^* [22], from which the optimal policy π^* can be derived:

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

Many practical reinforcement learning problems have very large and/or continuous state spaces, making basic tabular learning methods impossible to use. A very popular way to overcome this problem is to use tile-coding function approximation [23], a linear approximator which overlays the state space with multiple axis-parallel tilings. This allows for a discretization of the state-space, while the overlapping tilings guarantee a certain degree of generalization. The Q-function can then be approximated by learning weights that map the tiles activated by the current state s to an estimated Q-value:

$$\hat{Q}(s, a) = \theta_a^T \phi_s$$

ϕ_s is the feature vector representing state s, i.e., a binary vector indicating the tiles activated by this state, and θ_a is the parameter vector that needs to be learned to approximate the actual Q-function.

4.1. Multi-objective reinforcement learning

Multi-objective reinforcement learning [24] (MORL) is a generalization of standard single-objective reinforcement learning, with the environment formulated as a multi-objective MDP, or MOMDP $\langle S, A, T, \gamma, \mathbf{R}\rangle$. The difference with the single-objective case is the reward function. Instead of returning a scalar value, it returns a vector of scalars, one for each of the m objectives:

$$\mathbf{R}(s, a, s') = (R_1(s, a, s'), \ldots, R_m(s, a, s'))$$

Policies are in this case evaluated by their expected vector returns $\mathbf{J}^\pi$:

$$\mathbf{J}^\pi \equiv \left[E\left[\sum_{t=0}^{\infty} \gamma^t R_1(s_t, a_t, s_{t+1})\right], \ldots, E\left[\sum_{t=0}^{\infty} \gamma^t R_m(s_t, a_t, s_{t+1})\right]\right]$$

Since there are multiple (possibly conflicting) signals to optimize, there is typically no total order over policies. Policies may be incomparable, i.e., the first is better on one objective while the second is better according to another objective, and thus the notion of optimality has to be redefined. A policy π_1 is said to strictly Pareto dominate another policy π_2, i.e., $\pi_1 \succ \pi_2$, if for each objective, π_1 performs at least as well as π_2, and it performs strictly better on at least one objective. The set of non-dominated policies is referred to as the *Pareto optimal set* or *Pareto front*. The goal in multi-objective reinforcement learning, and multi-objective optimization in general, is either to find a Pareto optimal solution, or to approximate the whole set of Pareto optimal solutions.

With a multi-objective variant of Q-learning, Q-values for each objective can be learned in parallel, stored as Q-vectors [25,26]:

$$\mathbf{Q}^*(s, a) = \left[Q_1^*(s, a), \ldots, Q_m^*(s, a)\right]$$

The most common approach to deriving a policy from these estimates is to calculate a linear scalarization, or weighted sum based on the estimated Q-vectors and a weight vector w [24,26,27]:

$$\pi^*(s) = \arg\max_a w^T \mathbf{Q}^*(s, a)$$

The weight vector determines which trade-off solutions are preferred, although setting these weights *a priori* to achieve a particular trade-off is hard and non-intuitive [28], often requiring significant amounts of parameter tuning.

4.2. Reward shaping

Many reinforcement learning algorithms take a *tabula rasa* approach, assuming no knowledge of the problem beyond knowing the sensors and actuators (states and actions). In complex domains, learning may therefore require unacceptably large amounts of experience, with initial performance being no better than random. A significant portion of the reinforcement learning literature describes ways to incorporate external knowledge about the task to

help the learning agent learn with less experiences, i.e., reduce its sample complexity [29–31]. One popular approach is reward shaping, which is a way to incorporate heuristic knowledge by providing the agent with extra reward through a shaping function $F(s, a, s')$, on top of the reward from the environment $R(s, a, s')$. The shaping function F is simply added to the reward signal:

$$R_F(s, a, s') = R(s, a, s') + F(s, a, s')$$

If F is implemented as the difference of some potential function Φ over the state space, and incorporates γ, the discount factor, as follows, then the shaping function is guaranteed to not alter the optimality of policies [29], and will direct the agent's exploration to areas of higher Φ:

$$F(s, a, s') = \gamma\Phi(s') - \Phi(s) \tag{1}$$

For example, if one wants to make an agent learn to walk to a goal location in a forest, one possible potential function $\Phi(s)$ to incorporate in a shaping could be the distance to the nearest tree, encouraging the agent to avoid bumping into trees.

Potential-based reward shaping has been successfully applied in such complex domains as RoboCup KeepAway soccer [32] and StarCraft [33], improving agent performance significantly. This framework has furthermore been extended (with corresponding proofs) to state-action-timestep tuples (s, a, t) [34], allowing for shapings that encourage not only the visitation of certain states, but directly encode the believed quality of actions, as well as allowing for shapings that change over time (e.g., encoding information that arrives during learning).

5. Multi-objectivization in reinforcement learning

In this section, we describe how multi-objectivization can be achieved in a reinforcement learning context. First, we describe CMOMDPs, a subclass of multi-objective MDPs that contain multi-objectivized MDPs. Then, we describe how an MDP can be transformed into a CMOMDP through multi-objectivization using reward shaping. In the section after this one, we will describe how ensemble techniques can be used to solve such a CMOMDP.

5.1. CMOMDP

Recall that multi-objective MDPs (MOMDPs) require the simultaneous optimization of multiple feedback signals. As conflicts may exist between objectives, there is in general a need to identify (a set of) trade-off policies. The set of optimal, i.e., non-dominated, incomparable policies is called the Pareto-front. Assuming all objectives are to be maximized, define the set $\mathcal{S}_p^*$ to contain all policies π of MOMDP p that are within ϵ_o of optimality for at least one objective o (with respect to the expected discounted cumulative reward $\mathbf{J}^\pi$):

$$\pi \in \mathcal{S}_p^* \iff \exists o \in \mathcal{O}_p, \forall \pi' \in \Pi_p : \mathbf{J}_o^\pi + \epsilon_o \geq \mathbf{J}_o^{\pi'}$$

$\epsilon_o \geq 0$ defines the largest difference in utility of objective o that the system designer is indifferent about, $\mathcal{O}_p$ is the set of objectives in p and Π_p is the set of possible policies for p. $\mathcal{S}_p^*$ will include at least the extrema of the Pareto-front of p.

We identify MOMDPs with correlated objectives (CMOMDP) as a specific sub-class of MOMDPs, defined to contain those MOMDPs p whose set $\mathcal{S}_p^*$ (and by extension whose Pareto-front) is so small that trade-offs between solutions on the Pareto-front are negligible, i.e. there are no noteworthy conflicts between the objectives. By consequence, the system designer does not care about which of the very similar optimal policies is found, but rather how fast it is found (and perhaps how well it is approximated). Formally:

$$p \in \text{CMOMDP} \iff \forall o \in \mathcal{O}_p : \max_{\pi \in \mathcal{S}_p^*}(\mathbf{J}_o^\pi) - \min_{\pi' \in \mathcal{S}_p^*}(\mathbf{J}_o^{\pi'}) \leq \epsilon_o$$

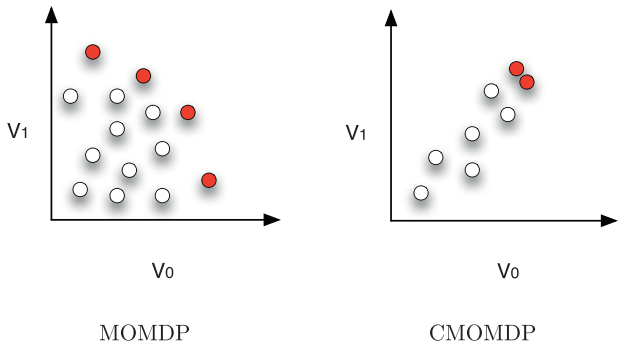


Fig. 1. Representative examples of an MOMDP and a CMOMDP. Dots represent policies, and red dots are optimal policies. In the CMOMDP, the optimal policies are so similar that the system designer/user does not care about which one is found, whereas in the MOMDP, it matters which trade-off is found. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

Thus, whether a multi-objective problem is contained in this class is to a certain extent subjective, depending on the system designer's preferences (ϵ_o). Such problems can be seen as providing multiple sources of information or feedback for the same basic single-objective problem, and intelligently combining such objectives may yield faster and better optimization. See Fig. 1 for a visual comparison between an MOMDP and a CMOMDP. An example of a CMOMDP is the traffic light control problem in our previous work [35]. Popular metrics to quantify agents' behavior in that setting are the average delay experienced by cars and the throughput of the system. In practice, optimizing either one of these results in optimizing the other, and combining these signals using a simple weighted sum was shown to result in faster learning.

5.2. Multi-objectivization

The relevance of this problem class may seem limited, as one does not encounter many such problems in the literature. But, we describe below how any single-objective MDP can be transformed (or multi-objectivized) into a CMOMDP using multiple reward shaping functions.

We reiterate that multi-objectivization [10] is the process of turning a single-objective problem into a multi-objective problem in order to improve solving of the single-objective problem. Our goal is to turn a single objective RL problem into a CMOMDP, i.e., a multi-objective problem without (significant) conflicts between the objectives, in order to better solve the single objective RL problem. We desire furthermore the property that finding an optimal policy in the CMOMDP equates to finding a (near-) optimal policy in the original single-objective problem.

MDPs can be multi-objectivized by copying the reward function multiple times, which results in a CMOMDP with a single Pareto-optimal point. Of course, this modification in itself can not improve learning (unless other sources of diversity are present). But, these copies of the basic reward signal can be diversified by enriching them with heuristic knowledge using the potential-based reward shaping paradigm. Since potential-based reward shaping is guaranteed to not alter the optimality of solutions [29], adding a different potential-based reward shaping to each of the copies of the reward signals keeps the problem a CMOMDP with a single Pareto optimal point. Yet, each of the different shaping functions provides a different evaluation of the behavior during the learning process.

More formally: to turn MDP M into CMOMDP M′ using m reward shaping functions F_i, the reward vector $\mathbf{R}$ of M′ is constructed as follows:

$$\mathbf{R}(s,a,s') = (R(s,a,s') + F_1(s,a,s'), \ldots, R(s,a,s') + F_m(s,a,s')) \quad (2)$$

where R is the reward function of M. Thus, we copy the base reward of M m times, and add a different shaping function to each.

We prove that this formulation preserves the total ordering, and thus also the optimality, of policies between M and M′, provided the shapings are potential-based. That is, that multi-objectivization by reward shaping does not introduce conflicts.

Theorem 1. *Let M be a given (finite, discrete) MDP, $M = (S, A, T, \gamma, R)$. We say that the MOMDP M' is a shaping-based multi-objectivization of M, iff $M' = (S, A, T, \gamma, \mathbf{R})$ with*

$$\mathbf{R}(s,a,s') = (R(s,a,s') + F_1(s,a,s'), \ldots, R(s,a,s') + F_m(s,a,s'))$$

If all shaping functions F_i, $i = 1, \ldots, m$ are potential-based, as defined in Eq. (1), we have the following properties:

- *Any policy π^* which is an optimal policy for M, is a Pareto optimal policy for M′.*
- *No other Pareto optimal policies for M′ exist, i.e., if π is not an optimal policy for M, π is not Pareto optimal in M′.*

Proof. The proof follows from the results in [29]. There, Ng et al. proved that if a policy is optimal for an MDP with reward function R, it is also optimal for the shaped MDP with rewards $R + F$ (and vice versa), provided that F is a potential-based shaping function. So, any policy π^* that is optimal for MDP M will also be optimal for each of the shaped rewards $R + F_i$. Since π^* maximizes the returns for all objectives, no policy which Pareto dominates π^* can exist (since such a policy would have to perform strictly better on at least one objective) and π^* must be part of the Pareto front for M′. Now suppose a policy π exists, which is part of the Pareto front of M′, but which is not optimal in M. Since π is suboptimal in M, according to Ng et al. [29] it must also be suboptimal for each of the $R + F_i$ objectives. However, this means that any policy π', that is optimal in M,[5] will achieve a strictly higher return for all objectives. Thus, π' Pareto dominates π and π cannot be part of the Pareto optimal set for M′, which contradicts our original assumption. □

Corollary 1. *Since all optimal policies of M′ (and thus also of M) achieve the highest expected return for each objective in M′, the Pareto front of M′ consists of a single point. Moreover, since Ng et al. [29] actually prove that the total order of policies is preserved when using potential-based shaping, and not just optimality, MOMDP M′ also has a total order over all policies. These all lie on a single line in the multi-objective value-function space, see Fig. 2.*

One may wonder what the purpose of this multi-objectivization is if all policies lie on the same line in the multi-objective space, i.e., if the objectives are all fully correlated. Why do we need extra objectives if they are correlated with all other objectives? The answer to that lies in the important distinction between return and reward. Rewards are generated at each time step for the state transition that occurred. Return is the discounted accumulation of these rewards. The returns (which measure the quality of a policy) of the objectives generated in the multi-objectivization we propose here are fully correlated, while the immediate rewards are not. These latter may differ wildly, and this heterogeneity in immediate rewards, combined with a guarantee of correlation of the total return, is the power of this proposed multi-objectivization.

[5] At least one such optimal policy must exist, see e.g., [36].

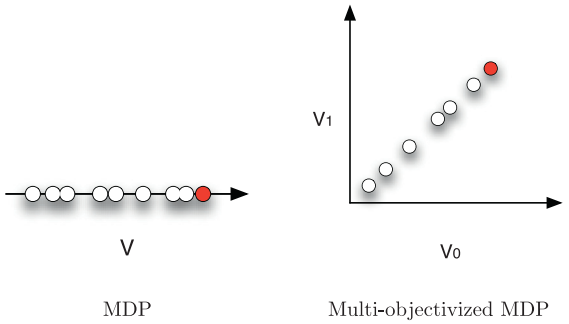


Fig. 2. Representative examples of an MDP and multi-objectivized MDP. Dots represent policies, and red dots are optimal policies. Note that in the multi-objectivized MDP, the value functions of policies are fully correlated between the different 'objectives'. Recall furthermore that this correlation is not necessarily true for the immediate reward + shaping. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

Whether a multi-objectivization will actually be successful in making a problem easier to solve depends on the formulation of the shaping functions themselves. The best potential function is V^* itself, the optimal value function of the problem.[6] While it is unlikely that the system designer can define a single potential function that equates V^* throughout the search space (which would amount to solving the problem), it is more likely that he can define several potential functions that correlate well with V^* in different parts of the state space, i.e., several rules of thumb for different situations,[7] or rules of thumb that weakly correlate with V^* throughout the state space. If these shapings are used to multi-objectivize an MDP, one can then attempt to strategically combine these signals, for which we will use ensemble techniques described in the next section.

6. Ensemble techniques in reinforcement learning

Ensemble techniques have been introduced in Reinforcement Learning by Wiering and van Hasselt [9]. Their approach involved several distinct reinforcement learning algorithms (agents) learning in parallel from the same experiences. The ensemble policy from which these experiences are generated is derived from a voting mechanism based on each agent's action preferences. Others have investigated the robustness of ensembles of neural network reinforcement learners, showing how these ensembles alleviate the problem of parameter tuning and are more stable [38,39] or how to select the best subset of ensemble agents for learning [40]. In our multi-objectivization setting, the diversity required for an ensemble will not primarily come from the algorithms or their parameters, which may all be the same, but from the reward signal, which is enriched with a different shaping signal for each ensemble agent.

The different ensemble strategies we will describe below can all be formalized as calculating a weighted (w_i) sum of each ensemble agent's preferences (p_i), where the actual choice for weight and preference functions will define the different strategies. The greedy ensemble policy then is:

$$\pi(s) = \arg\max_a \sum_i^n w_i(s)\, p_i(s,a) \quad (3)$$

[6] $V^*(s) = \max_a Q^*(s,a)$.

[7] E.g., shaping using kinetic (speed) or potential energy (height) in the Mountain Car domain [37], a problem where an underpowered car needs to build up momentum to climb a hill. These are opposite forces in this domain, as the car trades speed for height and vice versa, yet each is useful in a different situation: the car needs to focus on gaining speed when it can no longer gain height, and focus on gaining height when speed is already high.

The preference function $p_i(s, a)$ defines how high agent i values action a in state s. In the simplest case, $p_i(s, a) = Q_i(s, a)$, i.e., an agent's preference function is its estimated Q-function. The weight function $w_i(s)$ defines the relative importance of an agent in state s, i.e., how much an agent contributes to the ensemble decision relative to the other agents. Without loss of generality, we constrain w_i to: $\forall s : \sum_i^n w_i(s) = 1$. Without prior knowledge, it is advisable to set each agent's contribution to $w_i(s) = \frac{1}{n}$, independent of state.

Algorithm 1 describes the pseudocode for a greedy reward-function based ensemble of RL agents. At each step of the process, state s is observed, and action a is chosen given s, based on Eq. (3). Action a is executed, next state s' is observed, and each agent i observes reward R_i, which in the shaping case amounts to $R_i = R + F_i$. These are used to update each learning agent.[8] Eq. (3) gives a greedy policy; an ϵ-greedy, or soft-max policy can be defined analogously.

Algorithm 1 Greedy Ensemble of RL agents.

```
1: procedure GREEDY-ENSEMBLE
2:     initialize s
3:     for each step of episode do
4:         a ← arg max_a Σ_i^n w_i(s) p_i(s, a)
5:         take action a, observe r, s'
6:         for each learner i do
7:             update agent i with (s, a, R_i(s, a, s'), s')
8:         end for
9:         s ← s'
10:    end for
11: end procedure
```

Algorithm 2 illustrates a specific implementation of this ensemble of RL agents[9]: a greedy ensemble of standard Q-learners using the linear ensemble technique. The agents' preference function is their estimated Q-function, and all agents have equal contribution. Note that each agent i learns independently, in that there is no mixing of values. Only at the action selection stage is there 'mixing' between the agents.

Algorithm 2 Greedy linear ensemble of Q-learners.

```
1: procedure GREEDY-Q-ENSEMBLE
2:     initialize s
3:     initialize each learner
4:     for each step of episode do
5:         a ← arg max_a Σ_i^n (1/n) Q_i(s, a)
6:         take action a, observe r, s'
7:         for each learner i do
8:
```

$$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha\left[R_i(s,a,s') + \gamma \max_{a'} Q_i(s',a') - Q_i(s,a)\right]$$

```
9:         end for
10:        s ← s'
11:    end for
12: end procedure
```

Below, we describe four ensemble strategies, each defined by a different implementation of the weight and preference functions.

[8] For on-policy learners, the ensemble needs to choose next action a' as well, before updating the learners.

[9] Note that we can refer to such an algorithm as an ensemble of RL agents or a single agent learning multiple value-functions or policies.

6.1. Linear

The most straightforward ensemble strategy (also often used in multi-objective reinforcement learning in general), is to calculate a linear combination of the different estimated Q-functions: $p_i(s, a) = Q_i(s, a)$, as in Algorithm 2, line 5.

In this ensemble, the weight function serves two purposes:

1. to weigh the relative importance of the different ensemble agents,
2. to compensate for the differences in magnitude between the different agents' Q-functions.

Thus, we decompose the weight function into two weight functions: $w_{i,\,ri}$ and $w_{i,\,sc}$, respectively, used for the two goals described above. These are then combined in this way: $w_i(s) = \frac{w_{i,ri}(s) w_{i,sc}(s)}{\sum_i^n w_{i,sc}(s)}$. Unless prior information is available, which could come from a tuning phase prior to learning, or from expert knowledge on the efficacy of each reward shaping function,[10] the relative importance weight function is set to $w_{i,ri}(s) = \frac{1}{n}$, subject to $\sum_i^n w_{i,ri}(s) = 1$. Although the agents are learning on the same base reward signal, their shaping signals may have wildly varying scalings and therefore result in estimated Q-functions of varying scalings. This can be compensated for using the scaling weights $w_{i,\,sc}$, but, without prior knowledge, we set these weights to: $w_{i,sc}(s) = 1$.

Importantly, in contrast to what we claimed in prior work [42], this ensemble is not equivalent to a single learner learning with a single composite shaping (a reward shaping function $F(s, a, s') = \sum_i^n w_i(s) F_i(s, a, s')$), due to the non-linearity of the max operator employed [24]. A single composite shaping is a much more computationally and memory efficient combination of shapings, since it does not require the storage of several value functions, but may result in the loss of information, as any dimensionality reduction technique risks.[11] We will also compare with this approach in the experimental section.

6.2. Majority voting

Intuitively, in majority voting [9],[12] each agent casts a single vote, choosing its preferred action. Given equal weights, the greedy ensemble action is the one that received the most votes. Formally, an agent's preference function is this:

$$p_i(s,a) = \begin{cases} 1 & \text{if } a = \arg\max_b Q_i(s,b) \\ 0 & \text{else.} \end{cases}$$

In this case, the weight function only serves to weigh the relative importance of agents, since with voting, all agents' preference functions have the same codomain: (0, 1). Again, unless prior information is available, we set the weight function to: $w_i(s) = \frac{1}{n}$.

6.3. Rank voting

Rank voting [9] is a more fine-grained voting mechanism, where, as in majority voting, the lowest ranked action gets a score of 0, and the most preferred action a score of 1. In contrast, the in-between actions get a score proportional to their rank:

$$p_i(s,a) = \frac{\#\text{actions} - \text{rank(s, a)}}{n-1}$$

[10] Marivate and Littman [41] for example learn these weights in a limited setting.

[11] For example, if nine out of ten ensemble agents have attribute the highest expected return to action a, but the tenth believes this action will yield a very large negative return, in a linear combination, the fact that the majority agreed on action a will be lost, and another action will be deemed 'best' due to the effect of a single of the ensemble agents.

[12] In more recent work, van Hasselt preferred the term plurality voting [43].

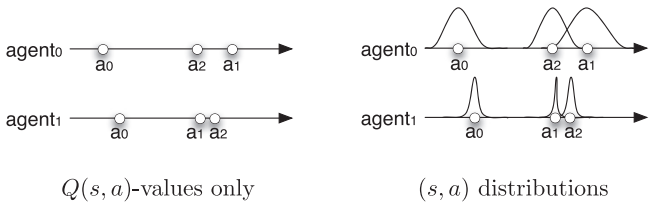


Fig. 3. Showing two agents' estimates (top and bottom, respectively). Determining which agent can be most confident about its estimates is impossible based on the estimated Q-values alone (left). Extra information is necessary (right). In this example, the second agent's estimates are deemed to be more reliable, as the actions' distributions are more significantly different/show less overlap.

where *rank*(*s, a*) outputs the index of (*s, a*) from [1,#actions]) in the sorted list according to some quality criterion. Commonly, one will sort according to $Q(s, a)$.

The weight function is again used to weigh the relative importance of agents.

6.4. Confidence-based

Adaptive objective selection is a technique we previously introduced specifically to solve CMOMDPs [44]. It builds on work on multi-objectivization in evolutionary computation where every optimization decision is based on feedback from only a single of the objectives [16]. This is possible since each of the objectives alone can be used to solve the original MDP.

Specifically, at every step, this ensemble technique tries to measure how confident each of the ensemble agents is about its estimates for the current state, and uses the most confident agent's estimates alone to make an action selection decision. Thus, the preference function is simply the *Q*-function, as in the linear ensemble. The difference with that ensemble lies in the weight function $w_i(s)$, which incorporates the measure of confidence:

$$w_i(s) = \begin{cases} 1 & \text{if } i = \arg\max_{agent} \text{confidence}(s, agent) \\ 0 & \text{else.} \end{cases}$$

We define confidence as an estimation of the likelihood that the estimates are correct. Higher-variance reward distributions will make any estimate of the average reward less confident, and always selecting the agent whose estimates are most likely to be correct will maximize the likelihood of correctly ranking the action set. This is loosely inspired by the use of confidence intervals in UCB [45].

To measure confidence in estimates, we model every (*s, a*)-pair as a distribution, and not just a mean (*Q*-value). This allows us to determine how well each agent can differentiate between the actions based on common statistical tests. See Fig. 3 for an illustration of this concept. Below, we describe an efficient way to represent these distributions and measure confidence for the case of tile-coding function approximation. For a discussion on how to represent these distributions in other cases, we refer the reader to our previous work [44].

Tile-coding provides a very natural way to represent (*s, a*)-pairs as distributions, without requiring the storage of extra information. For a (*s, a*)-pair, agent *i* can simply take the weights in $\theta_{i,a}$ activated by *s* as samples representing the distribution of that pair. Then, we can estimate confidence by applying a paired statistical test to the samples of every action (or of the estimated best and worst actions). Such tests calculate a *p*-value which indicates how likely it is to observe the given estimates under the hypothesis that they come from the same distribution. The smaller the *p*-value, the more likely it is the distributions are different, and that the agent can differentiate correctly between the actions. We can use a paired test, such as the paired Student's *t*-test or the Wilcoxon signed-rank test, because the weights for different actions will come from the same tiles in the same tilings, although stored in different weight vectors.

This confidence-based ensemble has several interesting properties. It makes its decisions a function of the state-space, which can account for different shapings being more or less reliable in different parts of the state space. Furthermore, it uses the different agents' estimates in a scale-invariant way. That is, its workings do not depend on the relative scalings of the shapings, since all statistical tests proposed are scale-invariant, and thus no parameters are introduced. This is a significant improvement over the simpler linear ensemble, which usually requires weight tuning, if only to align the magnitudes of the different shapings. Do note however that, despite using concepts such as distributions and statistical tests, there are no theoretical underpinnings to this confidence ensemble; it is no more than a heuristic.

7. Empirical validation

We empirically demonstrate the usefulness of the proposed approach in two domains: cart pole and the pursuit domain. All experiments are averaged over 100 trials for statistical significance, evaluated using the Student's *t*-test with $p = 0.05$. We always compared the following approaches, mainly by measuring their cumulative performance (an indication of how fast they learn), but also their final performance, showing how they (are) still converge(ing) to optimality:

1. vanilla $Q(\lambda)$-learning with pessimistic initialization,
2. vanilla $Q(\lambda)$-learning with optimistic initialization,
3. $Q(\lambda)$-learning shaped with a single shaping,
4. $Q(\lambda)$-learning shaped with a composite shaping, i.e., a linear combination of all the shapings ($\Phi(s) = \sum_i^n \frac{1}{n}\Phi_i(s)$),
5. an ensemble of $Q(\lambda)$-learners, each shaped with a single of the shapings, combined with one of the four ensemble techniques
   (a) linear
   (b) majority voting
   (c) rank voting
   (d) confidence.

We compare with optimistic initialization to analyze the effect of undirected uniform exploration, versus the more directed biased exploration that the shapings provide, which we hypothesize will be more effective. The shaping variants are always pessimistically initialized, so that exploration is almost exclusively (besides exploration induced by the action selection mechanism for example) driven by the shaping.

Furthermore, we perform two sets of experiments. One with normalized shapings, i.e., shapings with their potential function $\in [0, 1]$, and one with non-normalized shapings, to show how ensemble techniques are affected by the differences in magnitude between the shapings.

Lastly, we analyze the diversity induced by the different shaping signals.

7.1. Cart Pole

Cart Pole [46] is a task in which the learning agent controls a cart with a pole on top. The goal is to keep the pole balanced for as long as possible by moving the cart left and right within a given interval in a single dimension. The state space consists of the position of the cart, its velocity, the angle of the pole and its angular velocity $(x, \dot{x}, \theta, \dot{\theta})$. The agent receives step rewards of 0, and a final reward of -1 when the pole falls; in the experiments, performance is measured as the number of steps the pole is balanced, and an episode is limited to 1000 steps. To learn the task with RL, we use $Q(\lambda)$-learning with tile-coding function aproximation. We

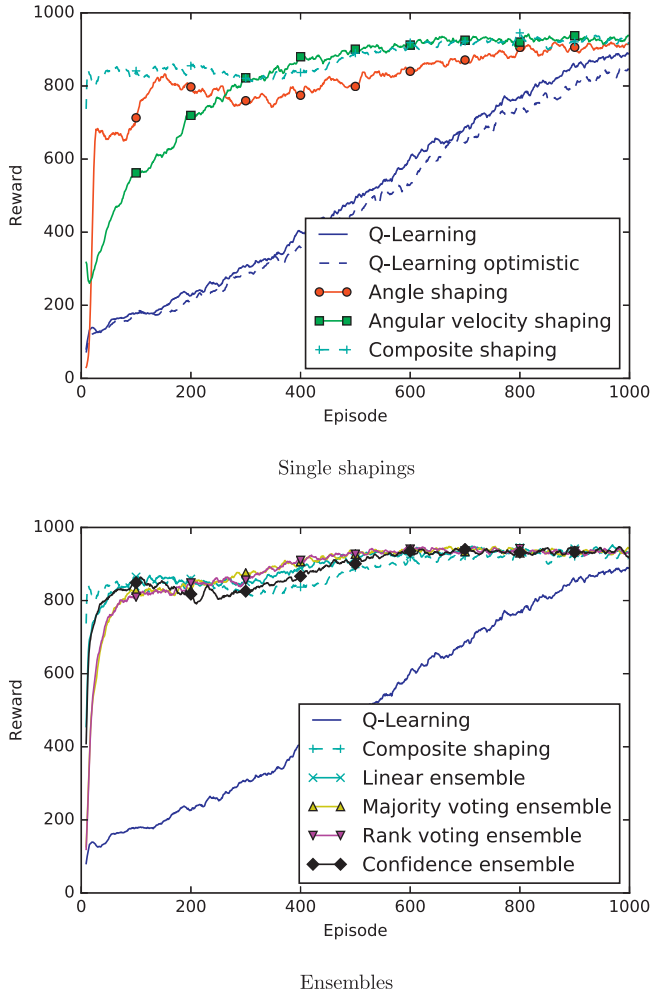


Fig. 4. Cart Pole – normalized shapings. The composite shaping outperforms the individual shapings, while the ensembles slightly outperform the composite shaping.

tuned the parameters to $\alpha = \frac{0.25}{16}$, $\gamma = 1$, $\epsilon = 0.05$, $\lambda = 0.25$, with 16 $10 \times 10 \times 10 \times 10$ tilings. We use 0 for optimistic initialization, and -1 for pessimistic.

We transform Cart Pole into a CMOMDP by multi-objectivizing the problem using two potential-based shaping functions [47]:

Angle encourages the agent to keep the angle of the pole close to 0, i.e., upright. Its potential function is defined as $\Phi_A(s) = -|\theta|$ (radians).

Angular velocity encourages the agent to keep the velocity of the pole low, since a fast moving pole is harder to control. Its potential function is defined as $\Phi_{AV}(s) = -|\dot{\theta}|$ (radians).

These shapings are scaled by 100 for optimal effect.[13] The angle is normalized by dividing by 0.20944 (the largest angle of the pole before failure in radians), and the angular velocity is normalized by dividing by 6, the maximum angular velocity.

Fig. 4 and Table 1 summarize the first experiment in Cart Pole with normalized shapings. As can be expected, both the angle and angular velocity shapings help the agent learn much faster than without this prior knowledge. Unguided, uniform exploration through optimistic initialization on the other hand results in slightly worse performance compared to the baseline vanilla $Q(\lambda)$, as the agent is encouraged to explore failure states too. Furthermore, a simple composite shaping (the average of the angle and angular velocity shapings) outperforms either of the individual shapings alone, while the ensembles slightly, but significantly, outperform this composite shaping. Arguably, in this case the composite shaping is preferable due to the lower computational and memory complexity, even though its sample complexity is slightly worse than the ensembles.

[13] The shaping tuning problem has been addressed by Harutyunyan et al. [47], where the authors propose to include the same shaping with a number of different scalings in the ensemble. That is beyond the scope of this article.

Table 1
Cart Pole – normalized. Averages of 100 trials of 1000 episodes each, measuring the number of steps the pole was up before falling. The standard error is indicated. Those results not significantly different from the best are indicated in bold.

Algorithm	Cumulative	Final
Q-Learning	502023 ± 30111	888.4 ± 28.9
Q-Learning optimistic	472069 ± 28615	846.0 ± 37.4
Angle shaping	810643 ± 7243	915.9 ± 13.4
Angular velocity shaping	815570 ± 17106	**938.2 ± 13.3**
Linear composite shaping	876975 ± 2876	918.2 ± 14.8
Linear ensemble	**891991 ± 2238**	**929.0 ± 13.3**
Majority voting ensemble	883880 ± 2631	**942.6 ± 12.0**
Rank voting ensemble	883077 ± 3286	**932.6 ± 14.9**
Confidence ensemble	881508 ± 2675	917.7 ± 15.1

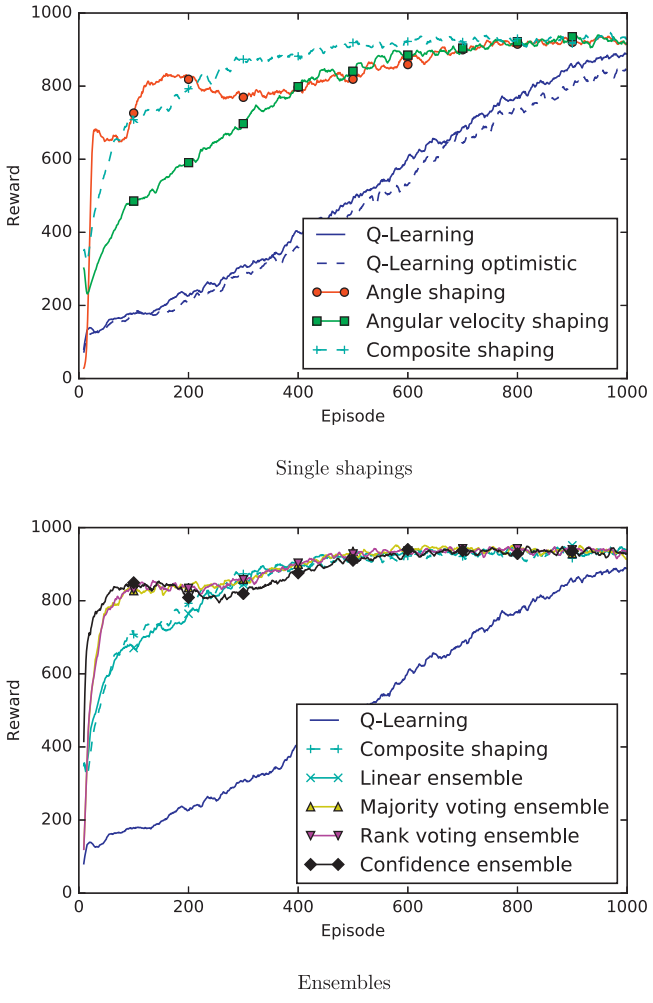


Fig. 5. Cart Pole – non-normalized shapings. The composite shaping and linear ensemble's performance is lower compared to the normalized shapings case, due to their sensitivity to the scaling of the signals they incorporate. The voting and confidence ensembles are scale-invariant and yield therefore similar performance to the normalized shaping case.

Table 2
Cart Pole – non-normalized. Averages of 100 trials of 1000 episodes each, measuring the number of steps the pole was up before falling. The standard error is indicated. Those results not significantly different from the best are indicated in bold.

Algorithm	Cumulative	Final
Q-Learning	502023 ± 30111	888.4 ± 28.9
Q-Learning optimistic	472069 ± 28615	846.0 ± 37.4
Angle shaping	826391 ± 6482	917.7 ± 12.7
Angular velocity shaping	767195 ± 23986	914.9 ± 15.3
Linear composite shaping	854326 ± 12198	**926.6 ± 13.8**
Linear ensemble	857799 ± 11613	**939.4 ± 12.7**
Majority voting ensemble	**883440 ± 2656**	913.5 ± 13.6
Rank voting ensemble	**883099 ± 2886**	**929.5 ± 13.1**
Confidence ensemble	**881709 ± 2782**	**928.4 ± 14.4**

In the case of non-normalized shapings (Fig. 5 and Table 2), the angular velocity shapings's performance drops significantly due to its much larger magnitude relative to the reward function. Consequently, the composite shaping and linear ensemble's performance drop too, since they are sensitive to the relative magnitudes of the different signals they are composed of. Importantly, the performance of the voting and confidence ensembles is unaffected, thanks to their scale-invariance. This suggests that the parameterless voting and confidence ensembles yield more robust combinations of shapings.

7.2. Pursuit domain

The Pursuit domain, or Predator/Prey, was proposed by Benda et al. [48] to investigate coordination mechanisms in a multi-agent system. The basic idea of pursuit is that a number of predators must capture a (number of) prey(s) by moving through a simple gridworld. Stone and Veloso [49] identify many variants of the problem and our implementation is as follows. There are two predators and one prey, and these can move in the four cardinal directions in a non-toroidal grid of 20 × 20 cells, as well as choose to stay in place. The prey is caught when a predator moves onto the same gridworld cell as the prey; predators are not allowed to share the same cell. The prey takes a random action 20% of the time, with the rest of the time devoted to moving away from the predators. To do that, it takes the action that maximizes the summed distance from both predators, making the problem harder than with a fully random prey. The predators are controlled by $Q(\lambda)$-learning agents, and both receive a reward of 1 when the prey is caught by one of them, and a reward of 0 the rest of the time; performance is measured in our experiments as the number of steps needed to catch the prey. The predators observe the relative x and y coordinates of the other predator and the prey. Tile-coding is used to discretize the state-space, with 32 randomly placed tilings, using tile-width 10.[14] Action selection is ϵ-greedy, with $\epsilon = 0.1$. Further parameters are $\gamma = 0.9$, $\lambda = 0.9$ and $\alpha = \frac{1}{10\times 32}$. We use 1 for optimistic initialization, and 0 for pessimistic.

We formulate a CMOMDP by multi-objectivizing the problem using three potential-based shaping functions[15]:

Proximity encourages a predator to move closer to the prey. Its potential function is defined as $\Phi_P(s) = -d(pred, prey)$, with d the Manhattan distance.

Angle encourages the predators to move to different sides of the prey, trapping it. It is defined to maximize the angle in radians between them and the prey to π: $\Phi_A(s) = \arccos(\frac{a\cdot b}{|a||b|})$, with a and b vectors pointing from the prey to the two predators, respectively.

Separation encourages the predators to move away from each other. Its potential function is defined as $\Phi_S(s, a, s') = d(pred_1, pred_2)$ with d again the Manhattan distance.

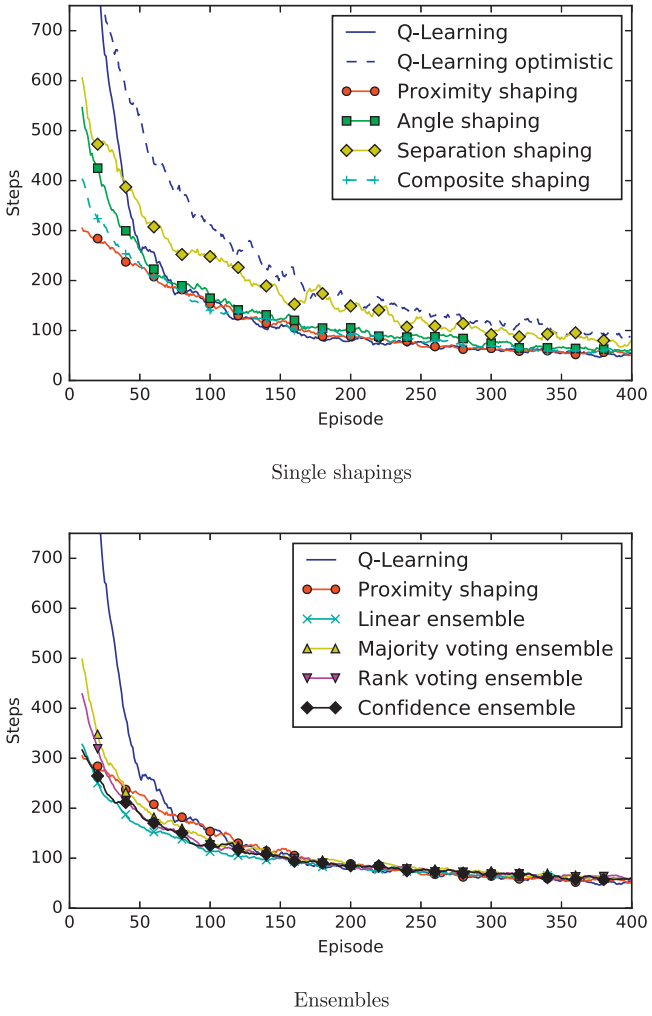


Fig. 6. Pursuit domain – normalized shapings. The ensembles yield similar final performance and better cumulative performance than the individual or composite shapings.

Proximity and Separation are normalized by dividing by $2 \times size$, with $size = 20$ both the width and height of the world; Angle is normalized by dividing by π. Furthermore, Proximity is implemented as $2 \times size - d(pred, prey)$, so that all shaping functions are positive, as theory indicates potentials should be when $\gamma < 1$ and there is no step-reward [51].

The results of the first experiment (with normalized shapings) are shown in Fig. 6 and Table 3. While the proximity and angle shapings improve performance compared to the baseline, the separation shaping appears not to be that useful and slightly degrades performance. Optimistic initialization results in much worse performance, due to the excessive exploration of unseen states far away from the prey. The composite shaping yields performance in-between the other shapings', not as good as the proximity shaping alone, showing that in this case the simple combination of signals is *not* better than the best of its constituting parts, unlike in the Cart Pole domain. The ensembles on the other hand outperform the composite shaping, and all but the majority voting one outperform the proximity shaping's performance.

[14] Tile-width is measured in whatever units each of the state-variables is measured.

[15] It has been proven that potential-based shaping in multi-agent RL does not alter the Nash Equilibria [50].

Table 3
Pursuit domain – normalized. Averages of 100 trials of 400 episodes each, measuring the number of steps needed to catch the prey. The standard error is indicated. Those results not significantly different from the best are indicated in bold. Most ensembles yield similar final performance and better cumulative performance than the individual or composite shapings.

Algorithm	Cumulative	Final
Q-Learning	73614 ± 3061	**56.0 ± 14.3**
Q-Learning optimistic	101233 ± 2161	101.6 ± 33.2
Proximity shaping	46309 ± 574	**49.7 ± 9.3**
Angle shaping	56099 ± 1177	**59.2 ± 19.4**
Separation shaping	75043 ± 1913	**104.6 ± 72.0**
Linear composite shaping	49265 ± 704	**49.9 ± 8.3**
Linear ensemble	**41447 ± 433**	**54.2 ± 11.4**
Majority voting ensemble	48985 ± 585	**61.7 ± 14.1**
Rank voting ensemble	45623 ± 442	**58.2 ± 14.2**
Confidence ensemble	43465 ± 551	**66.8 ± 20.0**

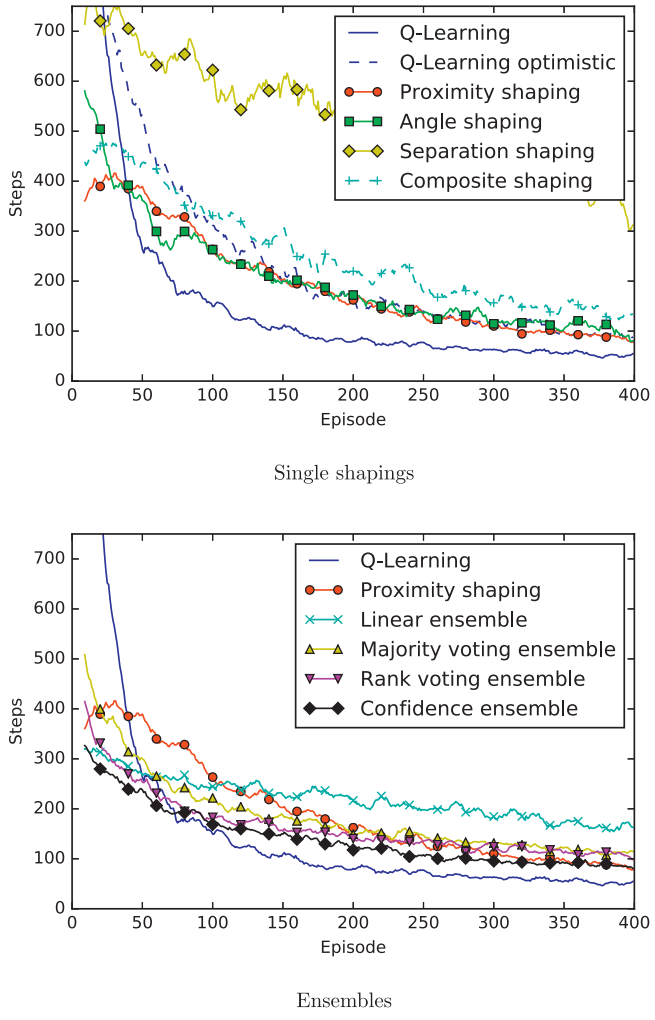


Fig. 7. Pursuit domain – non-normalized shapings. The individual and composite shapings, as well as the linear ensemble are most affected by the large differences in magnitude.

When one cannot normalize the shapings' magnitudes in this domain (because the size of the world is not known for example, or infinite), the performance yielded by the individual and composite shapings, as well as the linear ensemble is much worse, while the voting and confidence ensembles are again least affected due to their scale-invariance (Fig. 7 and Table 4). The fact that the performance of the ensembles still degrades is because they are scale-invariant across the different reward signals, but not within the different reward signals. That is, they are invariant

Table 4
Pursuit domain – non-normalized. Averages of 100 trials of 400 episodes each, measuring the number of steps needed to catch the prey. The standard error is indicated. Those results not significantly different from the best are indicated in bold.

Algorithm	Cumulative	Final
Q-Learning	73614 ± 3061	**56.0 ± 14.3**
Q-Learning optimistic	101233 ± 2161	101.6 ± 33.2
Proximity shaping	77912 ± 1070	**77.2 ± 30.2**
Angle shaping	81708 ± 2869	**99.1 ± 65.7**
Separation shaping	207095 ± 8633	372.9 ± 129.6
Linear composite shaping	101763 ± 2460	110.4 ± 30.7
Linear ensemble	88569 ± 1068	140.9 ± 28.6
Majority voting ensemble	75241 ± 687	98.0 ± 19.5
Rank voting ensemble	66477 ± 667	102.5 ± 35.4
Confidence ensemble	**57253 ± 680**	**79.7 ± 21.3**

to differences in scalings between the different reward+shaping signals, but they are not invariant to differences in scaling between the reward and shaping signals themselves, which is a different problem addressed elsewhere [47].

7.3. Diverity of shapings

Since our ensembles of shapings perform well in the experiments conducted, we can assume that the different shapings induced the diversity required for ensemble systems to work. In this section, we show that this diversity is indeed present and show how it evolves during the learning process. We measure diversity at a given time step of the learning process by calculating the Pearson correlation coefficient between Q-values. Specifically, in a given state, for each pair of agents in an ensemble, we calculate the Pearson correlation coefficient between the two agents' Q-values in that state. We average over all pairs of agents in the ensemble. A coefficient of 1 indicates full correlation (i.e., the agents fully agree on the relative quality of actions in that state), a coefficient of -1 indicates full anti-correlation (i.e., the agents have completely opposite ideas on the relative quality of actions in that state), and 0 indicates uncorrelated estimates (i.e., the agents disagree on the quality of actions, but in an inconsistent way). We hypothesize that full correlation (at least initially) and full anti-correlation are not useful for an ensemble, since with the former, there is no diversity, and with the latter, there is no agreement. We hypothesize that good trade-offs between diversity and agreement will yield a correlation coefficient around 0.

Fig. 8 shows this analysis for the four different ensembles in Cart Pole and the Pursuit domain, with normalized shapings. Observe that in both domains, early in the learning process, the estimated Q-values of the different ensemble agents are indeed uncorrelated, or even slightly anti-correlated. As learning progresses, the ensemble agents' estimates become more and more correlated, as they converge to their optimal value functions. Here we see the interplay between rewards and returns in the learning process. Immediate rewards are uncorrelated or even anti-correlated due to the different shapings sometimes disagreeing on what are 'good' states, and thus the initial estimates are also uncorrelated. As learning progresses, and rewards are propagated through the value function representation, the estimates converge to the actual expected returns, which *are* correlated, independent of the amount of anti-correlation that may exist between the shapings themselves. The initial diversity makes the ensemble learn faster, and the later lack of diversity is a result of the guarantee that the ensemble learns the optimal policy.

8. Conclusion

This paper introduces a novel perspective on diversity in ensemble techniques, where the necessary diversity is not derived

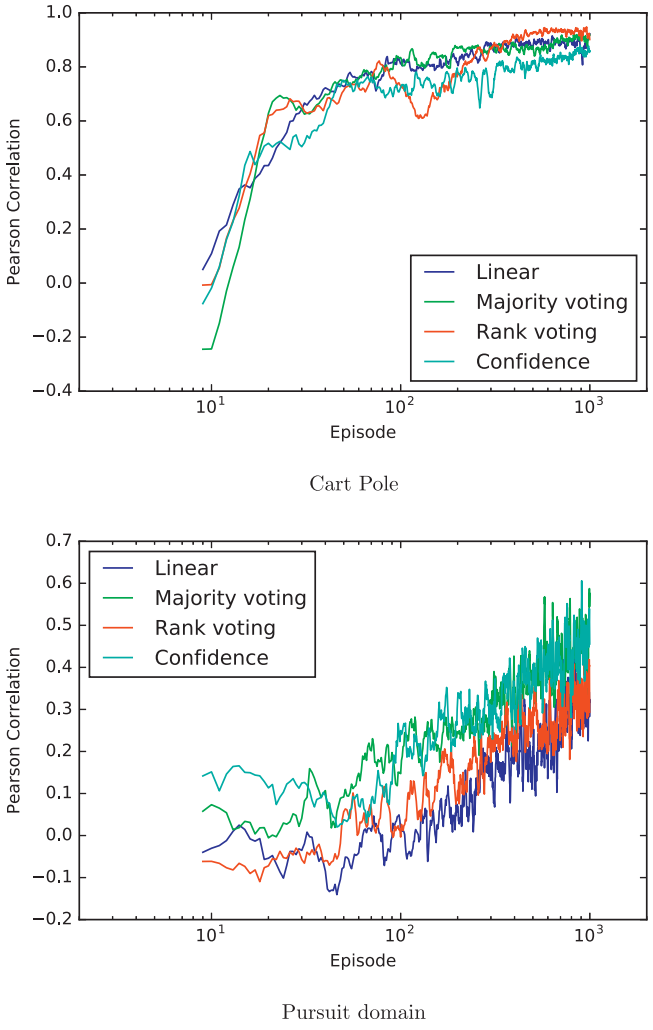


Fig. 8. Pearson correlation coefficient between the Q-values of the different ensemble agents over time. We can observe the diversity induced by the different reward shaping signals early on, which diminishes as the learners converge to their optimal value functions (which are fully correlated).

from the differences in the algorithms themselves or their input data, but rather from the way they evaluate their own performance, i.e., diversity of evaluation. This allows for the use of expert knowledge to complement an otherwise uninformative objective function (think for example of the objective function in classification, which is usually only approximate due to the sheer size and unavailability of all relevant data). Yet, certain guarantees are required for this approach to not compromise optimality of solutions, since changing the objective function typically means changing the problem (and thus the solutions to the problem).

Therefore, we developed a theoretical framework in the context of reinforcement learning that provably provides these guarantees, thus allowing the safe incorporation of many heuristics in a reinforcement learning process. It leverages the strong theory that underpins potential-based reward shaping and shows promising results in the experiments we conducted.

In a general context, future work involves the investigation of these ideas in other machine learning fields, as well as optimization literature, where the idea of multi-objectivization has had some successes, but without strong theoretical guarantees. More specifically in reinforcement learning, there remain several avenues for future research. Intra-signal scale-invariance, i.e., invariance to the relative magnitudes of reward and shaping signals is a vital avenue for research, since the magnitude shapings still require tuning in our ensemble approach. Harutyunyan et al. [47] recently started investigating this problem and provide a simple ensemble solution to this problem. The techniques should furthermore be validated in other applications, such as robotic tasks, where a reduction of sample complexity is very much needed, much more than in Cart Pole. How to automatically generate shaping functions is also an interesting question. It should not be hard to generate many combinations of state features, and many thousands of value functions can be tractably learned in parallel [52]. The difficulty will lie in building a shaping generator that satisfies conditions similar to that of a weak learner (a learner that generates classifiers that are merely better than random), as defined by Schapire [5], in this case providing shapings that yield slightly better performance than without shaping. Lastly, it remains unclear how many of the in supervised learning very successful ensemble algorithms such as AdaBoost could be used in reinforcement learning.

Acknowledgments

Tim Brys is supported by the FWO, and Anna Harutyunyan by the IWT-SBO project MIRAD (Grant no. 120057). This research has taken place in part at the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL research is supported in part by Grants AFRL FA8750-14-1-0069, AFRL FA8750-14-1-0070, NSF IIS-1149917, NSF IIS-1319412, USDA 2014-67021-22174.

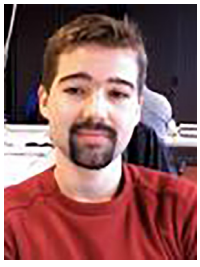
References

[1] S. Bonaccio, R.S. Dalal, Advice taking and decision-making: an integrative literature review, and implications for the organizational sciences, Organ. Behav. Human Decis. Process. 101 (2) (2006) 127–151.
[2] R. Polikar, Ensemble based systems in decision making, IEEE Circuits Syst. Mag. 6 (3) (2006) 21–45.
[3] A. Krogh, J. Vedelsby, et al., Neural network ensembles, cross validation, and active learning, Adv. Neural Inf. Process. Syst. 7 (1995) 231–238.
[4] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[5] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227.
[6] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139, doi:10.1006/jcss.1997.1504.
[7] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001.
[8] R. Mallipeddi, P.N. Suganthan, Q.-K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[9] M.A. Wiering, H. van Hasselt, Ensemble algorithms in reinforcement learning, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 38 (4) (2008) 930–936.
[10] J.D. Knowles, R.A. Watson, D.W. Corne, Reducing local optima in single-objective problems by multi-objectivization, in: Evolutionary Multi-Criterion Optimization, Springer, 2001, pp. 269–283.
[11] J.E. Fieldsend, Optimizing decision trees using multi-objective particle swarm optimization, in: Swarm Intelligence for Multi-objective Problems in Data Mining, Springer, 2009, pp. 93–114.
[12] C.A.C. Coello, Treating constraints as objectives for single-objective evolutionary optimization, Eng. Optim. 32 (3) (2000) 275–308.
[13] S. Watanabe, K. Sakakibara, Multi-objective approaches in a single-objective optimization environment, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, 2, IEEE, 2005, pp. 1714–1721.
[14] D.K. Saxena, K. Deb, Trading on infeasibility by exploiting constraints criticality through multi-objectivization: a system design perspective, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, IEEE, 2007, pp. 919–926.
[15] P. Raicevic, Parallel reinforcement learning using multiple reward signals, Neurocomputing 69 (16) (2006) 2171–2179.
[16] M.T. Jensen, Helper-objectives: using multi-objective evolutionary algorithms for single-objective optimisation, J. Math. Model. Algorithm. 3 (4) (2005) 323–347.
[17] S. Bleuler, M. Brack, L. Thiele, E. Zitzler, Multiobjective genetic programming: reducing bloat using spea2, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, 1, IEEE, 2001, pp. 536–543.
[18] E. de Jong, R. Watson, J. Pollack, Reducing Bloat and Promoting Diversity using Multi-objective Methods (2001) 11–18.
[19] D. Kim, Minimizing structural risk on decision tree classification, in: Multi-Objective Machine Learning, Springer, 2006, pp. 241–260.
[20] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, vol. 1, Cambridge University Press, 1998.

[21] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[22] J.N. Tsitsiklis, Asynchronous stochastic approximation and Q-learning, Mach. Learn. 16 (3) (1994) 185–202.
[23] J. Albus, Brains, Behavior and Robotics, McGraw-Hill, Inc., 1981.
[24] D.M. Roijers, P. Vamplew, S. Whiteson, R. Dazeley, A survey of multi-objective sequential decision-making, J. Artif. Intell. Res. 48 (2013) 67–113.
[25] Z. Gábor, Z. Kalmár, C. Szepesvári, Multi-criteria reinforcement learning, in: Proceedings of International Conference on Machine Learning, ICML, vol. 98, 1998, pp. 197–205.
[26] K. Van Moffaert, M.M. Drugan, A. Nowé, Scalarized multi-objective reinforcement learning: novel design techniques, in: Proceedings of the 2013 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, IEEE, 2013.
[27] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, E. Dekker, Empirical evaluation methods for multiobjective reinforcement learning algorithms, Mach. Learn. 84 (1–2) (2010) 51–80.
[28] I. Das, J.E. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems, Struct. Optim. 14 (1) (1997) 63–69.
[29] A.Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: theory and application to reward shaping, in: Proceedings of the Sixteenth International Conference on Machine Learning, vol. 99, 1999, pp. 278–287.
[30] M.E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: a survey, J. Mach. Learn. Res. 10 (2009) 1633–1685.
[31] W.B. Knox, P. Stone, Combining manual feedback with subsequent MDP reward signals for reinforcement learning, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 5–12.
[32] S. Devlin, M. Grześ, D. Kudenko, An empirical study of potential-based reward shaping and advice in complex, multi-agent systems, Adv. Complex Syst. 14 (02) (2011) 251–278.
[33] K. Efthymiadis, D. Kudenko, Using plan-based reward shaping to learn strategies in starcraft: broodwar, in: Proceedings of IEEE Conference on Computational Intelligence in Games, CIG, 2013.
[34] A. Harutyunyan, S. Devlin, P. Vrancx, A. Nowé, Expressing arbitrary reward functions as potential-based advice, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[35] T. Brys, T.T. Pham, M.E. Taylor, Distributed learning and multi-objectivity in traffic light control, Connect. Sci. 26 (1) (2014) 1–19.
[36] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, vol. 414, Wiley, 2009.
[37] S.P. Singh, R.S. Sutton, Reinforcement learning with replacing eligibility traces, Mach. Learn. 22 (1–3) (1996) 123–158.
[38] A. Hans, S. Udluft, Ensembles of neural networks for robust reinforcement learning, in: Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, ICMLA, IEEE, 2010, pp. 401–406.
[39] S. Faußer, F. Schwenker, Ensemble methods for reinforcement learning with function approximation, in: Multiple Classifier Systems, Springer, 2011, pp. 56–65.
[40] S. Faußer, F. Schwenker, Selective neural network ensembles in reinforcement learning: taking the advantage of many agents, Neurocomputing 169 (2015) 350–357.
[41] V. Marivate, M. Littman, An ensemble of linearly combined reinforcement-learning agents, in: Proceedings of Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.
[42] T. Brys, A. Harutyunyan, P. Vrancx, M.E. Taylor, D. Kudenko, A. Nowé, Multi-objectivization of reinforcement learning problems by reward shaping, in: Proceedings of 2014 International Joint Conference on Neural Networks, IJCNN, IEEE, 2014, pp. 2315–2322.
[43] H.P. van Hasselt, Insights in Reinforcement Learning, PhD thesis, Utrecht University, 2011.
[44] T. Brys, D. Kudenko, A. Nowé, M.E. Taylor, Combining multiple correlated reward and shaping signals by measuring confidence, in: Proceedings of the Twenty-Eight AAAI Conference on Artificial Intelligence, AAAI, 2014.
[45] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2–3) (2002) 235–256.
[46] D. Michie, R. Chambers, Boxes: An experiment in adaptive control, Mach. Intell. 2 (2) (1968) 137–152.
[47] A. Harutyunyan, T. Brys, P. Vrancx, A. Nowé, Multi-scale reward shaping via an off-policy ensemble, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1641–1642.
[48] M. Benda, V. Jagannathan, R. Dodhiawala, On Optimal Cooperation of Knowledge Sources – An Empirical Investigation, Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, WA, USA, 1986.
[49] P. Stone, M. Veloso, Multiagent systems: a survey from a machine learning perspective, Auton. Robot. 8 (3) (2000) 345–383.
[50] S. Devlin, D. Kudenko, Theoretical considerations of potential-based reward shaping for multi-agent systems, in: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems – Volume 1, 2011, pp. 225–232.
[51] M. Grześ, D. Kudenko, Theoretical and empirical analysis of reward shaping in reinforcement learning, in: Proceedings of International Conference on Machine Learning and Applications, ICMLA'09, IEEE, 2009, pp. 337–344.
[52] R.S. Sutton, J. Modayil, M. Delp, T. Degris, P.M. Pilarski, A. White, D. Precup, Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction, in: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems – Volume 2, International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 761–768.

Tim Brys is a final-year Ph.D. student at the AI-Lab of the Vrije Universiteit Brussel in Brussels. His research has spanned multiple topics, but he currently focuses on the incorporation of prior knowledge in reinforcement learning, looking at ways to efficiently incorporate demonstrations, prior experiences, domain knowledge, etc.

Anna Harutyunyan is a Ph.D. student at the AI lab of VU Brussel. Her research is primarily in reinforcement learning. She has been specifically interested in ways of augmenting a basic autonomous learner with auxiliary information, in order to aid its learning. The auxiliary information may stem either from the designer knowledge of a domain, or from other learners, attempting to solve the same problem, but perhaps from a different angle. In this context, she has been interested in reward shaping, and ensemble architectures in reinforcement learning.



Peter Vrancx is a postdoctoral researcher in the field of machine learning. For his Ph.D. research, he developed methods based on (evolutionary) game theory to analyze decentralized reinforcement learning algorithms. After his Ph.D., he worked on several research projects aimed at applying reinforcement learning in mechatronic systems and in telecom. His current research focuses on injecting additional knowledge into the reinforcement learning process (through shaping and ensembles), collaborative filtering applications, and the use of deep learning to obtain features for reinforcement learning.

Ann Nowé finished her Ph.D. in 1994 at the Vrije Universiteit Brussel in collaboration with Queen Mary and Westfield College, University of London. The subject of her Ph.D. is located in the intersection of Computer Science (A.I.), Control Theory (Fuzzy Control) and Mathematics (Numerical Analysis, Stochastic Approximation). After a period of 3 years as senior research assistant at the VUB, she became a Postdoctoral Fellow of the Fund for Scientific Research-Flanders (F.W.O.). Nowadays, she is a professor both in the Computer Science Department *Biography of the author(s) Click here to download Biography of the author(s): bio.pdf of the faculty of Sciences as in the Computer Science group of the Engineering.

Matthew E. Taylor received his doctorate from the University of Texas at Austin in the summer of 2008, supervised by Peter Stone. Matt then completed a two year postdoctoral research position at the University of Southern California with Milind Tambe and spent 2.5 years as an assistant professor at Lafayette College in the computer science department. He is currently an assistant professor at Washington State University in the School of Electrical Engineering and Computer Science and is a recipient of the National Science Foundation CAREER award. Current research interests include intelligent agents, multi-agent systems, reinforcement learning, transfer learning, and robotics.