Inverse Reinforcement Learning with Evaluation

Valdinei Freire da Silva and Anna Helena Reali Costa
Laboratório de Técnicas Inteligentes - Escola Politécnica
Universidade de São Paulo - São Paulo, BRAZIL
Email: {valdinei.silva|anna.reali}@poli.usp.br

Pedro Lima
Institute for System and Robotics
Instituto Superior Técnico - Lisboa, PORTUGAL
Email: pal@isr.ist.utl.pt

***Abstract*— Reinforcement Learning (RL) is a method that helps programming an autonomous agent through humanlike objectives as reinforcements, where the agent is responsible for discovering the best actions to fulfil the objectives. Nevertheless, it is not easy to disentangle human objectives in reinforcement like objectives. Inverse Reinforcement Learning (IRL) determines the reinforcements that a given agent behaviour is fulfilling from the observation of the desired behaviour. In this paper we present a variant of IRL, which is called IRL with Evaluation (IRLE) where instead of observing the desired agent behaviour, the relative evaluation between different behaviours is known by the access to an evaluator. We present also a solution for this problem under the assumption that a relative linear function that preserves the order assumed by the evaluator exists and that the evaluator evaluates policies instead of behaviours. This is posed as a linear feasibility problem, whose solution is well known. Results of simulations of a set of heterogeneous robots in a search and rescue scenario are presented to illustrate the method and the possibility to transfer the learned reinforcement function among robots.**

I. Introduction

Reinforcement Learning (RL) [1] is a learning method where an autonomous agent learns an action policy based on its own experience. This policy is inferred from a process of trial and error interactions with an environment. The process is guided by the agent itself and the environment does not return a complete evaluation of the agent's action, but a partial one, based on instant reinforcements associated to the corresponding environment's state transition and agent's chosen action. An autonomous agent can receive the reinforcement through a reinforcement sensor or be programmed through a reinforcement definition, so that it calculates when reinforcement occurs through its ordinary sensors.

As artificial agents are created to fulfil its creator's desire (for instance, a human or other artificial agent), RL allows the creator defining the task of an artificial agent using a more abstract level of description. The decision of the best way of acting is learned autonomously by the artificial agent.

However, when the task involves multiple objectives (for instance, moving fast to a target position and spending little energy), defining an adequate reinforcement function that represents the creator's desire is not at all obvious. The study of an human's desire has been conducted under the name preference elicitation [2], [3], where queries, that a human can answer easily, are considered, instead of asking to the human for a complete definition of his preference.

Ng and Russell [4], under the designation of Inverse Reinforcement Learning (IRL), have already reached some results concerning the problem of eliciting a behaviour preference, relying on a second agent acting optimally. We claim that it is easier for a creator to compare two executions to accomplish a task than defining completely the task. Therefore, considering the creator as a relative evaluator, we propose an extension of the IRL method, the IRL with Evaluation (IRLE) method.

In IRLE it is considered an evaluator that cannot execute or does not know the optimal policy or cannot define explicitly the reinforcement function, but can decide between two behaviours which one is the best; and the agent must fulfil the evaluator preferences. This paper defines the IRLE problem and presents a solution within particular conditions: the evaluator evaluate policies and there exists a linear evaluation function that accomplishes the same evaluations. Experiments concerning the transfer of the learned reinforcement function among agents are also made.

The theoretic models behind RL and IRL problems are presented in Sections II and III, respectively. The IRLE problem and a solution to the IRLE problem using a linear feasibility system are presented in Section IV. Experiments concerning such solution are presented in Section V. Conclusions and prospective work are presented in Section VI.

II. Reinforcement Learning

In the RL literature, Markov Decision Processes (MDPs) [5] are adopted as simplified models of real problems [1]. An MDP is defined by a tuple $[\mathcal{A}, \mathcal{S}, T(s_{t+1}|s_t, a_t), r(s, a)]$, where $\mathcal{A}$ is a finite set of possible actions a, $\mathcal{S}$ is a finite set of possible states s, $T(s_{t+1}|s_t, a_t)$ represents transition probabilities in a stationary process that has the Markov property and $r(s, a)$ is a bounded expected reinforcement function. A process has the Markov property if the next state depends only on the current state and action, and not on the past history of the process, i.e., $T(s_{t+1}|s_t, a_t, \ldots, s_0, a_0) = T(s_{t+1}|s_t, a_t)$.

The basic idea behind RL is that the agent can learn how to solve an MDP task through repeated interactions with the environment. Each time the agent performs an action $a \in \mathcal{A}$ in some state $s \in \mathcal{S}$, the environment reaches a new state $s' \in \mathcal{S}$ and the agent receives a reinforcement $r(s, a)$ that indicates the immediate value of this state-action pair.

The agent must find out a stationary policy of actions $a_t^* = \pi^*(s_t)$ that maximises the expected value function $V^\pi(s)$, which represents the expected reinforcement incurred for a policy π, where $\pi^*(s) = \arg\max_\pi[V^\pi(s)]$ [5]. It is common to assume the discounted-reinforcement value function, which makes use of a discount factor $\gamma \in (0,1]$ that forces short-term reinforcements to be more important than long-term reinforcements. $V^\pi(s)$ is thus defined by:

$$V^\pi(s) = \lim_{N\to\infty} \mathrm{E}[\sum_{t=0}^{N} \gamma^t r(s_t, a_t) | s_0 = s].$$

The RL problem modelled as an MDP can be solved by the Q-Learning algorithm [6], which finds an optimal policy incrementally without knowledge of the transition probabilities of the environment model. Q-Learning estimates a value function $Q(s,a)$ for each state-action pair. This value function is recursively calculated by:

$$\begin{aligned} Q_{t+1}(s_t,a_t) = \;& Q_t(s_t,a_t) + \alpha_t[r(s_t,a_t) \\ & +\gamma \max_a Q_t(s_{t+1},a) - Q_t(s_t,a_t)], \end{aligned}$$

where α_t is the learning rate and γ is the discount factor. The optimal value function is calculated by $V^*(s) = \max_a Q(s,a)$ and the optimal policy is $\pi^*(s) = \arg\max_a Q(s,a)$.

A. State space and features

Although the state space must be fully observed to meet the Markov condition, reinforcements can be defined compactly by a **vector of features** $\phi : \mathcal{S} \times \mathcal{A} \to [0,1]^k$ over states and actions and a **weight vector** $\mathbf{w} \in \Re^k$ [7]. Then, the reinforcement function is defined by:

$$r(s,a) = \langle \mathbf{w}, \phi(s,a) \rangle = \sum_{i=1}^{k} w_i \phi_i(s,a),\ \forall a \in \mathcal{A}\ \forall s \in \mathcal{S},$$

where k is the number of features. The features can represent local properties (for instance, has just hit a wall, is in a goal state, has just found a resource, has just walked, etc.) or, in the ordinary case, the global state space, where $\phi_{s,a}(s',a') = 1$ if $s' = s$ and $a' = a$, and $\phi_{s,a}(s',a') = 0$ if $s' \neq s$ or $a' \neq a$. In such case $k = |\mathcal{S}| \cdot |\mathcal{A}|$ but in general $k \ll |\mathcal{S}| \cdot |\mathcal{A}|$.

B. Feature Expectations

In the same way that $V^\pi(s)$ is the expected value of a state s under policy π, we can define the expected value of a policy π, which will be called **evaluation**. The evaluation of a given policy $\pi \in \Pi$, where Π is the set of all possible policies, is a function $Eval : \Pi \to \Re$ and is defined by the expected value of $V^\pi(s)$:

$$\begin{aligned} Eval(\pi) &= \mathrm{E}_{s_0 \sim D}[V^\pi(s_0)] = \textstyle\sum_{s\in\mathcal{S}} p(s_0 = s) V^\pi(s) \\ &= \langle \mathbf{w}, \mu(\pi) \rangle, \end{aligned}$$

where D is the probability distribution to initial state s_0, $p(s_0 = s)$ is the probability of initial state s_0 being s, and $\mu(\pi) \in \Re^k$ is the **feature expectations** that represents the expected discounted accumulated feature value vector [7] and is defined by:

$$\mu(\pi) = \sum_{s\in\mathcal{S}} p(s_0 = s) \mathrm{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a_t) | \pi].$$

III. Inverse Reinforcement Learning

Before being used as a learning model for artificial agents, RL was thought as a model of the learning process of animals, where reinforcement is a model to describe animals' objectives [8]. An interesting question when analysing the behaviour of an animal is how to determine which reinforcement such animal is considering to present such behaviour, i.e., what is the utility function that can explain the observed behaviour.

The same question can be applied to artificial agents. The problem is called Inverse Reinforcement Learning (IRL) [9] and is informally presented as follows:

> **Given** 1) measurements of an agent's behaviour over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agent; and 3) if available, a model of the environment.
>
> **Determine** the reinforcement function being optimised.

Every solution to the IRL problem within finite state space must meet the following inequality [1]:

$$(\mathbf{T}_{\pi^*} - \mathbf{T}_a)(\mathbf{I} - \gamma \mathbf{T}_{\pi^*})^{-1} \mathbf{r}^{\pi^*} \succeq \gamma^{-1}(\mathbf{r}^a - \mathbf{r}^{\pi^*}),\ \forall a \in \mathcal{A},\ (1)$$

where $\succeq$ denotes vector inequality, $\mathbf{T}_a$ are the state transition probabilities when executing action a, π^* is the optimal policy that represents the agent's behaviour, and $\mathbf{r}^a = (r(s,a))_{|\mathcal{S}|}$ is a state-reinforcement vector for action a , i.e., the reinforcement for each state where the action was already chosen.

This inequality characterises the set of all reinforcement functions for which a given policy π^* is optimal. However, an infinite number of reinforcement functions $r(s,a)$ can be found, including the trivial solution, where $r(s,a) = k$ for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}$ and $k \in \Re$.

Since the condition given in (1) presents many degenerate solutions, Ng and Russell [4] propose simple heuristics for removing this degeneracy, e.g., choosing a "simple" solution where most elements of $r(s,a)$ are relatively small with respect to the largest one, or considering solutions where the optimal policy is "clear", i.e., the value-function difference between the optimal policy and alternative ones is the largest possible.

IV. IRL with Evaluation

Another way of looking at the problem of not knowing the reinforcement function is to use, instead of the optimal policy, the help of an evaluator, who can decide, between two policies, which one is the best. We introduce here the Inverse Reinforcement Learning with Evaluation (IRLE) problem, defined as follows:

> **Given** 1) relative evaluation of measurements of arbitrary agent's behaviours over time, 2) if needed, measurements of the sensory inputs to the agent for

[1]This inequality is an extension of the inequality $(\mathbf{T}_{\pi^*} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi^*})^{-1}\mathbf{r}_{\pi^*} \succeq 0$ presented by Ng and Russell [4]. The original inequality considers the restriction $r(s,a) = r(s,b) = r(s)$ for all $a, b \in \mathcal{A}$ and for all $s \in \mathcal{S}$.

each behaviour; and 3) if available, a model of the environment.

Determine the utility function being used by the evaluator.

While the original IRL problem attempts to obtain the reinforcement function through an agent executing an optimal policy, the IRLE problem attempts to obtain the reinforcement function through an evaluator, who can evaluate pairs of behaviours relatively, but neither knows an optimal policy nor can describe the utility function to the agent.

The measurements used in this paper are feature expectations of a policy (μ', μ''), and it is considered that the evaluator receives such measurements from the agent, deciding which policy is the best[2]. Fig. 1 shows a scheme of such interaction.

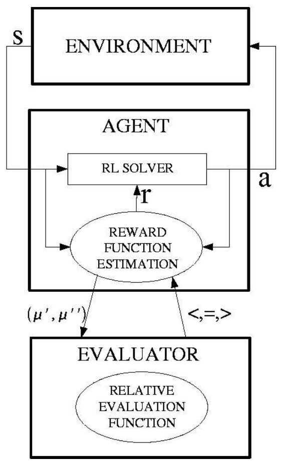

Fig. 1. IRLE architecture model.

We propose a technique based on constraints and measurements of the agent's policy (feature expectation). This technique, designed as the Incremental Feasibility Algorithm, builds incrementally and iteratively a linear feasibility problem, whose solution is also a solution to the IRLE problem.

A. Formal definition

Definition 1 (Preferred optimal policy): Consider an MDP $[\mathcal{A}, \mathcal{S}, T, r]$ and let $\mathcal{A}$ be an ordered set, if there exist different best actions $a', a'' \in \mathcal{A}$ for a state $s \in \mathcal{S}$, i.e., $Q(s, a') = Q(s, a'') = \max_{a \in \mathcal{A}} Q(s, a)$, then the preferred optimal policy chooses between a' and a'' as ordered by the set $\mathcal{A}$.

Definition 2 (IRLE problem): Consider an MDP\R $[\mathcal{A}, \mathcal{S}, T]$ (an MDP without reinforcement function), a vector of features ϕ, the preferred optimal policy $\pi_{\mathbf{w}}$ for an MDP $[\mathcal{A}, \mathcal{S}, T, r(s,a) = \langle \mathbf{w}, \phi(s,a) \rangle]$, the set Π of all stationary deterministic policies $\pi : \mathcal{S} \to \mathcal{A}$, and the access to a relative evaluation function $f_e(\mu', \mu'')$ that chooses the best between two feature expectations μ' and μ'', where:

$$f_e(\mu', \mu'') = \begin{cases} 0, & \text{if } \mu' \text{ and } \mu'' \text{ are indifferent} \\ 1, & \text{if } \mu' \text{ is preferred to } \mu'' \\ -1, & \text{if } \mu'' \text{ is preferred to } \mu' \end{cases} .$$

[2]A more general case is when the evaluator itself observes the agent's behaviour and has its own measurements, not necessarily equal to the agent's.

Then, determine a set W_e, such that:

$$f_e(\mu(\pi_{\mathbf{w}}), \mu(\pi)) \geq 0, \ \forall \mathbf{w} \in W_e \ \forall \pi \in \Pi.$$

Here, it was considered that the agent can only solve MDPs and the agent must find the reinforcement function that generates the best optimal policy regarding the evaluator.

B. Incremental Feasibility Algorithm

A linear feasibility problem with k variables and m constraints is defined by the following linear inequality system:

$$c_i : \sum_{j=1}^{k} a_{ij} w_j \left(\substack{\leq \\ = \\ \geq} \right) b_i \text{ for all } i = 1, 2, \ldots, k$$

where $\mathbf{w} = (w_j)_k$ are the variables of interest. If it is known that the feasibility set is bounded, then the solution to a linear feasibility problem is a polyhedron with vertices $\mathbf{w} \in P_W$.

The incremental feasibility algorithm, described in Table I, adds new linear constraints to old ones in such a way that, by the end of the algorithm, the feasible set is a solution to the problem defined in Definition 2.

The algorithm considers an access to two functions: $(\pi, \mu) = RL{-}solver(\mathbf{w})$, that receives an weight vector $\mathbf{w}$ and returns the optimal policy π to the MDP $[\mathcal{A}, \mathcal{S}, T, r(s,a) = \langle \mathbf{w}, \phi(s,a) \rangle]$ and the respective feature expectations μ; and $(W, P_W) = LIS{-}solver(G)$, that receives a set of linear inequalities G and returns the convex feasible set W to G and the polyhedron P_W that underlies the set W.

The set G is formed by linear constraints that are already satisfied. The set P_W is formed by vertices $\mathbf{w} \in \Re^k$ and P_W defines the polyhedron satisfying all linear constraints $c \in G$. The set H is formed by known linear constraints that are not satisfied yet.

The algorithm uses the current solution W, which satisfies G, to query the evaluator, obtaining new constraints, until G is enough to solve the problem of Definition 2.

Note that the $RL{-}solver$ can take advantage of off-policy techniques to calculate more than one policy at a time or use the previous solution as an initial guess for new problems [10].

The convergence property of the Incremental Feasibility algorithm is guaranteed by the following theorem, which is proved in appendix:

Theorem 1: Let W^* be the maximal solution to the problem in Definition 2, i.e.,

$$W^* = \{\mathbf{w} | \quad \mathbf{w} \in [0,1]^k \text{ and } \|\mathbf{w}\|_1 = 1 \text{ and } \min_{\pi \in \Pi} f_e(\mu(\pi_{\mathbf{w}}), \mu(\pi)) = 0\};$$

and W be the set returned by the Incremental Feasibility algorithm. If there exists a function $f_{\mathbf{w}^*}(\mu', \mu'') = sign(\langle \mathbf{w}^*, (\mu' - \mu'') \rangle)$, where $\mathbf{w}^* \in [0,1]^k$ and $\|\mathbf{w}^*\|_1 = 1$, such that $f_{\mathbf{w}^*}$ imposes in Π the same order imposed by f_e in Π, then $W \subseteq W^*$.

V. Experiments

A robot-rescue environment, originally proposed by Melo *et al.* [11], was chosen to test the IRLE algorithm proposed. The environment is a room scenario, where there are rooms, doors

TABLE I

INCREMENTAL FEASIBILITY ALGORITHM.

```
Define initial constraints:
  • $c_0 : \sum_{i=1}^{k} w_i = 1$
  • $H = \{c_0\}$
  • $c_i : w_i \geq 0$ for all $i = 1, 2, \ldots, k$
  • $G = \{c_1, c_2, \ldots, c_k\}$
Do
  • Choose a constraint $c \in H$
  • Make $H = H - \{c\}$ and $G = G \cup \{c\}$
  • Make $(W, P_W) = LIS{-}solver(G)$
  • Make $H = \emptyset$
  • For each pair $(\mathbf{w}', \mathbf{w}'') \in P_W^2$ such that $\mathbf{w}' \neq \mathbf{w}''$
    – Make $(\pi', \mu') = RL{-}solver(\mathbf{w}')$ and $(\pi'', \mu'') = RL{-}solver(\mathbf{w}'')$
    – Query the evaluator between feature expectations $\mu'$ and $\mu''$ and obtain $eval = f_e(\mu', \mu'')$
    – Build the constraint $c_{\mathbf{w}',\mathbf{w}''}$
        $$c_{\mathbf{w}',\mathbf{w}''} : \begin{cases} \langle \mathbf{w}, (\mu' - \mu'') \rangle = 0, & \text{if } eval = 0 \\ \langle \mathbf{w}, (\mu' - \mu'') \rangle > 0, & \text{if } eval = 1 \\ \langle \mathbf{w}, (\mu' - \mu'') \rangle < 0, & \text{if } eval = -1 \end{cases}$$
    – If $c_{\mathbf{w}',\mathbf{w}''}$ is linearly independent of $G$, make $H = H \cup \{c_{\mathbf{w}',\mathbf{w}''}\}$
Until $H$ is empty
Return $(W)$
```

and staircases and a robot must rescue (take away) victims to any one of the exits (see Fig. 2).

The experiment has two objectives: 1) to illustrate the application of the Incremental Feasibility algorithm by learning the weight vectors applied to features that describes the reinforcement function; and 2) to test the transfer of knowledge about the relative evaluation function f_e among robot types. Three types of robots with different skills were considered to learn the relative evaluation function and to transfer such knowledge among them. They are (adapted from Melo *et al.* [11]):

- the Crawler, which has tracker wheels and is capable of climbing and descending stairs. It is able to open doors only by pushing;
- the Puller, which is a wheeled mobile manipulator, able to open doors either by pushing or pulling. However, it is not able to climb stairs; and
- the Walker, which is a wheeled robot, neither able to climb stairs nor open doors.

A. Experiment preparation

As each robot has different abilities, their state transition probabilities are different. However, all of the robots have the same multiple objective: to rescue the victim and keep the victim's health. When rescuing a victim, we have to define features of such mission so that a robot can maximise their weighted sum. Here we will consider three features (vector of features $\phi \in [0, 1]^3$) : $\phi_1 =$ `Walk` (occurs when the robot walks), $\phi_2 =$ `Door` (occurs when the robot crosses doors) and $\phi_3 =$ `Stair` (occurs when the robot climbs or descends stairs). Such features were chosen as a simplification, since taking hurt people through stairs or narrow places (doors) can worsen their conditions.

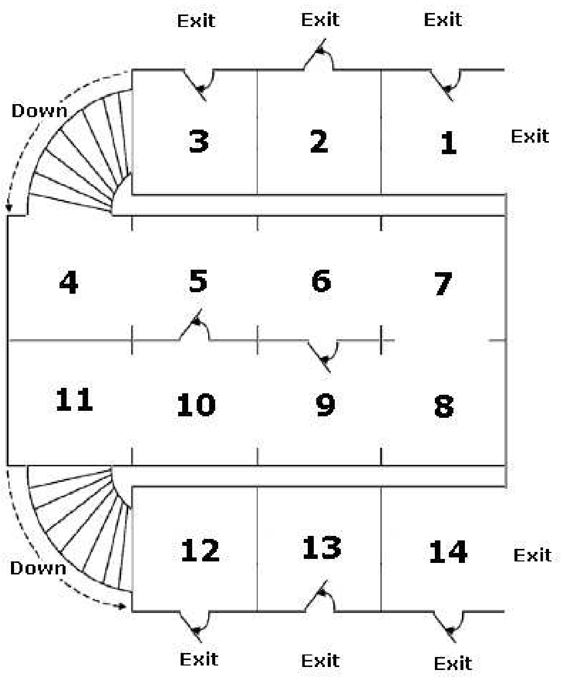

Fig. 2. Environment used in the experiments. Each door opens only in one direction and the robot must leave through any exit.

The robots have three actions: forward, backward and room-cross. The forward action takes the agent to the next room (as indicated by the room numbers), whereas the backward action takes the agent to the previous room. The cross-room action changes to the room in the other side of the door. If the agent does not have the skill to execute the action (climb stairs, pull or push doors), it remains in the same room. Every time step before exiting presents the feature `Walk` with value 1, whereas `Door` and `Stair` features are associated to the observations of respective behaviours, in both cases presenting values 1.

The evaluator considered for this experiment uses the following relative evaluation function:

$$f_e(\mu', \mu'') = \langle \mathbf{w}_e, (\mu' - \mu'') \rangle,$$

where $\mathbf{w}_e$ is a weight vector for each feature. Two different weight vectors (designed as objectives) were considered $\mathbf{w}_e = \mathbf{w}_1 = (-0.10, -0.15, -0.75)$ and $\mathbf{w}_e = \mathbf{w}_2 = (-0.10, -0.30, -0.60)$. The two vectors weight the features differently, i.e., for $\mathbf{w}_1$ there is advantage crossing some of the doors, while for $\mathbf{w}_2$ there is no advantage doing so. Since all of the features are undesirable, weights are negatives in this case. When considering negative weight vectors, in the Incremental Feasibility algorithm, w_j must be changed to $-w_j$ in constraints $c_0, c_1, \ldots, c_k$, where $k = 3$.

In both objectives it is desired to get to the exit as fast as possible, since the weights are negative. Due to the symmetry regarding rooms and stairs positions, the feature `Stair` does not affect optimal policies, since there is no other option to eventually reach an exit rather than going through some stairs when the robot is on rooms $4, 5, \ldots, 11$. So, different weights of this feature are indistinguishable to any optimal policies of the robots. The robot Walker never perceives the feature `Door`, so it is indifferent to the weight to such feature.

The purpose of these experiments is leading each type of robot to learn a reinforcement function, and checking how such function transfers to other types of robots for the same objectives. Note that the algorithm, presented in Table I, guarantees the optimal policy only for robots with the same skill (same state transition functions). On the other hand, the objectives are robot independent, hence, the reinforcement functions are also robot independent.

Table II shows the optimal evaluation $Eval(\pi^*)$, where π^* is the optimal policy, for all robot types reaching objectives w_1 and w_2 using $\gamma = 0.99$ and distribution $s_0 \sim D$ uniform in $\mathcal{S}$. Two notes must be made: 1) all of them reach different optimal policies (feature expectations) based on their skill, being the crawler the more skilled and the walker the less skilled; 2) this is not a comparison of objectives, since the objectives were chosen arbitrarily for the experiment purposes.

TABLE II

OPTIMAL EVALUATIONS FOR EACH ROBOT AND EACH OBJECTIVE.

Robots	Walker	Puller	Crawler
$\mathbf{w}_1$	-0.852	-0.794	-0.745
$\mathbf{w}_2$	-0.774	-0.768	-0.681

B. Learning weight vectors

The Incremental Feasibility algorithm was executed for each scenario (robot and objective) and after finding the polyhedron $P_{W^{robot}_{objective}}$ which represents the possible weights specific to each objective and robot, averages of the weight vectors $\mathbf{w}^{robot}_{objective}$ (average among vertices $\mathbf{w} \in P_{W^{robot}_{objective}}$) were determined. Table III shows the average weight vectors that were learned by each (robot type, objective) pair and the number of queries to the evaluator made before convergence (see Table I).

TABLE III

AVERAGE WEIGHTED VECTORS LEARNED FOR EACH OBJECTIVE AND ROBOT AND THE NUMBER OF QUERIES MADE BEFORE CONVERGENCE.

Weights	$\mathbf{w}_1^{Wlk}$	$\mathbf{w}_2^{Wlk}$	$\mathbf{w}_1^{Pll}$	$\mathbf{w}_2^{Pll}$	$\mathbf{w}_1^{Crl}$	$\mathbf{w}_2^{Crl}$
Walk	-0.338	-0.338	-0.242	-0.180	-0.162	-0.127
Door	-0.333	-0.333	-0.375	-0.507	-0.262	-0.477
Stair	-0.329	-0.329	-0.383	-0.313	-0.576	-0.396
Queries	3	3	28	15	15	10

Since the robot Walker is not affected by neither the `Door` feature nor the `Stair` feature, it learned little about the weight vectors before converging to its optimal policy with few queries. Considering the other robots, the ratios between the weights for features `Walk` and `Door` are approximately preserved when compared to the original ones ($\mathbf{w}_1$ and $\mathbf{w}_2$, but the same is not true for the feature `Stair`, due to its indistinguishability.

C. Knowledge transfer

Each learned weight vector was transferred to each type of robot, i.e., for each $\mathbf{w}^{robot}_{objective}$ was considered the reinforcement function $r(s,a) = \langle \mathbf{w}^{robot}_{objective}, \phi(s,a)\rangle$. Each robot learned the preferred optimal policies matching each reinforcement function. Table IV shows the evaluations of such preferred optimal policies regarding the corresponding original weight vector ($\mathbf{w}_1$ or $\mathbf{w}_2$). The evaluations in Table II are the reference to maximal performance, where the robots considered the same reinforcement function of the evaluator ($r(s,a) = \langle \mathbf{w}_{objective}, \phi(s,a)\rangle$).

TABLE IV

EVALUATION FOR EACH PREFERRED OPTIMAL POLICY RELATED TO EACH LEARNED WEIGHTED VECTOR.

Robots	Walker	Puller	Crawler
$\mathbf{w}_1^{Wlk}$	-0.852	**-0.801**	**-0.764**
$\mathbf{w}_2^{Wlk}$	-0.774	**-0.870**	**-0.793**
$\mathbf{w}_1^{Pll}$	-0.852	-0.794	-0.745
$\mathbf{w}_2^{Pll}$	-0.774	-0.768	-0.681
$\mathbf{w}_1^{Crl}$	-0.852	-0.794	-0.745
$\mathbf{w}_2^{Crl}$	-0.774	**-0.774**	-0.681

The learned reinforcement functions provide the different types of robots with the optimal policy (compare Tables II and IV) with some notable exceptions (in bold in Table IV): the robots Puller and Crawler when using the weights learned by the robot Walker ($\mathbf{w}_1^{Wlk}$ and $\mathbf{w}_2^{Wlk}$), since these resulted of too few queries; and the robot Puller when using the weights learned by the robot Crawler for objective 2 ($\mathbf{w}_2^{Crl}$), because in this case the optimal policy of robot Crawler does not use action room-cross, therefore the robot Crawler needed less queries to converge. In the latter case, the error is less than 1%.

This shows the abstraction property of the learned reinforcement function and the possibility of knowledge transfer among robots. However, the robot's skills must have some similarities so that they reach similar sets W. In our example, that did not happen with the robot Walker.

VI. CONCLUSION

In this paper we introduced the IRLE problem, where an agent acting in an environment has to determine the utility function of an evaluator, so that the agent can satisfy such utility function. The agent has access to relative evaluation of arbitraries behaviours presented by itself. The potential of such method is programming RL agents, when the evaluator cannot describe the task to the agent, but can made relative evaluations of the agent's behaviours. The description of the utility function as reinforcement functions allows abstracting the task from the environment, making it easier the transfer of knowledge among environments.

An algorithm which solves the IRLE problem under a given condition was introduced. The condition is that the evaluator must be coherent with a linear function, i.e., the order assumed by the evaluator for all possible policies can be accomplished by a linear function of the features and also the feature analysed by the evaluator is known and accessible by the agent. Future work concerning IRLE will relax such condition, so that IRLE can be used in a wider range of environments.

The algorithm was tested in a rescue environment where heterogeneous agents can actuate, but all of them have the same multiple objectives. The experiment shows the capacity of knowledge transfer that the IRLE method can present, but also shows some limitations regarding the set W to which the algorithm converges, since it guarantees optimal policy to the learning environment (Table IV), but not a good evaluation function (Table III) for unlimited differences among environments. Additional study must be made to establish a measure of distance between environments so that reinforcement transfer can be made.

Another potential use of IRLE is in non-Markovian environments (POMDPs), where the original reinforcement function can be redefined, so that common Markovian RL techniques can be used to reach better results, despite the non-full observability.

Appendix

Proof of Theorem 1

Proof: By the end of the algorithm, we have that $\mathbf{w}^* \in W$, since $\mathbf{w}^*$ must respect all order constraints, $\mathbf{w}^*$ is also a solution to constraints G. Considering the solution W and the associated polyhedron P_W, there are two possibilities: 1) for all vertices $\mathbf{w}', \mathbf{w}'' \in P_W$, $Eval(\pi_{\mathbf{w}'}) = Eval(\pi_{\mathbf{w}''})$; or 2) there are vertices $\mathbf{w}', \mathbf{w}'' \in P_W$, such that $Eval(\pi_{\mathbf{w}'}) \neq Eval(\pi_{\mathbf{w}''})$.

In the first case, it is also true that every vertex $\mathbf{w} \in P_W$ have the same preferred optimal policy, i.e., for all vertices $\mathbf{w}', \mathbf{w}'' \in P_W$, $\pi_{\mathbf{w}'} = \pi_{\mathbf{w}''}$ and we define $\pi^* \triangleq \pi_{\mathbf{w}'} = \pi_{\mathbf{w}''}$. Consider the feature expectations $\mu' = \mu(\pi_{\mathbf{w}'})$ and $\mu'' = \mu(\pi_{\mathbf{w}''})$, since both of them have the same evaluation and both vertices agree with this (both of them respect the constraints), then both of them must also agree with respect to the preferred optimal policy. As a result we have that $\pi_{\mathbf{w}^*} = \pi^*$, because for all $a \in \mathcal{A}$:

$$\begin{cases} (\mathbf{T}_{\pi^*} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi^*})^{-1}\phi^{\pi^*}\mathbf{w}' \geq \gamma^{-1}(\phi^a - \phi^{\pi^*})\mathbf{w}' \\ (\mathbf{T}_{\pi^*} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi^*})^{-1}\phi^{\pi^*}\mathbf{w}'' \geq \gamma^{-1}(\phi^a - \phi^{\pi^*})\mathbf{w}'' \end{cases}$$
$$\Rightarrow (\mathbf{T}_{\pi^*} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi^*})^{-1}\phi^{\pi^*}\mathbf{w}_\alpha \geq \gamma^{-1}(\phi^a - \phi^{\pi^*})\mathbf{w}_\alpha, \tag{2}$$

where $w_\alpha = (\alpha\mathbf{w}' + (1-\alpha)\mathbf{w}'')$ for $\alpha \in [0,1]$. The last line in (2) can be generalised for a combination of every vertex $\mathbf{w} \in P_W$, which means that any reinforcement function $r(s,a) = \langle \mathbf{w}, \phi(s,a) \rangle$ whose weight vector $\mathbf{w}$ is a convex combination of elements in P_W (set W including $\mathbf{w}^*$) has policy π^* as the preferred optimal policy.

In the second case, we also can conclude the same. Consider two vertexes $\mathbf{w}', \mathbf{w}'' \in P_W$, their associated policies $\pi' \triangleq \pi_{\mathbf{w}'}$ and $\pi'' \triangleq \pi_{\mathbf{w}''}$, and their associated feature expectation $\mu' = \mu(\pi')$ and $\mu'' = \mu(\pi'')$, where without lost of generalisation $Eval(\pi') < Eval(\pi'')$ and $\pi^* \triangleq \pi_{\mathbf{w}''}$. Then we have:

$$Eval(\pi') < Eval(\pi'') \Rightarrow \begin{cases} \langle \mathbf{w}', \mu' \rangle & \leq & \langle \mathbf{w}', \mu'' \rangle \\ \langle \mathbf{w}'', \mu' \rangle & \leq & \langle \mathbf{w}'', \mu'' \rangle \end{cases} . \tag{3}$$

We also have that:

$$\begin{cases} \langle \mathbf{w}', \mu'' \rangle & \leq & \langle \mathbf{w}', \mu' \rangle \\ \langle \mathbf{w}'', \mu' \rangle & \leq & \langle \mathbf{w}'', \mu'' \rangle \end{cases} , \tag{4}$$

because μ' and μ'' are optimal feature expectations respectively to $\mathbf{w}'$ and $\mathbf{w}''$. Putting together (3) and (4) gives:

$$\begin{cases} \langle \mathbf{w}', \mu' \rangle & = & \langle \mathbf{w}', \mu'' \rangle \\ \langle \mathbf{w}'', \mu' \rangle & \leq & \langle \mathbf{w}'', \mu'' \rangle \end{cases} .$$

But if $\langle \mathbf{w}'', \mu' \rangle = \langle \mathbf{w}'', \mu'' \rangle$ both vertices would have the same preferred optimal policy, which is not the case, hence:

$$\langle \mathbf{w}'', \mu' \rangle < \langle \mathbf{w}'', \mu'' \rangle$$

Since $\langle \mathbf{w}', \mu' \rangle = \langle \mathbf{w}', \mu'' \rangle$, the policy used to reach feature expectations μ'' is also optimal regarding $\mathbf{w}'$, i.e., we can say that $(\mathbf{T}_{\pi''} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi''})^{-1}\phi^{\pi''}\mathbf{w}' \geq \gamma^{-1}(\phi^a - \phi^{\pi''})\mathbf{w}'$ and we have for all $a \in \mathcal{A}$:

$$\begin{cases} (\mathbf{T}_{\pi''} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi''})^{-1}\phi^{\pi''}\mathbf{w}' \geq \gamma^{-1}(\phi^a - \phi^{\pi''})\mathbf{w}' \\ (\mathbf{T}_{\pi''} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi''})^{-1}\phi^{\pi''}\mathbf{w}'' \geq \gamma^{-1}(\phi^a - \phi^{\pi''})\mathbf{w}'' \end{cases}$$
$$\Rightarrow (\mathbf{T}_{\pi''} - \mathbf{T}_a)(\mathbf{I} - \gamma\mathbf{T}_{\pi''})^{-1}\phi^{\pi''}\mathbf{w}_\alpha \geq \gamma^{-1}(\phi^a - \phi^{\pi''})\mathbf{w}_\alpha,$$

where $\mathbf{w}_\alpha = (\alpha\mathbf{w}' + (1-\alpha)\mathbf{w}'')$ for $\alpha \in [0,1]$. The last formula indicates that the policy π'' is optimal for any vertex $\mathbf{w}_\alpha$. We also can say that for any $\alpha \in [0,1)$, the preferred optimal policy of $\mathbf{w}_\alpha$ is π''. ■

Acknowledgements

This research was conducted under the CAPES / GRICES Project MultiBot (Grant no. 099/03). Valdinei F. Silva is grateful to FAPESP (proc. 02/13678-0) and CAPES (proc. BEX-3388/04-2).

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[2] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. New York: Wiley, 1976.

[3] U. Chajewska, D. Koller, and R. Parr, "Making rational decisions using adaptive utility elicitation," in *AAAI/IAAI*, 2000, pp. 363–369.

[4] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *In Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[5] S. M. Ross, *Applied probability models with optimization applications*. San Francisco: Holden-Day, 1970.

[6] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, 1989.

[7] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *In Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[8] J. A. Dinsmoor, "The etymology of basic concepts in the experimental analysis of behavior," *Journal of the Experimental Analysis of Behavior*, vol. 82, no. 3, pp. 311–316, 2004.

[9] S. Russell, "Learning agents for uncertain environments (extended abstract)," in *In Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. ACM Press, 1998.

[10] N. Sprague and D. Ballard, "Multiple-goal reinforcement learning with modular sarsa(0)," in *In International Joint Conference on Artificial Intelligence.*, 2003, pp. 1445–1447.

[11] F. A. Melo, M. I. Ribeiro, and P. Lima, "Navigation controllability of a mobile robot population," in *In Proceedings of the RoboCup 2004 Symposium*, Lisbon, Portugal, 2004.