UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

VICENTE NEJAR DE ALMEIDA

Knowledge Transfer in Multi-Objective Multi-Agent Reinforcement Learning via Generalized Policy Improvement

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering

Advisor: Prof. Dr. Ana L. C. Bazzan
Co-advisor: Lucas N. Alegre

Porto Alegre
October 2022

“You insist that there is something a machine cannot do.
If you will tell me precisely what it is that a machine cannot do,
then I can always make a machine which will do just that.”

— JOHN VON NEUMANN

ACKNOWLEDGEMENTS

I would like to thank Professor Ana L. C. Bazzan and my colleague at the Multiagent Systems Lab. at UFRGS, Lucas N. Alegre, for their invaluable feedback, advice and insightful discussions.

ABSTRACT

Even though many real-world problems are inherently distributed and multi-objective, most of the reinforcement learning (RL) literature deals with single agents and single objectives. That being said, some of these problems can be solved using a single-agent single-objective RL solution, by centralizing the learning and decision making, and specifying a preference over objectives before training the agent. However, most of these problems cannot be centralized due to robustness issues, such as the need to avoid a central point of failure and possible communication failures. Also, in many cases, an agent's preferences might change over time, or it's behavior for several different preferences over objectives might have to be analyzed to better understand tradeoffs among objectives. Therefore, a need arises for a way to train multiple agents (so as to tackle the distributed structure of some tasks) so that they can perform well for any given preferences with respect to their objectives. To address this need, this work proposes a multi-objective multi-agent reinforcement learning (MOMARL) method in which agents decentrally build a shared set of policies during training, and then combine these policies using a generalization of policy improvement and policy evaluation (fundamental operations of RL algorithms) to generate effective behaviors for any possible preferences over their objectives, without requiring any additional training. This method is applied to two different environments: a multi-agent extension of a domain commonly used in the related literature, and a complex, inherently distributed and multi-objective problem (traffic signal control in a scenario with both vehicles and pedestrians). The results show that the approach is able to effectively and efficiently generate behaviors for the agents, given any preference among objectives.

Keywords: Reinforcement learning, multi-agent systems, multi-objective decision making, generalized policy improvement.

Transferência de Conhecimento em Aprendizado por Reforço Multiobjetivo Multiagente via *Generalized Policy Improvement*

RESUMO

Embora muitos problemas do mundo real sejam inerentemente distribuídos e multiobjetivos, a maior parte da literatura de aprendizado por reforço (RL) lida com agentes únicos e objetivos únicos. Dito isto, alguns desses problemas podem ser resolvidos usando uma solução de RL com apenas um agente e um objetivo, centralizando o aprendizado e a tomada de decisão e especificando uma preferência sobre os objetivos antes de treinar o agente. No entanto, grande parte desses problemas não podem ser centralizados devido a questões de robustez, como a necessidade de evitar um ponto central de falha e possíveis falhas de comunicação. Além disso, em muitos casos, as preferências de um agente podem mudar ao longo do tempo, ou seu comportamento para várias preferências diferentes sobre os objetivos pode precisar ser analisado para entender melhor as vantagens e desvantagens entre os objetivos. Portanto, surge a necessidade de uma maneira de treinar vários agentes (de modo a lidar com a estrutura distribuída de algumas tarefas) para que eles possam ter um bom desempenho para quaisquer preferências em relação aos seus objetivos. Para atender a essa necessidade, este trabalho propõe um método de aprendizado por reforço multiobjetivo multiagente (MOMARL) no qual os agentes constroem descentralizadamente um conjunto compartilhado de políticas durante o treinamento e, em seguida, combinam essas políticas usando uma generalização de *policy improvement* e *policy evaluation* (operações fundamentais de algoritmos de RL) para gerar comportamentos eficazes para quaisquer preferências possíveis sobre seus objetivos, sem exigir nenhum treinamento adicional. Este método é aplicado a dois ambientes diferentes: uma extensão multiagente de um domínio comumente utilizado na literatura relacionada, e um problema complexo, inerentemente distribuído e multiobjetivo (controle semafórico em um cenário com carros e pedestres). Os resultados mostram que a abordagem é capaz de eficazmente e eficientemente gerar novos comportamentos para os agentes, dada qualquer preferência entre objetivos.

Palavras-chave: Aprendizado por reforço, sistemas multiagente, tomada de decisão multiobjetivo, *generalized policy improvement*.

LIST OF ABBREVIATIONS AND ACRONYMS

CTDE	Centralized Training with Decentralized Execution
DP	Dynamic Programming
GPE	Generalized Policy Evaluation
GPI	Generalized Policy Improvement
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent System
MAUSF	Multi-Agent Universal Successor Feature
MDP	Markov Decision Process
MOMARL	Multi-Objective Multi-Agent Reinforcement Learning
MOMDP	Multi-Objective Markov Decision Process
MORL	Multi-Objective Reinforcement Learning
MOSG	Multi-Objective Stochastic Game
OLS	Optimistic Linear Support
PWLC	Piecewise linear and convex
RL	Reinforcement Learning
SF	Successor Feature
SG	Stochastic Game
TL	Transfer Learning
USF	Universal Successor Feature

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Reinforcement Learning (RL) (SUTTON; BARTO, 2018) is one of the three main paradigms of Machine Learning, besides Supervised Learning and Unsupervised Learning. RL deals with agents that learn by acting in an environment and receiving rewards (numerical signals) that guide them toward selecting better actions. In recent years, RL has been successfully applied to complex tasks, such as healthcare (YU et al., 2021), robotics (KOBER; BAGNELL; PETERS, 2013), game playing (SILVER et al., 2017; MNIH et al., 2015) and combinatiorial optimization (MAZYAVKINA et al., 2021).

RL has also been successfully applied to Multi-Agent Systems (MAS), that is, Multi-Agent Reinforcement Learning (MARL). MAS is a natural way to model problems that are inherently distributed, but adds many challenges to the already complex single-agent RL. In MARL, there are multiple agents interacting in a common environment, which makes the RL task more complex than that regarding single-agent RL because agents' actions are usually highly coupled and agents are trying to adapt to other agents that are also learning. Besides, several convergence guarantees that were true in single-agent RL no longer hold. However, MARL has many benefits, as it enables the agents to exploit the decentralized structure of a task, and in many cases helps to accelerate learning. Also, in many real-world problems, where the control is decentralized, it is not always possible, feasible or desirable to avoid a MARL formulation.

Besides multiple agents, RL has also been extended to deal with multiple objectives, that is, Multi-Objective Reinforcement Learning (MORL). This is of fundamental practical importance, since in most real-world scenarios, several different aspects and goals are important, and therefore most of these scenarios are inherently multi-objective. There are two main ways to encode a MORL problem: using separate reward functions for each objective, or using a single scalarized reward which combines the agent's preferences with the values received for each objective, thus converting a MORL problem into a single-objective RL problem.

There is some disagreement in the literature as for the need for methods that tackle MORL with separate reward functions, since many multi-objective problems can be converted to single-objective problems using scalarization, as mentioned before. However, in (ROIJERS et al., 2013), three situations are presented to clarify the need for having methods that deal with separate reward functions to solve MORL problems. Two of them, the unknown weights scenario and the decision support scenario, are sufficient to motivate

the need for the method presented in this work.

The unknown weights scenario states that there are multi-objective problems in which the agent's preferences among objectives aren't known during the training phase. In this case, a priori scalarization of the reward becomes impossible. The decision support scenario states that there are multi-objective problems in which it is not possible to precisely specify the agent's preferences with respect to its objectives, so a set of possible solutions needs to be determined so that the tradeoffs between objectives become clear. A need arises, therefore, for a way to train an agent so that it can perform sufficiently given any preference among objectives.

This need is tackled in (ALEGRE; BAZZAN; SILVA, 2022), where a method is presented that enables an RL agent with multiple objectives to incrementally construct during the training phase a set of policies that can later be combined to generate optimal policies for any preference among objectives. However, only single-agent scenarios are addressed. So, there still remains a gap to be addressed: multi-objective problems that fit either of the scenarios discussed previously (unknown weights or decision support scenario) and that have a decentralized structure.

To help fill this gap, this work extends the method presented in (ALEGRE; BAZZAN; SILVA, 2022) to multiple agents, which build a shared set of policies in a decentralized way during training, and then combine these policies to create new policies for new preferences with respect to their objectives. The proposed method is applied to two different environments: a multi-agent extension of the Four-Room environment (a domain commonly used in the related literature), and a complex, inherently distributed and multi-objective problem (traffic signal control in a scenario with both vehicles and pedestrians).

The Four-Room domain is an environment made up of four rooms (which are separated by walls) in which the agents move around and collect different types of objects, which relate to different objectives. In the traffic signal control domain, each agent controls the traffic phases in an intersection, and has two objectives: minimize the waiting time of vehicles and minimize the waiting time of pedestrians.

By applying the proposed method to these two environments, it is empirically shown that the approach is able to effectively and efficiently generate new policies for any given preferences over objectives, without requiring additional training.

During the training phase, the agents are iteratively given different preferences over their objectives, and build policies based on these preferences. As mentioned above, these policies are shared among all agents. During the execution phase (after the training

is over), by making use of generalized policy evaluation (GPE) and generalized policy improvement (GPI), which are generalizations of two key operations of RL algorithms (policy evaluation and policy improvement), the agents are able to construct policies for any given preference over objectives.

What is proposed in this work, therefore, is a Transfer Learning (TL) method for Multi-Objective Multi-Agent Reinforcement Learning (MOMARL). This helps fill a gap within the literature, since the vast majority of the RL literature focuses on single-agents with single-objectives, and only a handful of works address MOMARL settings.

The main contributions of this work can be summed up as follows:

- This is one of the first works to address TL in the context of MOMARL.
- This is the first work (to the author's best knowledge) that leverages generalized policy improvement for settings with multiple agents and objectives.
- One of the domains used to evaluate the method is a multi-objective multi-agent traffic signal control environment with traffic controllers optimizing for vehicles and pedestrians, which has rarely been addressed in the traffic signal control literature.

The reader can find a discussion on the main underlying concepts behind this work and a review of the related literature in Chapter 2 and in Chapter 3, respectively. In Chapter 4, the proposed method is presented and explained in details. Chapter 5 presents the experimental settings and discusses the results. Finally, Chapter 6 concludes this work.

2 THEORETICAL BACKGROUND

This chapter presents an overview of the related literature and explains underlying concepts on RL, MORL, MARL, MOMARL and TL. It also provides a brief overview of specific concepts within these fields that are fundamental to the comprehension of the method: Successor Features (SFs), Generalized Policy Evaluation (GPE) and Generalized Policy Improvement (GPI).

2.1 Reinforcement Learning

In RL, an agent learns how to act in an environment interacting and receiving a feedback signal (reward). This reward signal defines the goal of a reinforcement learning problem (SUTTON; BARTO, 2018), and the agent's objective is to maximize this reward.

An RL problem can be formulated as a Markov Decision Process (MDP), which is a mathematical model for sequential decision making. An MDP can be formally defined as a tuple of the form $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a (possibly infinite) set of states, $\mathcal{A}$ is a (possibly infinite) set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition function mapping state transitions caused by actions to probabilities, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1]$ is the discount factor.

At each time step t, an agent observes a state S_t, selects and takes an action A_t, receives a reward R_{t+1} and transitions to state S_{t+1} according to the state transition function.

A policy π is used to define how the agent behaves in the environment. It maps states to probabilities of selecting each possible action in the set of actions. Hence, $\pi(a|s)$ is the probability of taking action a if the current state is s, where $a \in \mathcal{A}$ and $s \in \mathcal{S}$. This work only deals with deterministic policies, that is, policies in which, for every state, one action is chosen with probability 1 (this does not introduce any issues, because an MDP always has at least one deterministic policy that is optimal).

As mentioned, a reward signal defines the goal of an RL problem, and agents aim to maximize the reward received. However, since RL deals with sequential decision making, at time step t, the best decision isn't necessarily to select an action that will probably lead to the greatest next reward, as it may lead the agent to a sequence of states with a total reward smaller than if it had taken a different action at the initial time step considered. Nevertheless, a reward received immediately is probably much better to an

agent than a reward received a long time in the future, even if both rewards have a close value to each other.

Therefore, the true objective of an RL agent is to maximize the present discounted sum of all rewards received by the agent. The discount factor γ is used to discount the impact of rewards received in later time steps, thus determining the present value of a future reward.

The state-value function of a state s under a policy π, which is denoted by $V^{\pi}(s)$, is the expected value of all returns received by an agent that starts in state s and follows the policy π. This function is given by Eq. 2.1.

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s, \pi\right]. \tag{2.1}$$

Also, the action-value function for a policy π, which is the expected return of taking action a in state s and then following the policy π, is denoted by $Q^{\pi}(s, a)$ and given by Eq. 2.2.

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s, A_t = a, \pi\right]. \tag{2.2}$$

As previously stated, the goal of an RL agent is to maximize the discounted reward it will receive, by finding a policy that achieves high returns. A policy π is better than or equal to a policy π' if and only if the state-value function of any state s under policy π is greater than or equal to the state-value function for the same state under policy π'. This relation, shown in Eq. 2.3, induces a complete ordering over policies.

$$\pi \succeq \pi' \iff \forall s, V^{\pi}(s) \geq V^{\pi'}(s). \tag{2.3}$$

If a policy is better than or equal to all others, it is an optimal policy, denoted by π^*. It is important to note that there is always at least one optimal policy. Therefore, solving a problem in RL means finding an optimal policy (or at least a policy that sufficiently approximates an optimal policy).

All optimal policies share the same state-value function (which is the optimal state-value function, denoted v^*) and the same action-value function (which is the optimal action-value function, denoted q^*). If either the optimal state-value function or the optimal action-value function are known, an optimal policy π^* can be defined as in Eq. 2.4.

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) = \arg\max_{\pi} Q^{\pi}(s, \pi(s)). \tag{2.4}$$

RL algorithms are generally categorized as either being model-based or model-free. Model-based algorithms attempt to learn a model of the environment in order to increase sample efficiency through planning, while model-free methods focus on directly learning a policy or value function.

An important subset of RL algorithms are based on Dynamic Programming (DP) (BELLMAN, 1957). These methods convert Bellman equations into update rules (SUTTON; BARTO, 2018), and make use of mathematical properties of MDPs to decrease the complexity of searching for optimal policies. RL methods based on DP rely on two fundamental operations: policy evaluation and policy improvement. Policy evaluation is the computation of the value function of a policy π, and policy improvement refers to finding a policy π' that is better than π.

2.2 Multi-Agent Reinforcement Learning

When there are multiple agents interacting in a common environment, the RL task is inherently more complex than that regarding single-agent RL, because agents' actions are highly coupled and agents are trying to adapt to other agents that are also learning. Besides, several convergence guarantees no longer hold.

In order to formally describe an RL problem, Section 2.1 introduced MDPs. However, these only consider a single-agent scenario. So, in order to model Multi-Agent decision-making problems, MDPs are extended to MAS as Stochastic Games (SG) (SHAPLEY, 1953), which can be formally defined as a tuple $\langle n, \mathcal{S}, \mathcal{A}_{1..n}, \mathcal{T}, \mathcal{R}_{1..n}, \gamma \rangle$, where n is the number of agents, $\mathcal{S}$ is the state space, $\mathcal{A}_i$ is the set of actions of agent i ($\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_n$ is the joint action space), $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function, $\mathcal{R}_i$ is the reward function of agent i, and $\gamma \in [0, 1]$ is the discount factor.

Differently than single-agent RL, in MARL each reward received by an agent depends not only on its own actions, but on the actions of other agents. Because of this, there is not a clear concept defining an optimal policy as in single-agent RL (SILVA; COSTA, 2019). Thus, as previously mentioned, several convergence guarantees that are true in single-agent RL no longer hold.

Not only do other agent's actions influence the reward received by each agent, but

in fact agents may have opposing objectives, and the action of one agent might negatively impact the rewards received by another. Generally, the nature of a multi-agent task is classified as being either cooperative, competitive or mixed (BUŞONIU; BABUSKA; SCHUTTER, 2008). In cooperative settings, all agents have the same reward function, and agents work together to distributively optimize the performance of a larger system. In competitive settings, agents have opposite goals, and a win for one agent implies a loss for another. In mixed settings, there are no such restrictions on reward function definitions.

An important aspect of MARL is the extent of decentralization of planning and learning. All agents can be trained in a centralized manner using, for instance, a global reward. However, the global reward doesn't address the credit assignment problem (MINSKY, 1961), because agents have difficulty understanding and distinguishing their own contribution to the larger system (RĂDULESCU et al., 2020). Another option is full decentralization, that is, agents learn and act individually, in a fully decentralized manner. A third option is a middle ground between centralized and fully decentralized, in which training and learning is improved with strategies for sharing and knowledge transfer, but actions are taken by agents individually, in a decentralized manner.

2.3 Transfer Learning in MARL

There are a few surveys on TL for RL, such as (TAYLOR; STONE, 2009) and (LAZARIC, 2012). Even though they present many concepts that are fundamental to this work, they only focus on single-agent and single-objective RL. Of particular interest to this work is (SILVA; COSTA, 2019), which is a survey on TL for MARL. However, in this survey, focus is only given to single-objective techniques.

The basic motivation behind transfer learning is improving the performance of a target learner or target domain by reusing existing knowledge that is transferred from a different source domain (ZHUANG et al., 2020). TL has been widely applied to many different areas of machine learning, but as this paper is focused on applying it to RL (more particularly MOMARL), this Section only presents a general overview of TL in the context of Multi-Agent Reinforcement Learning (considerations regarding multiple objectives are mostly omitted from this Section, as the existence of multiple objectives is orthogonal to the concept of transfer learning, and very few works address TL for MOMARL).

In a MARL setting, each agent learns via an algorithm A_{agent}, which is a mapping

shown in Eq. 2.5, where K is a knowledge space and H is a space of hypotheses, more specifically the space of all possible policies (LAZARIC, 2012; SILVA; COSTA, 2019).

$$A_{agent} : K \rightarrow H. \tag{2.5}$$

Transfer learning in MARL aims at improving the performance of the agent's current task, which is called the target task. The knowledge space K is composed of knowledge that the agent acquired from the target task, denoted K^{target}, prior knowledge derived from the solution of previous tasks (called source tasks), denoted K^{source}, and knowledge received from other agents, denoted K^{agents}. Therefore, K can be defined as in Eq. 2.6 (SILVA; COSTA, 2019).

$$K = K^{target} \cup K^{source} \cup K^{agents}. \tag{2.6}$$

It is important to note that not all knowledge transfer is beneficial. In many cases, transfer might reduce the learning performance of agents, bringing a negative effect to the system. This is called negative transfer, and is a challenging problem in TL.

In (SILVA; COSTA, 2019), a taxonomy for classifying TL methods in MAS is proposed. This taxonomy is of great interest to the present discussion because it is able to classify all of the existing publications in the literature so far. This taxonomy classifies transfer methods as either being Intra-Agent Transfer or Inter-Agent Transfer. In Intra-Agent Transfer, agents reuse knowledge from different tasks and domains in order to improve the target task. In Inter-Agent Transfer, agents receive knowledge from other agents (which may be different from each other) via communication and reuse this knowledge to improve the target task.

There are many important matters that a TL method for MARL has to deal with. Which agent initiates the transfer or sharing of knowledge? How are source tasks selected? How to determine the aspects in which a source task is similar to a target task, in order to enable transfer? What is the object of the transfer (for instance, experiences, value functions, policies, action advice, heuristics, among many others)? How much knowledge can be transferred, that is, is there a budget and, if so, how to optimize according to its restrictions? These questions and many more present very interesting challenges to TL, and there are many different ways to deal with them. Section 3.1 presents an overview of many TL methods for MARL, which address these questions in varying manners.

2.4 Multi-Objective Reinforcement Learning

Most of the RL literature and algorithms deal with agents that have a single objective. However, approaches that only take into account a single objective are generally too shallow when compared to most real-world decision problems, which are inherently multi-objective.

One common approach to enable agents to deal with multiple objectives is scalarization to combine all of the objectives in a single scalar reward function via a linear combination. Unfortunately, this rather simple solution has many drawbacks, as for instance it reduces drastically the explainability of the model (as it will be hard to distinguish which particular objectives prompted a specific behavior), is unable to handle many different types of preferences that might be required by a specific problem, and will require that the agents be retrained if preferences between objectives change (HAYES et al., 2021).

With these drawbacks in mind, it becomes clear why, in order to deal with multiple objectives, an explicitly multi-objective approach is preferable over a scalarized reward approach (which is equivalent to a single-objective approach). So, instead of using a scalar reward, a vector-valued reward function will be used.

As was the case with MARL, in Section 2.2, the definitions of MDPs also need to be extended for the sake of the current discussion. To formally describe a multi-objective decision-making problem, MDPs are extended to multiple objectives as a Multi-Objective Markov Decision Process (MOMDP), which can be formally defined as a tuple of the form $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$, where $\mathcal{S}$ is a (possibly infinite) set of states, $\mathcal{A}$ is a (possibly infinite) set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition function mapping state transitions caused by actions to probabilities, $\boldsymbol{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^d$ is a vector-valued reward function that specifies the immediate reward for each of the considered d objectives, and $\gamma \in [0, 1]$ is the discount factor.

Since rewards are now vector-valued, state-value functions and action-value functions are now vector-valued as well, that is, $\boldsymbol{V}^\pi \in \mathbb{R}^d$ and $\boldsymbol{Q}^\pi \in \mathbb{R}^d$. For problems with single objectives, a policy π is either better, equal or worse than policy π', as value functions completely order all policies. This is not necessarily the case in multi-objective decision-making problems, as value functions under policy π, when compared to value functions under policy π', might have a greater value for some objectives, but a smaller value for others.

Because there is no complete ordering over policies, as mentioned above, in MOMDPs

value functions only offer a partial ordering over the policy space. So, in this setting, there can exist several possibly optimal value vectors $\boldsymbol{V}$ and $\boldsymbol{Q}$. In order to deal with this, a few sets of possibly optimal policies and value vectors need to be defined.

Before the sets mentioned above are defined, the utility function (or scalarization function) $u : \mathbb{R}^d \rightarrow \mathbb{R}$ needs to be introduced. This function, shown in Eq. 2.7, maps a multi-objective state-value vector under a policy π to a scalar value. This function commonly takes the form of a linear combination.

$$V_u^{\pi}(s) = u(\boldsymbol{V}^{\pi}(s)). \tag{2.7}$$

The first set that needs to be defined is the undominated set, $\mathrm{U}(\Pi)$, shown in Eq. 2.8. Π is the set of all possible policies, and $\mathrm{U}(\Pi)$ is a subset of Π that contains all policies that are optimal for any given utility function. In other words, this set contains all policies that satisfy any possible preference among objectives.

$$U(\Pi) = \{\pi \in \Pi \mid (\exists u)\,(\forall \pi' \in \Pi)\,[V_u^{\pi} \geq V_u^{\pi'}]\}. \tag{2.8}$$

The undominated set solves any MOMDP, but it may contain policies in excess, that are redundant. With this is mind, the next set that needs to be defined is the coverage set, $\mathrm{CS}(\Pi)$ (BECKER et al., 2003), shown in Eq. 2.9. This set contains at least one optimal policy for any utility function, instead of containing all optimal policies for any utility function, which is the undominated set, defined above. This coverage set is, therefore, a subset of $\mathrm{U}(\Pi)$, that is, $\mathrm{CS}(\Pi) \subseteq \mathrm{U}(\Pi)$. So, the undominated set is also a coverage set (since the undominated set contains all policies that are optimal for any given utility function, it contains at least one policy that is optimal for any utility function, thus it is a coverage set). It is important to note that $\mathrm{U}(\Pi)$ is unique, but $\mathrm{CS}(\Pi)$ is not necessarily unique.

$$CS(\Pi) = \{\pi \in \Pi \mid (\forall u)\,(\exists \pi \in \Pi)\,(\forall \pi' \in \Pi)\,[V_u^{\pi} \geq V_u^{\pi'}]\}. \tag{2.9}$$

If the coverage set $\mathrm{CS}(\Pi)$ is known, an optimal policy for any given utility function u can be determined via Eq. 2.10.

$$\pi^* = \underset{\pi \in CS(\Pi)}{\arg\max}\, V_u^{\pi}(s). \tag{2.10}$$

According to a MOMDP problem taxonomy proposed by (ROIJERS et al., 2013),

the three major factors that classify the nature of an MOMDP problem are: (i) whether the goal is to find one or multiple policies; (ii) whether the utility function is a linear function or, more generally, any monotonically increasing function; (iii) whether stochastic policies (instead of only deterministic policies) are allowed. For the sake of the present discussion, more definitions will be given regarding factor (ii).

A linear utility function, shown in Eq. 2.11, is the inner product of a vector of weights $\boldsymbol{w}$ and a value vector $\boldsymbol{V}^{\pi}$.

$$V_u^{\pi} = \boldsymbol{w} \cdot \boldsymbol{V}^{\pi}. \tag{2.11}$$

If the utility function is linear, and $\boldsymbol{w}$ adheres to the simplex constraints (that is, $\forall i \;\; w_i \geq 0$ and $\sum_i w_i = 1$) (ROIJERS, 2016), the undominated set $U(\Pi)$ is a convex hull CH(Π), defined in Eq. 2.12. Since this convex hull is an undominated set, it may contain policies in excess (as explained previously). So, a coverage set for the convex hull can be defined. Because the utility function is linear, this coverage set is called convex coverage set CCS(Π), shown in Eq. 2.13.

$$CH(\Pi) = \{\pi \in \Pi \mid (\exists \boldsymbol{w})\,(\forall \pi' \in \Pi)\,[\boldsymbol{w} \cdot \boldsymbol{V}^{\pi} \geq \boldsymbol{w} \cdot \boldsymbol{V}^{\pi'}]\}. \tag{2.12}$$

$$CCS(\Pi) = \{\pi \in \Pi \mid (\forall \boldsymbol{w})\,(\exists \pi \in \Pi)\,(\forall \pi' \in \Pi)\,[\boldsymbol{w} \cdot \boldsymbol{V}^{\pi} \geq \boldsymbol{w} \cdot \boldsymbol{V}^{\pi'}]\}. \tag{2.13}$$

In the more general case, where the utility function is simply monotonically increasing, a set of viable policies commonly used in the literature is the Pareto front PF(Π). A policy π Pareto-dominates another policy π' if its value is equal or greater for every objective, and strictly greater for at least one objective, according to Eq. 2.14. The Pareto front set is defined in Eq. 2.15. It is important to note that the Pareto front set is not necessarily the undominated set, as the Pareto front may be larger than the undominated set, depending on the utility function.

$$\boldsymbol{V}^{\pi} \succ_P \boldsymbol{V}^{\pi'} \iff \forall i, V_i^{\pi} \geq V_i^{\pi'} \wedge \exists i, V_i^{\pi} > V_i^{\pi'}. \tag{2.14}$$

$$PF(\Pi) = \{\pi \in \Pi \mid (\nexists \pi' \in \Pi)\,[\boldsymbol{V}^{\pi'} \succ_P \boldsymbol{V}^{\pi}]\}. \tag{2.15}$$

A final definition required for this Section is the Pareto coverage set PCS(Π),

which is shown in Eq. 2.16.

$$PCS(\Pi) = \{\pi \in \Pi \mid (\exists \pi \in \Pi)\,(\forall \pi' \in \Pi)\,[\boldsymbol{V}^{\pi} \succ_P \boldsymbol{V}^{\pi'} \vee \boldsymbol{V}^{\pi} = \boldsymbol{V}^{\pi'}]\}. \tag{2.16}$$

All of the sets defined in this Section are of extreme importance to multi-objective problems, because MORL algorithms seek to compute the policies of these sets in order to solve multi-objective decision-making problems.

2.5 Successor Features and Generalized Policy Improvement

In (BARRETO et al., 2020), it is argued that many complex problems tackled by RL can be broken down into multiple tasks, encoded by different reward functions. A reward function $r_{\boldsymbol{w}}$ for a task is defined as in Eq. 2.17, where $\boldsymbol{\phi}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ is an arbitrary function representing the features of the environment and $\boldsymbol{w} \in \mathbb{R}^d$ is a vector of weights.

$$r_{\boldsymbol{w}}(s, a, s') = \boldsymbol{w} \cdot \boldsymbol{\phi}(s, a, s'). \tag{2.17}$$

With this reward definition, given a policy π, the action-value function shown in Eq. 2.2 can be rewritten as in Eq. 2.18, where $\boldsymbol{\psi}^{\pi}(s, a)$ are successor features (SFs) (BARRETO et al., 2017). It is important to note that SFs satisfy a Bellman equation, so they can be computed using conventional RL algorithms.

$$\begin{aligned} Q^{\pi}_{\boldsymbol{w}}(s, a) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{w} \cdot \boldsymbol{\phi}_{t+k} \middle| S_t = s, A_t = a, \pi\right] \\ &= \boldsymbol{w} \cdot \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{\phi}_{t+k} \middle| S_t = s, A_t = a, \pi\right] \\ &= \boldsymbol{w} \cdot \boldsymbol{\psi}^{\pi}(s, a). \end{aligned} \tag{2.18}$$

The definitions given above, combined with a generalization of two fundamental operations of DP methods for RL (policy evaluation and policy improvement, mentioned in Section 2.1), provide a framework for knowledge transfer among tasks.

Suppose a RL agent, using an arbitrary RL method, has, through training, learned

policies for k tasks (given by k different instances of $\boldsymbol{w}$). The agent thus knows a set of policies $\Pi = \{\pi_i\}_{i=1}^{k}$ and a set of corresponding SFs $\Psi = \{\boldsymbol{\psi}^{\pi_i}\}_{i=1}^{k}$.

Using Eq. 2.18, it is possible to compute the value-function of a policy π on the k tasks. This is called generalized policy evaluation (GPE), and is a generalization of policy evaluation. Also, given a new task, generalized policy improvement (GPI) refers to finding a policy that is better than all of the policies in Π. This is a generalization of policy improvement. Note that GPE and GPI are simply policy evaluation and policy improvement over multiple policies and tasks, so if $k = 1$, GPE and GPI are equivalent to the standard DP operations.

The reader would be correct in noticing that there is a strong similarity between the SF framework and MORL problems encoded by separate reward functions. In fact, in (ALEGRE; BAZZAN; SILVA, 2022) it is shown that the transfer problem addressed by SFs is equivalent to the multi-objective optimization in MORL. That work demonstrates this by showing that it is possible to map any SF problem to a MORL problem by converting each dimension of $\boldsymbol{\phi}$ into an objective. That is, an MOMDP is created with $\boldsymbol{\mathcal{R}}(s, a, s') = \boldsymbol{\phi}(s, a, s')$.

Since there exists this equivalence between SF problems and MORL problems, it is possible to apply GPI to MORL. Furthermore, (ALEGRE; BAZZAN; SILVA, 2022) also shows that, if a convex coverage set is learned, performing GPI on this set enables an RL agent to learn an optimal policy for any weight vector $\boldsymbol{w}$.

2.6 Multi-Objective Multi-Agent Reinforcement Learning

In Section 2.1, a general explanation of Reinforcement Learning is provided. In Section 2.2, RL is extended to Multi-Agent Systems. In Section 2.4, RL is extended to multiple objectives. In this section, a brief overview of Multi-Objective Multi-Agent Reinforcement Learning, combining concepts explained in all of these previous sections, is presented. MOMARL decision-making problems are particularly difficult, as they combine many of the challenges of MARL settings with challenges of MORL settings.

A MOMARL problem involves multiple agents, each of them with multiple objectives, interacting in a common environment. To formally describe a MOMARL problem, Stochastic Games are extended to multiple objectives as a Multi-Objective Stochastic Game (MOSG) (RĂDULESCU et al., 2020), which can be formally defined as a tuple $\langle n, \mathcal{S}, \mathcal{A}_{1..n}, \mathcal{T}, \boldsymbol{\mathcal{R}_{1..n}}, \gamma \rangle$, where n is the number of agents, $\mathcal{S}$ is the state space, $\mathcal{A}_i$ is the

set of actions of agent i ($\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_n$ is the joint action space), $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function, $\boldsymbol{\mathcal{R}_i} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}^d$ is the vector-valued reward function of agent i that specifies the immediate reward for each of the considered d objectives, where $d \geq 2$ objectives, and $\gamma \in [0, 1]$ is the discount factor.

The reader might notice that the formalization presented assumes that all agents have the same types of objectives. This definition is sufficient for the present work, as the proposed method deals with agents that are homogeneous (one agent is exactly the same as any other agent in the system). However, agents can have different preferences over their objectives.

Each agent i has its own utility function u_i. Also, as is the case in single-agent MORL, state-value functions and action-value functions are also vector-valued, that is, $\boldsymbol{V}^{\pi_i} \in \mathbb{R}^d$ and $\boldsymbol{Q}^{\pi_i} \in \mathbb{R}^d$, where π_i is a policy of agent i.

In (RĂDULESCU et al., 2020), a taxonomy of multi-objective multi-agent decision-making settings is presented. This taxonomy is very useful not only to classify problems but to offer a broad view of problems in these settings. The taxonomy is based on reward functions and utility functions. Multi-objective multi-agent decision-making settings are considered as either having a team reward (agents receive the same reward) or individual rewards (agents receive their own distinct reward). For settings with team rewards, the problems are further classified as having team utility, social choice utility or individual utility, and in settings with individual rewards, problems either have social choice utility or individual utility. In team utility settings, all agents have the same interests. In social choice utility settings, agents work to improve the overall social welfare of all agents. And in individual utility settings, agents have their own particular interests.

3 RELATED WORK

This chapter presents a review of the related literature. Section 3.1 presents an overview of TL methods for MARL and MOMARL. Section 3.2 and Section 3.3 address relevant works that have tackled SFs and GPI in MORL and MARL settings, respectively. Finally, Section 3.4 discusses applications of RL for traffic signal control (which is one of the domains used in Chapter 5).

The works discussed in Section 3.2 and Section 3.3 either tackled SFs and, more specifically, GPI, in MORL settings or MARL settings. To the author's best knowledge, no relevant work has addressed GPI in MOMARL settings. Also, in Section 3.1, it is mentioned that only one relevant work tackled TL in MOMARL settings. To help fill this gap in the literature, this work proposes a MOMARL method that uses GPI to combine previously learned policies in order to compute policies for new preferences of objectives, being one of the few works to address any TL method for MOMARL, and the first work to address SFs and GPI in a MOMARL setting.

3.1 Broad overview of TL methods for MARL and MOMARL

In order to present a brief overview of different TL methods for MARL, related works that address, in different ways, many of the questions raised at the end of Section 2.3 are discussed. An extensive overview of every relevant work that has dealt with TL for MARL is beyond the scope of the present discussion. However, a few relevant works that apply varied TL methods to MARL have been selected, and are compared in Table 3.1. These works are varied enough that they offer a broad perspective into many of the different ways that knowledge can be transferred in MARL.

Table 3.1 – Selected works that address TL for MARL

Study	MAS Interaction	Transfer Type	What is Transferred
(BARRETT; STONE, 2015)	Mixed	Intra-Agent	Policies
(BIANCHI et al., 2014)	Competitive	Inter-Agent	Heuristics
(LE et al., 2017)	Cooperative	Inter-Agent	Experiences
(LONG et al., 2020)	Mixed	Intra-Agent	Parameters
(PROPER; TADEPALLI, 2009)	Mixed	Intra-Agent	Value Function
(SHI et al., 2021)	Mixed	Intra-Agent	Features
(TORREY; TAYLOR, 2013)	Cooperative	Inter-Agent	Action Advice

In (BARRETT; STONE, 2015), an agent uses knowledge acquired from cooperat-

ing with different teams to better cooperate with new teams, by learning an approximate model of teammates and different policies that are better suited for each type of teammate.

In (BIANCHI et al., 2014), a heuristic function to suggest actions is used in competitive MARL. This work in particular is interesting because this heuristic can not only be learned from the current domain, but it can also be learned from different domains.

In (LE et al., 2017), imitation learning is used for teaching coordination in MARL. Agents observe other agents, that is, receive demonstration data (which are the experiences acquired by the agents that are being observed), and attempt to approximate the implicit coordination in the data. They do this by attempting to understand the roles of the observed agents, and learn a role assignment function that maps agents to different roles (and thus different policies).

The work (LONG et al., 2020) deals with Curriculum Learning (where agents learn to reuse knowledge across increasingly more difficult tasks) for MARL by progressively increasing the amount of agents in multiple stages, keeping different sets of agents at each stage, and selectively promoting the best performing sets of agents, in an evolutionary manner, where parameters are implicitly mutated. This allows MARL to be efficiently scaled to bigger populations, by starting with a small population and increasing it at each stage, thus reusing knowledge acquired from interacting with smaller populations in order to better interact with larger populations.

In (PROPER; TADEPALLI, 2009), a task decomposition approach is used that breaks a problem in two steps: task assignment and task execution. This decomposition, which allows for better generalization of a problem, is combined with TL, by transferring value function approximators from smaller domains to larger domains. Essentially, this work is important because it allows for knowledge gathered from simple tasks to be reused in more complex tasks.

The work (SHI et al., 2021) is of particular interest because of its versatility, as it proposes a method that can transfer knowledge not only among different tasks, but also among heterogeneous agents. Policies from source domains are clustered based on their value functions, using a Gaussian mixture model, and the features of several similar policies are transferred to the same agent.

In (TORREY; TAYLOR, 2013), a teacher-student framework is presented in which teachers provide action advice to students (that is, suggest actions to be taken), but are limited by a budget, which means they can only provide a limited amount of advice, and thus need to optimize their usage of the advice budget. Even though this paper isn't

applied to MARL specifically, it has been selected because it can easily be extended to multiple agents (which is even mentioned in its related work section), and addresses a very important concept for TL: transfer on a budget.

The reader might have noticed that, although this work focuses on TL for RL with multiple agents and multiple objectives, no works that address all these areas at the same time are mentioned. This is because, as mentioned before, in spite of the existence of many real-world problems that are multi-objective and inherently distributed, and that could greatly benefit from research in TL techniques for MOMARL, there is a lack of works that investigate and apply TL in the context of MOMARL. In fact, to the author's best knowledge, (TAYLOR et al., 2014) is the only relevant work that addressed TL in the context of RL with multiple agents and multiple objectives.

3.2 SFs and GPI in MORL

SFs and GPI are presented in (BARRETO et al., 2017) as a generalization of the concept of successor representation (DAYAN, 1993) and policy improvement, respectively. This provides a framework for reusing knowledge across RL tasks, which is further elaborated in (BARRETO et al., 2020).

By formally demonstrating the equivalence between the SF framework and MORL, (ALEGRE; BAZZAN; SILVA, 2022) is able to make use of GPI within the context of MORL, thus proposing a method that is able to combine previously learned policies in order to compute an optimal policy for any preference of objectives. However, this work only deals with single-agent scenarios.

Leveraging GPI within MORL is achieved by first using SFOLS, a SFs based extension of Optimistic Linear Support (OLS) (ROIJERS, 2016) to assign different weights to the RL agent, so that it iteratively computes a CCS. Then, once a CCS is learned, applying GPI to this set enables the agent to generate an optimal policy for any possible preference of objectives.

Most of the works discussed in this section use as one of their experimental settings the Four-Room domain, which is an environment made up of four rooms (separated by walls) in which a single agent moves around and collects different types of objects, which relate to different objectives. Because this environment is such a common benchmark in the literature, this work extends the Four-Room environment to multiple agents, and uses this domain as one of the experimental settings.

3.3 SFs and GPI in MARL

Besides the works discussed in Section 3.2, only a handful of works have mentioned SFs in the context of MORL, and always briefly so, such as (HAYES et al., 2021), which briefly states that SFs are a subclass of multi-objective decision making problems, and (ABELS et al., 2019), which also briefly mentions that SFs and MORL are analogous. However, there have been a few works that have extended the concepts of SFs and GPI to multi-agent scenarios.

Decomposing a problem using a multi-agent solution generally introduces a source of non-stationarity. One approach that has been used to tackle this is the centralized training with decentralized execution (CTDE) framework, in which the agents learn their policies together, but execute them independently.

This CTDE method is used in the universal value exploration (UNeVEn) algorithm, presented in (GUPTA et al., 2021), which extends universal successor features (USFs) (BORSA et al., 2019) to multi-agent universal successor features (MAUSFs), and combines this with GPI to improve the joint exploration of agents during training. Furthermore, CTDE is also used in (LIU et al., 2022), which presents an approach that makes use of SFs and GPI, and enables knowledge transfer among tasks in MARL settings.

Another relevant work that tackles SFs using CTDE is (KIM et al., 2022). They use SFs to isolate each agent's impact on the performance of the entire system, which allows the method to create individual utility functions for each agent tailored to stabilize their training.

It is important to note that, since the works addressing SFs and GPI in MARL settings do so using the CTDE framework, they all suffer from scalability issues, for obvious reasons. The more agents in the environment, the more difficult it is to train these agents in a centralized manner. The method proposed in this work has no such scalability issues, as both execution and training occur in a decentralized fashion.

3.4 RL for Traffic Signal Control

Traffic signal controllers seek to determine a split of green times among various phases at an intersection. A phase is defined as a group of non-conflicting movements (e.g., flow in two opposite traffic directions) that can have a green light at the same time without conflict. There are many different ways that traffic controllers can go about mak-

ing decisions, and (ROESS; PRASSAS; MCSHANE, 2004) provides an overview of several of them.

Th most basic form of traffic control is based on fixed times, where the split of green times among the phases is computed using historical data on traffic flow, if available. However, this approach is unable to adapt to changes in traffic demand, which may lead to an increase in waiting times. To mitigate this issue, an adaptive approach, such as RL, can be used to determine the split of green times using some measure of performance (for instance, accumulated waiting time).

The RL literature for traffic signal control is very extensive and diverse, and a detailed overview of different techniques is beyond the scope of this work. Thus, the reader is directed to surveys such as (BAZZAN, 2009; YAU et al., 2017; WEI et al., 2020; NOAEEN et al., 2021). Of particular interest to this work are multi-objective RL approaches to traffic signal control.

Besides reducing the waiting time of vehicles, traffic signal controllers can also have different objectives, such as reducing queue lengths (DUAN; LI; ZHANG, 2010), reducing the environmental impact of traffic by minimzing fuel consumption (KHAMIS; GOMAA, 2012) and reducing the waiting time of pedestrians (EGEA; CONNAUGHTON, 2020). However, it must be emphasized that the literature on MORL for traffic signal control is small, and besides the work just mentioned, there are very few MORL works that address pedestrians. This is another literature gap that this work helps to fill.

4 POLICY TRANSFER IN MOMARL USING GPI

This chapter describes the proposed method in details. However, before the method is explained, some preliminary conditions need to be set forth. First, all agents must be homogeneous (that is, have the same sensors, possible actions and objectives), but have individual weights. Also, each agent's weights must adhere to the simplex constraints (induce a convex combination of the objectives), that is, $\forall i \;\; w_i \geq 0$ and $\sum_i w_i = 1$, and their utility functions must be the inner product between the objectives and the weights.

During the training phase, the method uses a multi-agent extension of the SFOLS algorithm (ALEGRE; BAZZAN; SILVA, 2022) to distributively solve scalarized versions of the multi-objective problem, by assigning different weights to each agent. Each agent computes a policy that is stored in a set of policies shared by all agents, until they have computed a CCS (or at least an incomplete, but good enough, CCS, that is, an ϵ-CCS). This process is explained in Section 4.1.

Once the agents have decentrally computed a CCS, the training phase is over. Then, during the execution phase, each agent is able to compute an effective policy for any possible instance of $\boldsymbol{w}$, by applying GPI (ALEGRE; BAZZAN; SILVA, 2022) to the shared set. This is explained in Section 4.2.

4.1 Decentrally computing a shared set of policies

During training, the agents seek to distributively compute a CCS. In order to do so, they iteratively receive different weights, and for each of these weights learn a policy for a scalarized version of the multi-objective task. That is, every time an agent receives a weight, it uses an arbitrary single-objective RL algorithm to determine a policy for that weight (as explained previously, the weight is used to scalarize the reward functions, thus producing a scalar reward, which converts the multi-objective problem into a single-objective problem). The policies learned by the agents are stored in a shared set of policies, Π.

In a nutshell, each agent receives a weight vector $\boldsymbol{w}$, learns a policy for that weight, stores that policy in Π, and then receives a new weight vector and learns a new policy, repeating the process until Π corresponds to a CCS. The key, then, is to know which weights to assign to the agents, and when to stop (that is, how to know when Π is a CCS). This is determined by using the OLS algorithm, which is an extension of Cheng's linear

support algorithm (originally intended for use on partially observable MDPs) (CHENG, 1988).

Let Π be a set of policies shared among all agents and $Q_{\boldsymbol{w}}$ a priority queue of weights. At each iteration of the algorithm, the weights in $Q_{\boldsymbol{w}}$ with greater priority are popped off the queue and assigned to the agents. If there are more agents than weights in $Q_{\boldsymbol{w}}$, random weights are assigned to the remaining agents. After each iteration, new weights are added, until Π corresponds to a CCS. If, at the end of an iteration, $Q_{\boldsymbol{w}}$ is empty, that means that Π is a CCS, and the algorithm terminates.

The first step is to add to $Q_{\boldsymbol{w}}$ all weights in the extremum of the weight simplex (that is, all weights that have one component equal to 1 and all others equal to zero), and assign to these weights an infinite priority (infinity of course is relative, but this just means assigning such a large number to the priorities of these weights in the extremum of the weight simplex that they are the first weights to be popped off the queue and assigned to the agents).

After this initialization of the priority queue, the algorithm's main loop commences. The weights in $Q_{\boldsymbol{w}}$ are popped off according to their priorities and assigned to the agents, which then proceed to learn policies for the scalarized version of their multi-objective task (using their respective weights to scalarize the task). Once the agents finish learning their respective policies, these policies are stored in Π, and the algorithm evaluates the multi-objective value of the policies learned.

Regular policy evaluation for single-objective RL can be used to determine the value of a policy for each of its objectives. This produces the multi-objective value of that policy. Having determined the multi-objective value of the policies learned, the algorithm determines the *corner weights*, which, as per Theorem 1, are the instances of $\boldsymbol{w}$ that offer the greatest possible improvement (that is, the instances of $\boldsymbol{w}$ whose optimal scalarized value functions are farthest from the current greatest value function for that $\boldsymbol{w}$) and inserts them in $Q_{\boldsymbol{w}}$.

Theorem 1 *(CHENG, 1988) Let $\mathcal{W}$ be the set of all possible reward vectors, let CCS be a convex coverage set, and let Π be a set of deterministic policies. The maximum value of*

$$\max_{\boldsymbol{w}\in\mathcal{W},\pi\in CCS} \min_{\pi'\in\Pi} \boldsymbol{w}\cdot\boldsymbol{V}^{\pi} - \boldsymbol{w}\cdot\boldsymbol{V}^{\pi'} \tag{4.1}$$

is at one of the corner weights of

$$V^{CB}_{\boldsymbol{w}} = \max_{\pi\in\Pi} \boldsymbol{w}\cdot\boldsymbol{V}^{\pi} \tag{4.2}$$

Let $V_{\boldsymbol{w}}^{CB}$, given by Eq. 4.2, be the current best value function. To determine the *corner weights*, the algorithm exploits the fact that $V_{\boldsymbol{w}}^{CB}$ is a piecewise linear and convex (PWLC) (ROIJERS, 2016; CHENG, 1988) function. Thus, the *corner weights* are the points in which $V_{\boldsymbol{w}}^{CB}$ changes slope.

Each *corner weight* $\boldsymbol{w}_c$ is inserted into $Q_{\boldsymbol{w}}$ with priority $\Delta(\boldsymbol{w}_c)$, which is an optimistic estimate of the greatest possible improvement caused by learning a policy with respect to $\boldsymbol{w}_c$. This priority is given by Eq. 4.3, where $\overline{V}_{\boldsymbol{w}}^{*}$ is an optimistic upper-bound for the optimal value function $V_{\boldsymbol{w}}^{*}$.

$$\Delta(\boldsymbol{w}_c) = \overline{V}_{\boldsymbol{w}}^{*} - V_{\boldsymbol{w}}^{CB} \tag{4.3}$$

After inserting the *corner weights* in $Q_{\boldsymbol{w}}$, the algorithm's main loop (pop off weights with greatest priority from $Q_{\boldsymbol{w}}$, assign them to the agents, learn policies, add *corner weights* to $Q_{\boldsymbol{w}}$) is repeated, until $Q_{\boldsymbol{w}}$ is empty, which indicates that there are no more *corner weights*, and therefore Π forms a CCS.

A pseudocode for the algorithm is shown in Algorithm 1. The agents can use any temporal difference learning algorithm to learn policies for the weights they receive, since they use these weights to scalarize the task, thus converting a multi-objective RL problem into a single-objective RL problem.

Let us propose an example to better illustrate the method. For the sake of simplicity, suppose a single agent (extending this example to multiple agents is trivial, as the only difference would be that more than one policy would be learned at each iteration) with three actions (A, B and C) and two objectives. Suppose there are 5 states: s_0, s_1, s_2, s_3 and s_4. s_0 is the initial state and s_4 is the terminal state.

For this simple example, at each time step, the agent always transitions to the same states, regardless of the action selected. Choosing different actions in each state only changes the reward received. The MOMDP for this example is shown in Figure 4.1. A run of the OLS algorithm for the example is shown graphically in Figure 4.2.

Since this example has only two objectives, and the weights adhere to the simplex constraints, only one of the dimensions of $\boldsymbol{w}$ is sufficient to express all possible instances of weights ($w_0 = 1 - w_1$).

Algorithm 1: Multi-Agent SFOLS (MA-SFOLS)

$\Psi \leftarrow \{\}$; $\Pi \leftarrow \{\}$; $W \leftarrow \{\}$; $Q_{\boldsymbol{w}} \leftarrow \{\}$;
foreach *weight* $\boldsymbol{w}_e$ *in extremum of weight simplex* **do**
    Insert ($\boldsymbol{w}_e$, ∞) into $Q_{\boldsymbol{w}}$;
end
while $Q_{\boldsymbol{w}}$ *is not empty* **do**
    foreach *agent* $\boldsymbol{a}$ **do**
        if $Q_{\boldsymbol{w}}$ *is not empty* **then**
            $\boldsymbol{w_a} \leftarrow Q_{\boldsymbol{w}}$.pop();
        else
            $\boldsymbol{w_a} \leftarrow$ random weight;
        end
        Insert $\boldsymbol{w_a}$ into W;
        Assign $\boldsymbol{w_a}$ to agent $\boldsymbol{a}$;
    end
    Wait until agents learn policies for their current weights;
    foreach *agent* $\boldsymbol{a}$ **do**
        $\psi_{\boldsymbol{a}} \leftarrow$ last SF computed by agent $\boldsymbol{a}$;
        $\pi_{\boldsymbol{a}} \leftarrow$ last policy computed by agent $\boldsymbol{a}$;
        $\boldsymbol{w_a} \leftarrow$ current weight of agent $\boldsymbol{a}$;
        if $\psi_{\boldsymbol{a}} \notin \Psi$ **then**
            Insert $\psi_{\boldsymbol{a}}$ into Ψ;
            Insert $\pi_{\boldsymbol{a}}$ into Π;
            Remove obsolete corner weights from $Q_{\boldsymbol{w}}$;
            $W_c \leftarrow$ getCornerWeights($\psi, \boldsymbol{w_a}, \Psi$);
            foreach $\boldsymbol{w} \in W_c$ **do**
                $\Delta(\boldsymbol{w}) \leftarrow$ getImprovementEstimate($\boldsymbol{w}, \Psi, W$);
                Insert ($\boldsymbol{w}$, $\Delta(\boldsymbol{w})$) into $Q_{\boldsymbol{w}}$;
            end
        end
    end
end
return Π, Ψ

Figure 4.1 – Example: a simple MOMDP

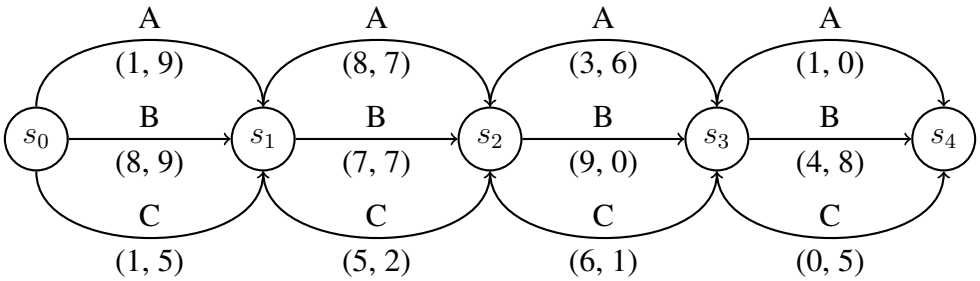
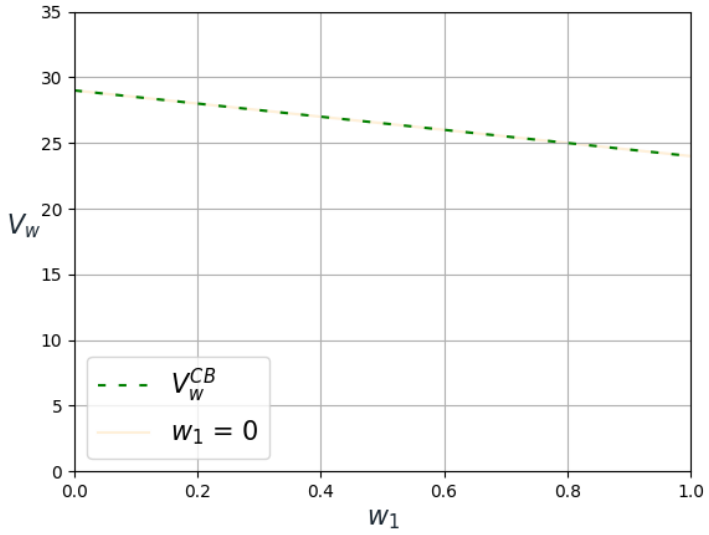
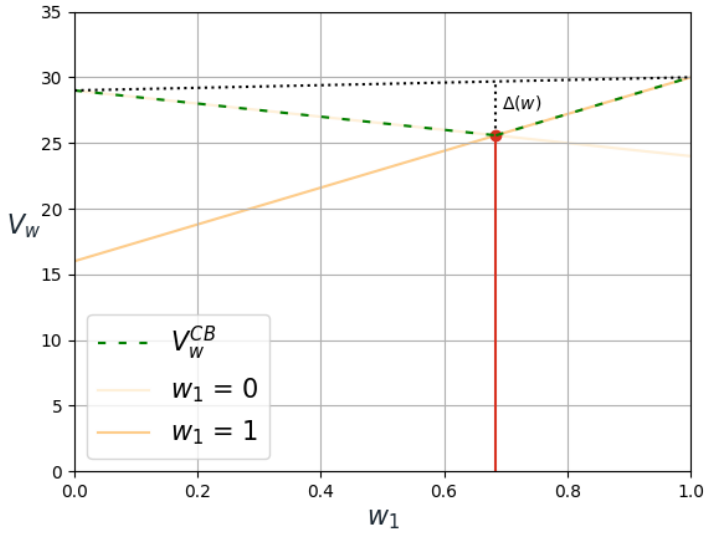


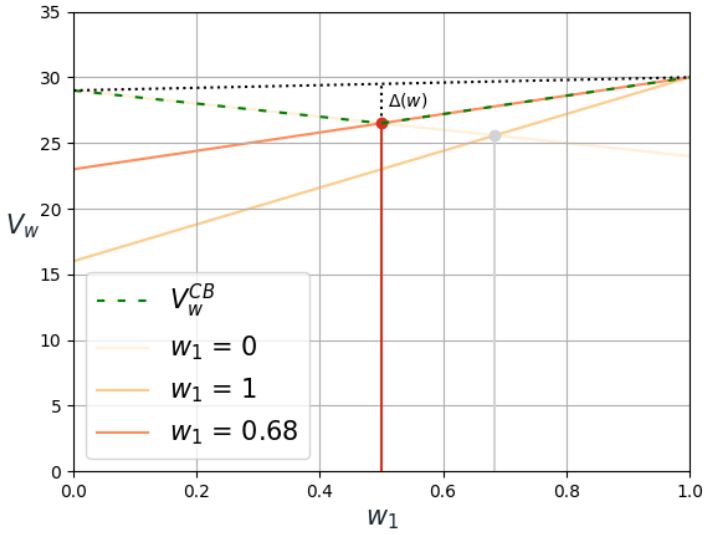
Figure 4.2 – Computing a CCS for the example using OLS



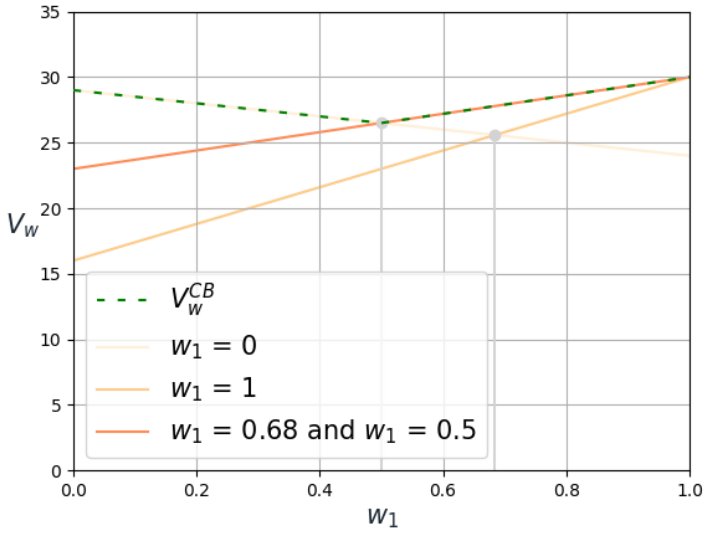
(a) First, the method inserts into $Q_{\boldsymbol{w}}$ the weight vectors (1, 0) and (0, 1), the weights in the extrema of the weight simplex, with infinite priority. $\boldsymbol{w}$ = (1, 0) is popped off $Q_{\boldsymbol{w}}$, and the best policy for this weight is computed.



(b) $\boldsymbol{w}$ = (0, 1) is popped off $Q_{\boldsymbol{w}}$, and the best policy for this weight is computed. The corner weight $\boldsymbol{w}$ = (0.32, 0.68) is found, with greatest possible improvement of $\Delta(\boldsymbol{w})$. This corner point is inserted in $Q_{\boldsymbol{w}}$, with a priority of $\Delta(\boldsymbol{w})$.



(c) $\boldsymbol{w}$ = (0.32, 0.68) is popped off $Q_{\boldsymbol{w}}$, and the best policy for this weight is computed. The corner weight $\boldsymbol{w}$ = (0.5, 0.5) is found, with greatest possible improvement of $\Delta(\boldsymbol{w})$. This corner point is inserted in $Q_{\boldsymbol{w}}$, with a priority of $\Delta(\boldsymbol{w})$.



(d) $\boldsymbol{w}$ = (0.5, 0.5) is popped off $Q_{\boldsymbol{w}}$, and the best policy for this weight is computed. Notice that the best policy for this weight is the same as the best policy for $\boldsymbol{w}$ = (0.32, 0.68), so both curves overlap. No new corner point is found, and $Q_{\boldsymbol{w}}$ is empty. The policies computed form a CCS, and the method terminates.

4.2 Computing behaviors for new preferences using GPI

In Section 4.1, it is shown how the method makes use of a multi-agent extension of the OLS algorithm to decentrally compute a shared set of policies, and their respective successor features, which corresponds to a CCS. This section shows how the agents can create new policies for any possible weights using the policies in this shared set, thus transferring knowledge acquired from the source tasks (scalarized versions of the multi-objective problem using the weights in $Q_{\boldsymbol{w}}$) to the target tasks (scalarized versions of the multi-objective problem using new weights given to the agents during execution).

It is important to note that agents reuse knowledge acquired from policies learned by themselves and by others. This constitutes a type of multi-agent TL method, henceforth referred to as policy transfer (since previous policies are used to create new policies, both the source and target of the transfer are policies, thus policy transfer).

Policy improvement and GPI are only mentioned briefly in Section 2.1 and Section 2.5, respectively. Since they constitute a fundamental part of the proposed method, composing the core of the TL process, these operations are explained in more detail here, for the sake of a better order of presentation of information.

The *policy improvement theorem*, given by Theorem 2, is an extremely important result for RL. It shows that, for any state s, by selecting an action a in which $Q^{\pi}(s,a) > Q^{\pi}(s,\pi(s))$, a better policy emerges. This means that, by acting greedily with respect to the value function, policies can be improved upon.

Theorem 2 *(SUTTON; BARTO, 2018; BELLMAN, 1957) Let π and π' be two deterministic policies such that, for any possible state s,*

$$Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s). \tag{4.4}$$

Then, it holds that

$$\pi' \succeq \pi. \tag{4.5}$$

The operation of improving policies by acting greedily with respect to their value functions is called policy improvement, and is shown in Eq. 4.6.

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a). \tag{4.6}$$

Within the framework of SFs, (BARRETO et al., 2017) extended Theorem 2 to

a set of multiple policies. This is the *generalized policy improvement theorem*, given by Theorem 3.

Theorem 3 *(BARRETO et al., 2017) Let* Π *be a set of deterministic policies and let* π' *be a deterministic policy such that, for any possible state* s*,*

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \qquad \forall \pi \in \Pi. \tag{4.7}$$

Then, it holds that

$$\pi' \succeq \pi \qquad \forall \pi \in \Pi. \tag{4.8}$$

As aforementioned, the operation of improving a set of policies by acting greedily with respect to all of their value functions is called GPI, a generalization of policy improvement which was originally applied to the SFs framework. However, by showing the equivalence between SFs and MORL, (ALEGRE; BAZZAN; SILVA, 2022) showed that it is possible to use GPI to improve policies in MORL settings. This result is key to the proposed method.

To understand how Theorem 3 can be applied to MORL, suppose the policies in Π correspond to policies learned by scalarizing a multi-objective problem using different weights. Now consider that a new weight vector is given to an agent. Because of the result in Theorem 3, this agent can create a new policy for this new weight vector by simply acting greedily with respect to the value function of all policies in Π for a scalarized version of the multi-objective problem using this new weight vector.

Now that a theoretical foundation has been laid, let us continue explaining the method. After having decentrally computed a shared set of policies Π, the agents are now ready to create new policies for any possible weight vector. Given a new weight $\boldsymbol{w}$, the agent uses Eq. 4.9, where $\boldsymbol{Q}^{\pi}$ is the vector-valued action-value function of policy π, to generate a new policy, $\pi_{\boldsymbol{w}}^{GPI}$, best suited for the new weight vector given.

$$\pi_{\boldsymbol{w}}^{GPI}(s) = \arg\max_{a \in \mathcal{A}} \max_{\pi \in \Pi} \boldsymbol{w} \cdot \boldsymbol{Q}^{\pi}(s, a). \tag{4.9}$$

For a single-agent scenario, (ALEGRE; BAZZAN; SILVA, 2022) proved that the policy $\pi_{\boldsymbol{w}}^{GPI}$ is optimal for $\boldsymbol{w}$ if Π corresponds to a CCS. However, since we are interested in multi-agent scenarios, the convergence guarantees no longer hold. Nevertheless, Chapter 5 empirically shows that for complex MARL problems, even though $\pi_{\boldsymbol{w}}^{GPI}$ is not theoretically optimal, its performance is effective.

5 EXPERIMENTS AND RESULTS

This chapter presents the experimental settings and results, and empirically shows that the method is both efficient at building a shared set of policies and effective at combining these policies using GPI to generate behaviors for different preferences over objectives. To evaluate the proposed method, two domains are used. The first, discussed in Section 5.1, is a multi-agent extension of an environment commonly used as a benchmark in the SFs literature. The second domain, presented in Section 5.2, is a traffic signal control environment with vehicles and pedestrians.

The Four-Room environment is used as one of the experimental settings in this work because of its common presence in many relevant related works. The results for this domain show that the proposed method can efficiently generate a shared set of policies that can be combined via GPI.

To show that the method also performs well for even more complex domains, and that the policies generated for new preferences over objectives are effective, a traffic signal control environment where traffic controllers optimize for both vehicles and pedestrians is also used, since this is an inherently distributed and multi-objective domain. Also, the agent's actions in this environment are highly coupled, which makes it even more challenging.

As explained in Section 4.1, the agents can use any temporal difference learning algorithm to learn policies for the weights they receive. For our experiments, in both domains, Q-learning with experience replay (FEDUS et al., 2020) was used. Table 5.1 shows the learning parameters used. Several values for each parameter were tested, but these were the ones that induced the best performance.

Table 5.1 – Q-learning parameters

Parameter	Value
α	0.1
ϵ	0.05
γ	0.95
Experience replay buffer size	1000000

5.1 Four-Room

The Four-Room environment is a domain commonly used in the SFs literature (BARRETO et al., 2017; GIMELFARB et al., 2021; ALEGRE; BAZZAN; SILVA, 2022). To evaluate the proposed method, this environment has been extended to a multi-agent scenario.

Figure 5.1 – Four-Room domain

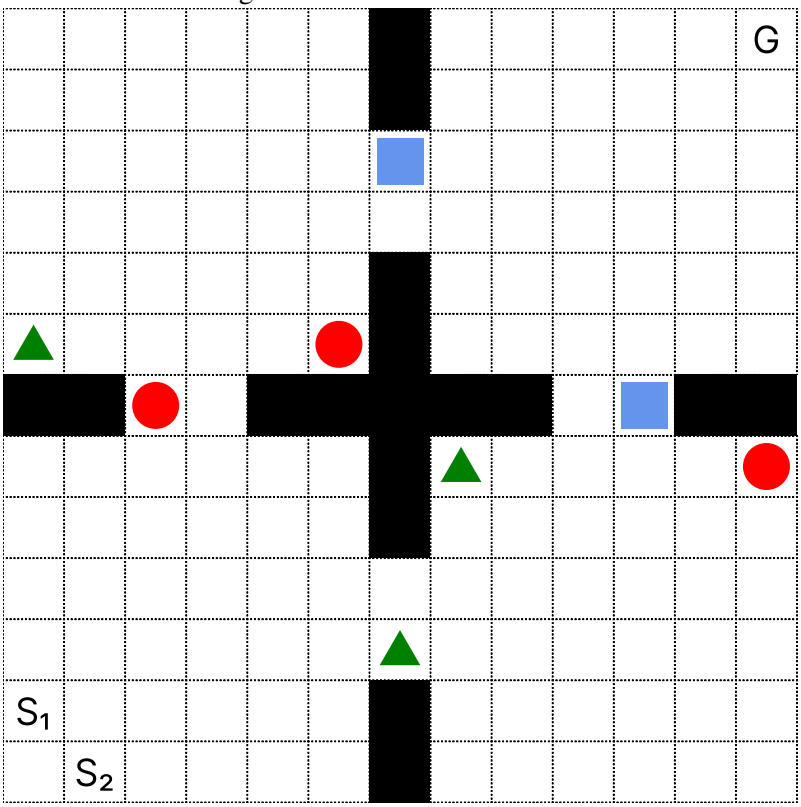


Figure 5.1 depicts the environment. It is a 13×13 grid composed of four separate rooms, hence its name. The black squares represent the walls separating the rooms (the agents are unable to walk through these walls). There are three different types of objects (blue squares, green triangles and red circles), which are collected by the agents once they step on their corresponding squares. Each object corresponds to a different objective (there are, therefore, three different objectives). For this multi-agent extension, a second agent has been added. The agents start in the squares indicated by the labels S_1 and S_2. Once they both reach the square indicated by the label G, the episode ends.

At each time step t, each agent observes a vector $s_t = [x, y]$ which represents the coordinates describing the agent's current position in the grid. The agents are unaware of each other, but their actions are highly coupled.

Each agent has four possible actions: up, down, right and left (which move the

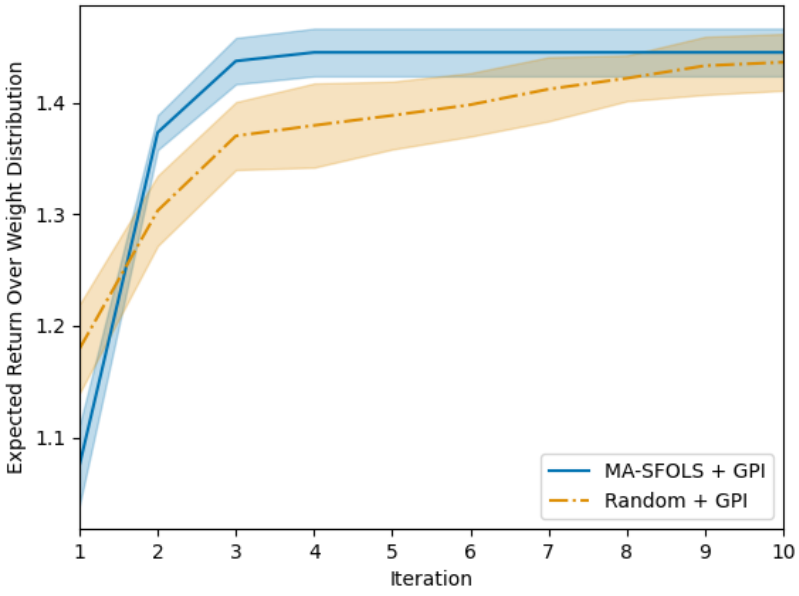
agent one square in the corresponding direction). Agents are unable to step on black squares (which represent walls) and unable to leave the grid.

The reward function $\boldsymbol{\mathcal{R}} \rightarrow \mathbb{R}^3$ is a three-dimensional vector representing the type of object in the agent's current cell ([1, 0, 0] if blue square, [0, 1, 0] if green triangle, [0, 0, 1] if red circle and [0, 0, 0] if blank). Once an agent steps on a square with an object, the agent receives the respective reward, and the object disappears, so the other agent can no longer receive a non-zero reward by stepping on that square. The agents, therefore, compete against each other.

This section aims at showing that the proposed method is efficient at generating a shared set of policies that can be effectively combined via GPI. Since the method iteratively assigns weights to the agents, which learn policies to scalarized versions of the multi-objective problem at each iteration, efficiency here can be measured by the amount of iterations required to build the complete set of policies. A common baseline in the literature, which is used here as well, is to use random weights at each iteration.

Figure 5.2 shows the average return of both agents for 12 random test weights, after each iteration of MA-SFOLS + GPI and Random + GPI. As mentioned, for Random + GPI each agent is assigned a random weight at each iteration (instead of the corner weights, which offer the greatest possible improvement). For both MA-SFOLS and Random, GPI is used at every iteration to combine the policies learned so far to generate behaviors for the test weights.

Figure 5.2 – Expected return of the agents over different weights (Four-Room)



From the plot, it is clear that the proposed method takes considerably less iterations in order to build an approximate CCS. For this example, the shared set of policies in most runs of the MA-SFOLS was already sufficient at iteration 4, whilst Random took more than twice as much iterations (approximately 10) to reach a similar level of returns.

The results empirically show the concept explained in Theorem 1: that corner weights offer the greatest possible improvement to expand the current set of policies. In fact, according to this theorem, no other weights assigned to the agents besides the ones selected by MA-SFOLS at each iteration could have built a CCS in less iterations.

5.2 Traffic Signal Control

For reasons mentioned previously, besides a common benchmark in the literature, a more complex domain, traffic signal control, is also used. The traffic scenario, shown in Figure 5.3, contains three intersections. The traffic lights in each intersection are controlled by independent RL agents. The experiments were performed using a microscopic traffic simulator, namely SUMO (LOPEZ et al., 2018) (Simulation of Urban MObility).

The scenario selected is especially interesting and non-trivial because it considers pedestrians. Figure 5.4 shows a zoomed view of an intersection, in which lanes, vehicles, sidewalks, pedestrians and a crossing can be seen. We stress that pedestrians are rarely addressed in RL-based methods for signal control.

Every link has 150 m in length, two lanes and is one-way. There is one vertical and three horizontal Origin-Destination (OD) pairs. A vehicle is inserted in an origin node, and is removed from the simulation in a destination node. Vehicles go in the North-South (N-S) direction in vertical links, and in the West-East (W-E) direction in horizontal links. Each vertical link (the vertical OD pair is composed of four vertical links) has a pedestrian sidewalk, and each intersection has a crossing for pedestrians. Pedestrians are also inserted in an origin node and removed from the simulation in a destination node. Pedestrians only go in the North-South (N-S) direction.

Traffic signals in this scenario have a minimum and maximum time they must remain green. They are referred to as *minGreenTime* and *maxGreenTime*, respectively. For the experiments, the value used for *minGreenTime* was 10 seconds, and 50 seconds for *maxGreenTime*. All signals have three phases, two for vehicles and one for pedestrians. The phase for pedestrians is green when the North-South phase for vehicles is green, and is red when the West-East phase for vehicles is green.

Figure 5.3 – Traffic signal control environment

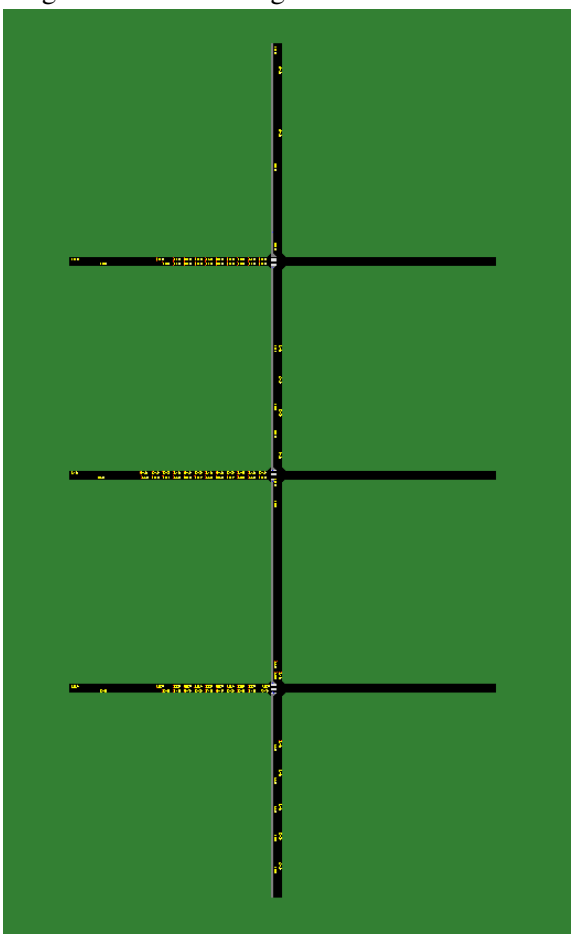
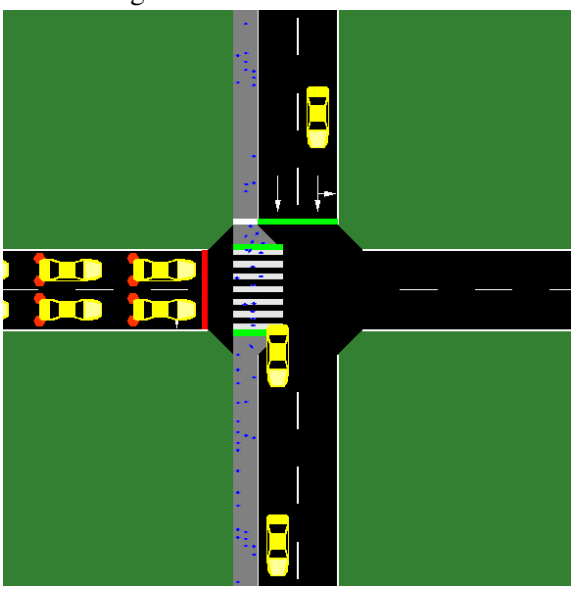


Figure 5.4 – Traffic intersection



For this scenario, the vehicular demand is the following: one vehicle is inserted in all 3 horizontal Origin-Destination pairs every 3.3 seconds, and one vehicle is inserted in the vertical Origin-Destination pair every 20 seconds. As for pedestrians, one pedestrian is inserted every second in the simulation. Each episode runs for 500 seconds.

At each time step t (which corresponds to five seconds of real-life traffic dynamics), each agent observes a vector s_t, given by Eq. (5.1), which describes the current state of the respective intersection. $\rho \in \{0, 1\}$ is binary variable that indicates the current active green phase ($\rho = 0$ when the West-East phase is green, and $\rho = 1$ when the North-South and pedestrian phases are green). $\tau \in [0, 1]$ is the elapsed time of the current signal phase divided by *maxGreenTime*. L is the set of all incoming lanes (for both vehicles and pedestrians). The density $\Delta_l \in [0, 1]$ is defined as the number of vehicles or pedestrians in the incoming lane $l \in L$ divided by the total capacity of the lane. $q_l \in [0, 1]$ is defined as the number of queued vehicles or pedestrians in the incoming lane $l \in L$ divided by the total capacity of the lane. A vehicle is considered to be queued if its speed is below 0.1 m/s. A pedestrian is considered to be queued if it is stopped before a crossing waiting for its phase to turn green.

$$s_t = [\rho, \tau, \Delta_1, ..., \Delta_{|L|}, q_1, ..., q_{[L]}] \tag{5.1}$$

Each learning agent chooses a discrete action a_t at each time step t. For our scenario, since all intersections have two incoming links, there are two phases (there are actually three phases, but the North-South phase and the pedestrian phase are always the same), so each agent has only two actions: *keep* and *change*. The former keeps the current green signal active, while the latter switches the current green light to another phase. The agents can only choose *keep* if the current green phase has been active for less than *maxGreenTime*, and can only choose *change* if the current green phase has been active for more than *minGreenTime*.

For this scenario, the traffic controllers have two objectives: to minimize the waiting time of vehicles, and to minimize the waiting time of pedestrians. Thus, the reward function $\boldsymbol{\mathcal{R}} \rightarrow \mathbb{R}^2$ is a two-dimensional vector, given by Eq. (5.2), where the first component is the change in cumulative vehicle waiting time between successive time steps, and the second component is the change in cumulative pedestrian waiting time between successive time steps.

$$\boldsymbol{\mathcal{R}}_t = [Wv_t - Wv_{t+1}, Wp_t - Wp_{t+1}] \tag{5.2}$$

Wv_t is the cumulative vehicle waiting time at time step t, given by Eq. (5.3), where V_t is the set of incoming vehicles, and $w_{v,t}$ is the total waiting time of vehicle v since it entered the incoming road until time step t.

$$Wv_t = \sum_{v \in V_t} w_{v,t} \tag{5.3}$$

Wp_t is the cumulative pedestrian waiting time at time step t, given by Eq. (5.4), where P_t is the set of incoming pedestrians, and $w_{p,t}$ is the total waiting time of pedestrian p at the crossing until time step t.

$$Wp_t = \sum_{p \in P_t} w_{p,t} \tag{5.4}$$

Figure 5.5 – Expected return of the agents over different weights (Signal Control)

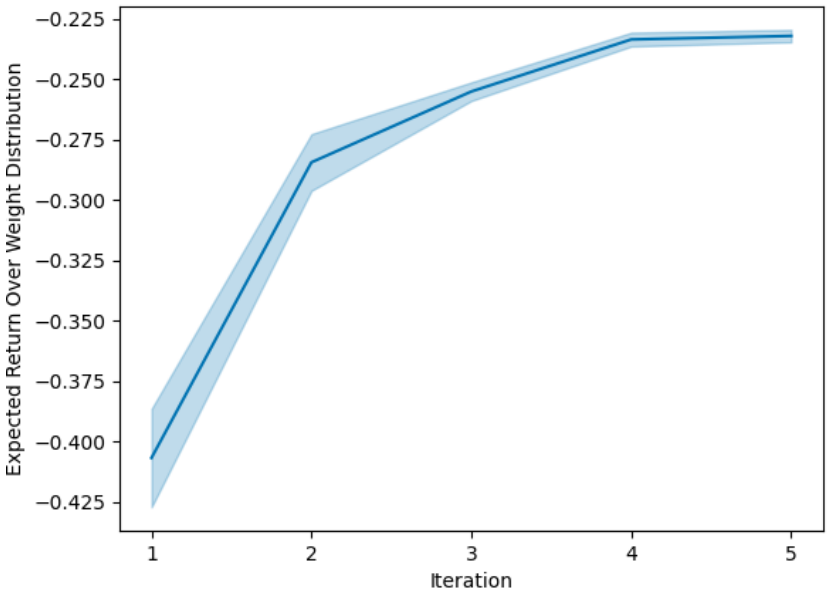


Figure 5.5 shows the average return of all three traffic controllers for 25 random test weights, after each iteration of MA-SFOLS + GPI. The performance of a few policies generated by GPI is shown in Figure 5.6, Figure 5.7 and Figure 5.8. Note that the legends of the plots only show one dimension of the weight vector because since there are only two objectives, and the weights adhere to the simplex constraints, only one of the dimensions of $\boldsymbol{w}$ is sufficient to express all possible instances of weights ($w_0 = 1 - w_1$).

It is clear from these plots how conflicting both objectives are (the higher the preference over one objective, the worse the performance is for the other objective). This makes this problem even more difficult, and also helps to serve as a motivation for multi-objective methods. Clearly, for this problem, a multi-objective method is required, as a priori scalarization of the problem would not be sufficient to address this scenario if the weights were not known in advance.

Figure 5.6 – Waiting time of pedestrians for a few policies generated by GPI

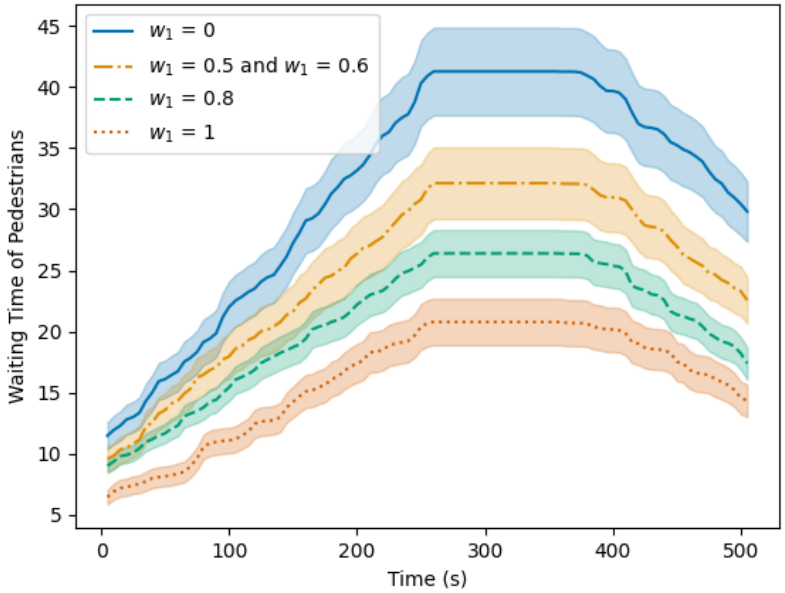
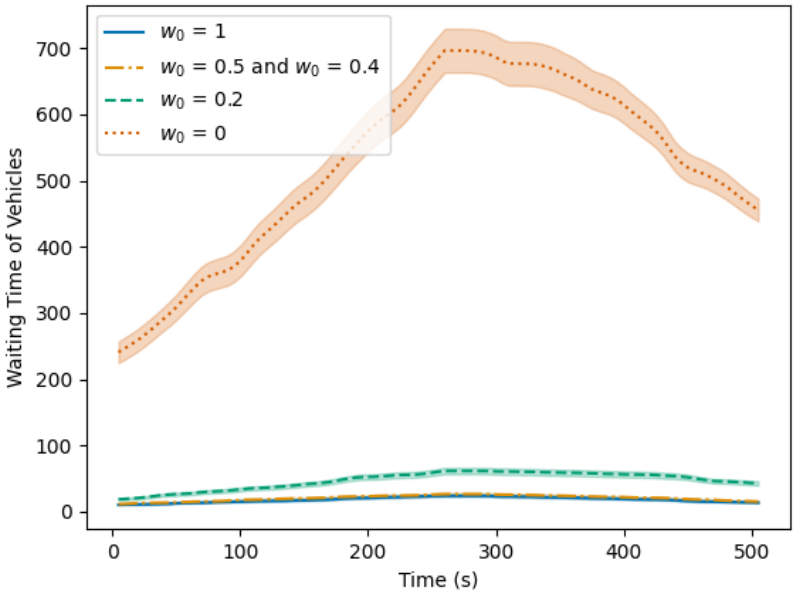


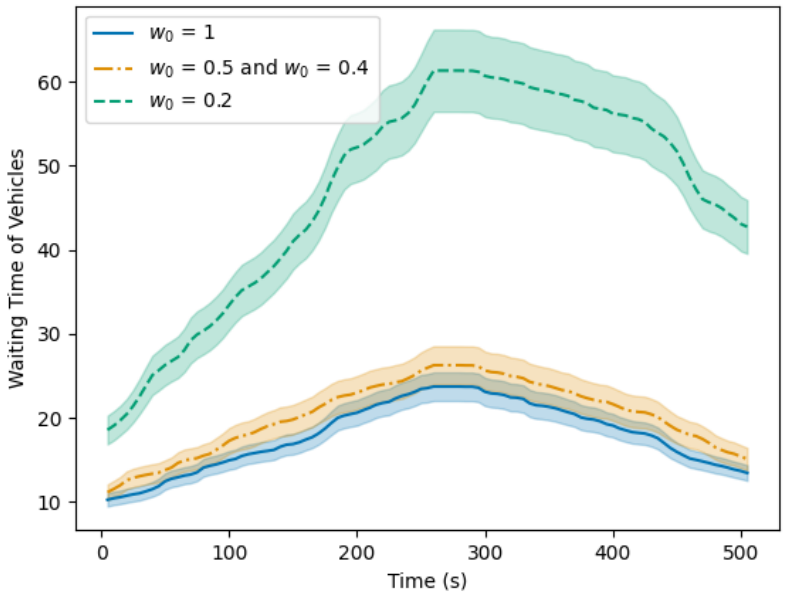
Figure 5.7 – Waiting time of vehicles for a few policies generated by GPI



Section 5.1 showed the efficiency of the proposed method (measured by iterations required to form an approximate CCS). This section aims at showing the effectiveness of the method, that is, that the policies generated by GPI are effective for new weights. To show this, the performance of policies generated by GPI are compared to policies learned by Q-learning for a few selected weights.

Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12 and Figure 5.13 show a compar-

Figure 5.8 – Zoomed view of waiting time of vehicles



ison between the performance (for both objectives) of policies generated by MA-SFOLS + GPI and the performance of policies learned by Q-learning via a priori scalarization for a few weights. It is clear that the policies generated by GPI have a very similar performance to the policies directly learned by Q-learning via a priori scalarization. Therefore, these results show that the method is able to appropriately leverage previous knowledge in order to create effective policies for new values of $\boldsymbol{w}$.

The results in this section and in Section 5.1 empirically show that the method is both efficient and effective at building a shared set of policies that can be combined via GPI to generate behaviors for new weights, so that the RL agents can perform well for any required preference of objectives.

Figure 5.9 – MA-SFOLS + GPI vs Q-learning: $\boldsymbol{w} = [0.5, 0.5]$ (pedestrians)

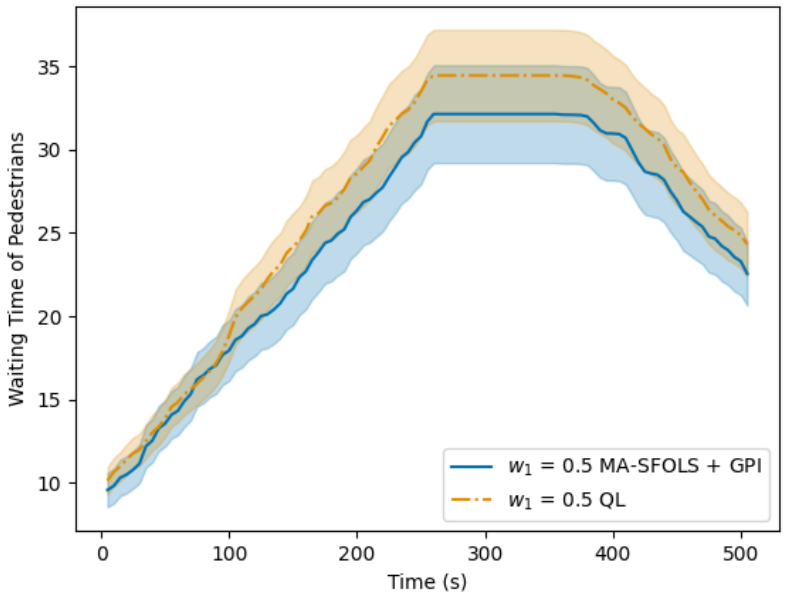


Figure 5.10 – MA-SFOLS + GPI vs Q-learning: $\boldsymbol{w} = [0.4, 0.6]$ (pedestrians)

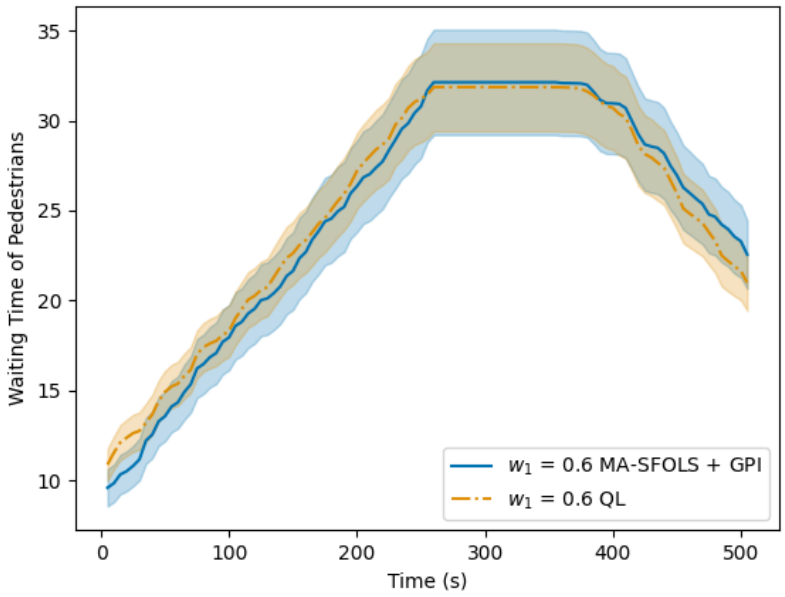


Figure 5.11 – MA-SFOLS + GPI vs Q-learning: $\boldsymbol{w} = [0.2, 0.8]$ (pedestrians)

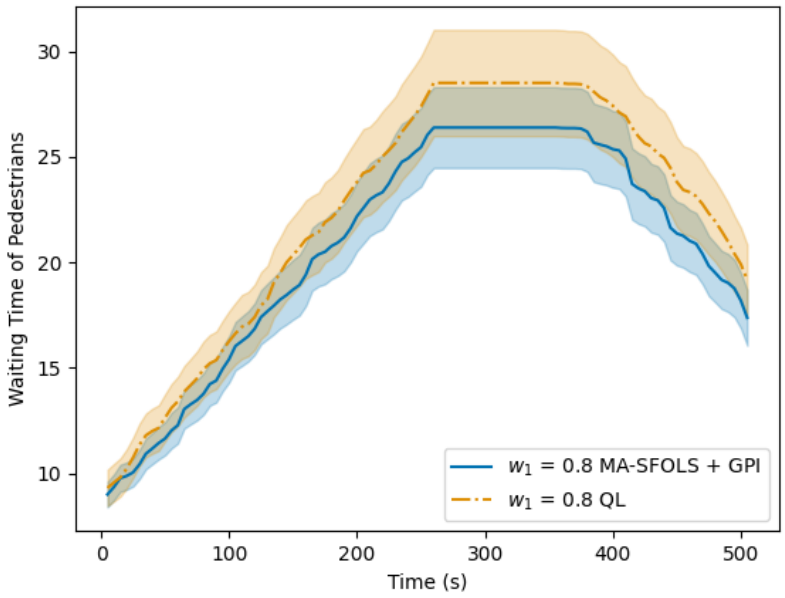


Figure 5.12 – MA-SFOLS + GPI vs Q-learning: $\boldsymbol{w} = [0.2, 0.8]$ (vehicles)

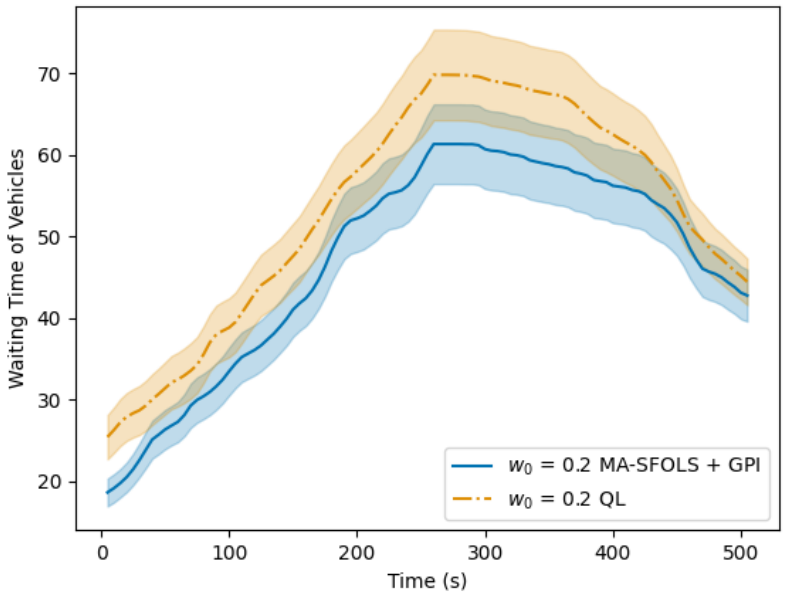
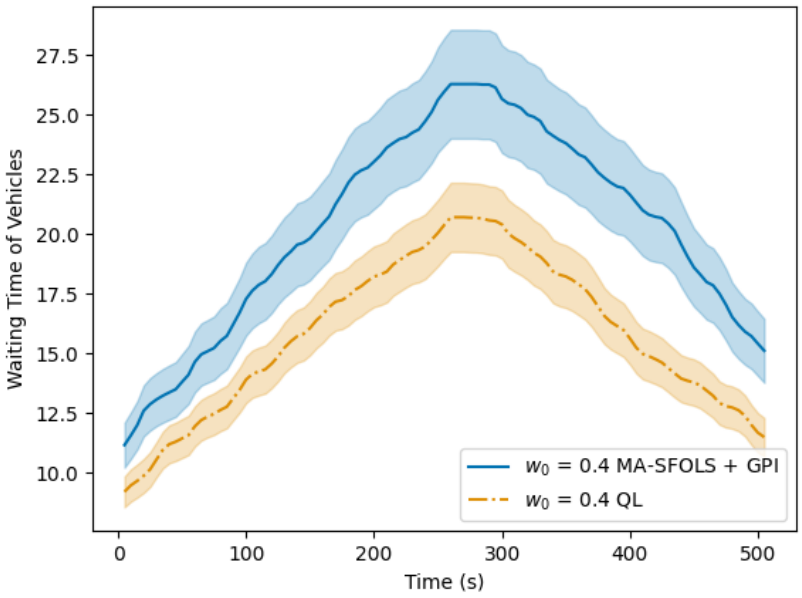


Figure 5.13 – MA-SFOLS + GPI vs Q-learning: $\boldsymbol{w} = [0.4, 0.6]$ (vehicles)



6 CONCLUSION

This work presented a multi-objective multi-agent transfer learning method in which agents decentrally build a shared set of policies during training, and then combine these policies to create new policies for new preferences with respect to their objectives during the execution phase. Within the method, two layers of knowledge transfer can be discerned: knowledge sharing and policy transfer.

Knowledge sharing, mentioned above, refers to the fact that the policies learned by one agent can be used by another, thus all the agents in the system share their knowledge with each other. Policy transfer refers to the fact that, by leveraging a generalization of policy improvement, the proposed method also enables agents to combine policies learned by themselves and by the other agents in order to create new policies, thus, policy transfer.

The proposed method is very useful for problems that are inherently distributed (therefore, a multi-agent solution is preferable or required) and multi-objective, and scalarizing them is not an option, either because the weights for the objectives are not known during training, or because the tradeoffs between objectives are not clear.

To evaluate the method, besides a standard domain in the SFs literature, a traffic signal control domain with both vehicles and pedestrians was used. It must be noted that optimizing for pedestrians is rarely addressed in the RL literature for traffic signal control.

The results empirically showed that the method is both efficient and effective at building a shared set of policies that can be combined via GPI to generate behaviors for new weights, so that the RL agents can perform well for any required preference of objectives.

This is one of the first works to address TL in the context of MOMARL, and the first work (to the author's best knowledge) that leverages successor features and generalized policy improvement for settings with multiple objectives and multiple agents.

As future work, the author would like to increase the information available to each agent, in a way that every agent knows the current state and current weight of other agents, so as to decrease the effects of non-stationarity that are present in multi-agent systems. Another potential line of work would be designing an inherently multi-agent coverage set, and changing the method so that it computes this multi-agent coverage set instead of a regular CCS.

REFERENCES

ABELS, A. et al. Dynamic weights in multi-objective deep reinforcement learning. In: **Proceedings of the 36th International Conference on Machine Learning**. [S.l.]: International Machine Learning Society (IMLS), 2019. v. 97, p. 11–20.

ALEGRE, L. N.; BAZZAN, A. L. C.; SILVA, B. C. da. Optimistic linear support and successor features as a basis for optimal policy transfer. In: CHAUDHURI, K. et al. (Ed.). **Proceedings of the 39th International Conference on Machine Learning**. PMLR, 2022. (Proceedings of Machine Learning Research, v. 162), p. 394–413. Available from Internet: <https://proceedings.mlr.press/v162/alegre22a.html>.

BARRETO, A. et al. Successor features for transfer in reinforcement learning. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2017. v. 30.

BARRETO, A. et al. Fast reinforcement learning with generalized policy updates. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 117, n. 48, p. 30079–30087, 2020. ISSN 0027-8424.

BARRETT, S.; STONE, P. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2015. (AAAI'15), p. 2010–2016. ISBN 0262511290.

BAZZAN, A. L. C. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. **Autonomous Agents and Multiagent Systems**, v. 18, n. 3, p. 342–375, June 2009.

BECKER, R. et al. Transition-independent decentralized markov decision processes. In: **Proceedings of The Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)**. [S.l.]: New York, ACM, 2003. p. 41–48.

BELLMAN, R. E. **Dynamic Programming**. Princeton: Princeton University Press, 1957.

BIANCHI, R. A. C. et al. Heuristically-accelerated multiagent reinforcement learning. **IEEE Transactions on Cybernetics**, v. 44, n. 2, p. 252–265, 2014.

BORSA, D. et al. Universal successor features approximators. In: **Proceedings of the 7th International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2019.

BUŞONIU, L.; BABUSKA, R.; SCHUTTER, B. D. A comprehensive survey of multiagent reinforcement learning. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, IEEE, v. 38, n. 2, p. 156–172, 2008.

CHENG, H.-T. **Algorithms for partially observable Markov decision processes**. Thesis (PhD) — University of British Columbia, 1988. Available from Internet: <https://open.library.ubc.ca/collections/ubctheses/831/items/1.0098252>.

DAYAN, P. Improving generalization for temporal difference learning: The successor representation. **Neural Computation**, v. 5, n. 4, p. 613–624, 1993.

DUAN, H.; LI, Z.; ZHANG, Y. Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network. **EURASIP Journal on Advances in Signal Processing**, v. 2010, 12 2010.

EGEA, A. C.; CONNAUGHTON, C. **Assessment of Reward Functions in Reinforcement Learning for Multi-Modal Urban Traffic Control under Real-World limitations**. 2020. ArXiv preprint arXiv:2010.08819.

FEDUS, W. et al. Revisiting fundamentals of experience replay. In: **Proceedings of the 37th International Conference on Machine Learning**. Vienna, Austria: [s.n.], 2020.

GIMELFARB, M. et al. Risk-aware transfer in reinforcement learning using successor features. In: **Proceedings of the 35th Annual Conference on Advances in Neural Information Processing Systems**. Online: [s.n.], 2021.

GUPTA, T. et al. **UneVEn: Universal Value Exploration for Multi-Agent Reinforcement Learning**. 2021. ArXiv preprint arXiv:2010.02974.

HAYES, C. F. et al. A practical guide to multi-objective reinforcement learning and planning. **CoRR**, abs/2103.09568, 2021. Available from Internet: <https://arxiv.org/abs/2103.09568>.

KHAMIS, M. A.; GOMAA, W. Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In: **2012 11th International Conference on Machine Learning and Applications**. [S.l.: s.n.], 2012. v. 1, p. 586–591.

KIM, S. H. et al. Disentangling successor features for coordination in multi-agent reinforcement learning. In: **International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022**. [S.l.: s.n.], 2022. p. 751–760.

KOBER, J.; BAGNELL, J. A.; PETERS, J. Reinforcement learning in robotics: A survey. **The International Journal of Robotics Research**, v. 32, n. 11, p. 1238–1274, 2013.

LAZARIC, A. Transfer in reinforcement learning: A framework and a survey. In: **Reinforcement Learning: State-of-the-Art**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 143–173. ISBN 978-3-642-27645-3.

LE, H. M. et al. Coordinated multi-agent imitation learning. In: **Proceedings of the 34th International Conference on Machine Learning**. [S.l.: s.n.], 2017. (ICML'17), p. 1995–2003.

LIU, W. et al. Efficient exploration for multi-agent reinforcement learning via transferable successor features. **IEEE/CAA Journal of Automatica Sinica**, v. 9, 2022.

LONG, Q. et al. **Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning**. [S.l.]: arXiv, 2020.

LOPEZ, P. A. et al. Microscopic traffic simulation using SUMO. In: **The 21st IEEE International Conference on Intelligent Transportation Systems**. [S.l.: s.n.], 2018.

MAZYAVKINA, N. et al. Reinforcement learning for combinatorial optimization: A survey. **Computers and Operations Research**, v. 134, p. 105400, 2021. ISSN 0305-0548.

MINSKY, M. Steps toward artificial intelligence. **Proc. IRE**, 1961. Available from Internet: <http://web.media.mit.edu/~minsky/papers/steps.html>.

MNIH, V. et al. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, feb. 2015.

NOAEEN, M. et al. **Reinforcement Learning in Urban Network Traffic Signal Control: A Systematic Literature Review**. 2021. Available from Internet: <engrxiv.org/ewxrj>.

PROPER, S.; TADEPALLI, P. Multiagent transfer learning via assignment-based decomposition. In: **2009 International Conference on Machine Learning and Applications**. [S.l.: s.n.], 2009. p. 345–350.

RĂDULESCU, R. et al. Multi-objective multi-agent decision making: a utility-based analysis and survey. **Autonomous Agents and Multi-Agent Systems**, v. 34, 04 2020.

ROESS, R. P.; PRASSAS, E. S.; MCSHANE, W. R. **Traffic Engineering**. 3rd. ed. [S.l.]: Prentice Hall, 2004. 816 p.

ROIJERS, D. **Multi-Objective Decision-Theoretic Planning**. Thesis (PhD) — University of Amsterdam, 2016.

ROIJERS, D. M. et al. A survey of multi-objective sequential decision-making. **J. Artificial Intelligence Research**, AI Access Foundation, El Segundo, CA, USA, v. 48, n. 1, p. 67–113, oct. 2013. ISSN 1076-9757.

SHAPLEY, L. S. Stochastic games. **Proceedings of the National Academy of Sciences**, v. 39, n. 10, p. 1095–1100, 1953.

SHI, H. et al. Lateral transfer learning for multiagent reinforcement learning. **IEEE Transactions on Cybernetics**, p. 1–13, 2021.

SILVA, F. L. d.; COSTA, A. H. R. A survey on transfer learning for multiagent reinforcement learning systems. **Journal of Artificial Intelligence Research**, v. 64, p. 645–703, 2019. ISSN 1076-9757.

SILVER, D. et al. **Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm**. [S.l.]: arXiv, 2017.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. Second. [S.l.]: The MIT Press, 2018.

TAYLOR, A. et al. Accelerating learning in multi-objective systems through transfer learning. In: **International Joint Conference on Neural Networks (IJCNN)**. Beijing, China: IEEE, 2014. p. 2298–2305.

TAYLOR, M. E.; STONE, P. Transfer learning for reinforcement learning domains: A survey. **Journal of Machine Learning Research**, v. 10, n. 56, p. 1633–1685, 2009.

TORREY, L.; TAYLOR, M. E. Teaching on a budget: Agents advising agents in reinforcement learning. In: **Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems**. St. Paul, MN, USA: IFAAMAS, 2013. Available from Internet: <https://dl.acm.org/doi/10.5555/2484920.2485086>.

WEI, H. et al. **A Survey on Traffic Signal Control Methods**. 2020. Preprint arXiv:1904.08117. Available from Internet: <http://arxiv.org/abs/1904.08117>.

YAU, K.-L. A. et al. A survey on reinforcement learning models and algorithms for traffic signal control. **ACM Comput. Surv.**, ACM, v. 50, n. 3, 2017. ISSN 0360-0300.

YU, C. et al. Reinforcement learning in healthcare: A survey. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 55, n. 1, nov 2021. ISSN 0360-0300.

ZHUANG, F. et al. A comprehensive survey on transfer learning. **Proceedings of the IEEE**, p. 1–34, 07 2020.