Multi-objective ω-Regular Reinforcement Learning

ERNST MORITZ HAHN, University of Twente, The Netherlands
MATEO PEREZ, University of Colorado Boulder, USA
SVEN SCHEWE, University of Liverpool, UK
FABIO SOMENZI and ASHUTOSH TRIVEDI, University of Colorado Boulder, USA
DOMINIK WOJTCZAK, University of Liverpool, UK

The expanding role of reinforcement learning (RL) in safety-critical system design has promoted ω-automata as a way to express learning requirements—often non-Markovian—with greater ease of expression and interpretation than scalar reward signals. However, real-world sequential decision making situations often involve multiple, potentially conflicting, objectives. Two dominant approaches to express relative preferences over multiple objectives are: (1) *weighted preference*, where the decision maker provides scalar weights for various objectives, and (2) *lexicographic preference*, where the decision maker provides an order over the objectives such that any amount of satisfaction of a higher-ordered objective is preferable to any amount of a lower-ordered one. In this article, we study and develop RL algorithms to compute optimal strategies in Markov decision processes against multiple ω-regular objectives under weighted and lexicographic preferences. We provide a translation from multiple ω-regular objectives to a scalar reward signal that is both *faithful* (maximising reward means maximising probability of achieving the objectives under the corresponding preference) and *effective* (RL quickly converges to optimal strategies). We have implemented the translations in a formal reinforcement learning tool, Mungojerrie, and we present an experimental evaluation of our technique on benchmark learning problems.

CCS Concepts: • **Theory of computation → Automata over infinite objects**; **Logic and verification**; • **Computing methodologies → Reinforcement learning;**

Additional Key Words and Phrases: Multi-objective reinforcement learning, ω-regular objectives, lexicographic preference, weighted preference, automata-theoretic reinforcement learning

ACM Reference format:
Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2023. Multi-objective ω-Regular Reinforcement Learning. *Form. Asp. Comput.* 35, 2, Article 12 (July 2023), 24 pages.
https://doi.org/10.1145/3605950

The authors acknowledge the support received from various sources. In particular, this work was supported in part by the Engineering and Physical Sciences Research Council through Grants No. EP/P020909/1 and No. EP/X017796/1, the National Science Foundation (NSF) through Grant No. CCF-2009022 and the NSF CAREER Award No. CCF-2146563. Furthermore, funding was provided by the European Union's Horizon 2020 research and innovation programme under Grant Agreements No. 101032464 (SyGaST), No. 864075 (CAESAR), and No. 956123 (FOCETA).

Authors' addresses: E. M. Hahn, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands; email: e.m.hahn@utwente.nl; M. Perez and A. Trivedi, Department of Computer Science, University of Colorado Boulder, Boulder, Colorado, USA; emails: mateo.perez@colorado.edu, ashutosh.trivedi@colorado.edu; S. Schewe and D. Wojtczak, Department of Computer Science, University of Liverpool, Liverpool, UK; emails: Sven.Schewe@liverpool.ac.uk, d.wojtczak@liv.ac.uk; F. Somenzi, Department of Electrical, Computer & Energy Engineering, University of Colorado Boulder, Boulder, Colorado, USA; emails: fabio@colorado.edu.

0934-5043/2023/07-ART12 $15.00
https://doi.org/10.1145/3605950

1 INTRODUCTION

Reinforcement learning (RL) [65] is a sequential optimisation approach whereby a learning agent (the *learner*) optimally resolves a sequence of choices based on feedback received from the decision maker (the *interpreter*). This feedback often takes the form of rewards and penalties, with strength proportional to the fitness of the choices made by the learner, as judged by the interpreter, toward some higher-level objectives; we call such objectives *learning objectives*. The RL paradigm reflects the way dopamine-driven organisms latch on to past rewarding choices [19, 32], and hence, historically, RL paradigms integrated a nuanced, myopic way of looking at the reward sequences in the form of *discounted-sum* of reward. More recently, other forms of reward aggregation, such as limit-average, have also been considered (interestingly, there seems to be a correspondence of average-reward RL with dopamine-serotonin opponent interactions [8, 20]). We call such objectives (discounted-sum, average reward) *learner's aggregators*.

Reward Translation for Reinforcement Learning. *We posit that the key programming challenge in developing Reinforcement Learning-driven systems is translation. It involves designing a reward function that maps the sequence of learner's choices to scalar rewards, given a class of learning objectives and aggregator functions. The aim is to ensure that an RL agent, maximising the aggregated sum of rewards, converges to a policy that maximises the learning objective.*

Currently, reinforcement learning often relies on a manual translation of, often instinctive, objectives to reward signals. Such translations not only depend on the expertise of the decision maker in reward engineering, they pose obstacles to an adoption of the formal methods paradigm to system design due the lack of formal specifications and the lack of guarantees on the faithfulness of the translation. As the applications of RL, powered by deep learning [33], continue to provide creative solutions [44, 55, 62, 68] to problems that traditionally relied on human ingenuity, the integration of RL in safety-critical system design [54, 56, 71] is inevitable. As a response, a number of different translation schemes [10, 13, 28, 35, 40, 60] have been proposed to translate single-objective learning requirements expressed in formal languages (such as linear temporal logic [4] and ω-automata [4]) to scalar rewards. However, when the learning objectives are given as multiple—potentially conflicting—objectives [51, 59, 66, 67], the aforementioned translation schemes are not directly applicable.

Multi-objective ω-Regular Reinforcement Learning. *This article investigates the problem of reward translation in reinforcement learning, specifically focusing on scenarios where the learning requirements are presented as a finite set of ω-regular objectives.*

1.1 Pareto Optimality and Weighted Preferences

In scenarios where various ω-regular objectives may conflict with each other, a single strategy that maximizes the probability of satisfaction for all individual objectives may not exist. In such cases, decision makers may be interested in analyzing the trade-offs among different strategies. A strategy is considered *Pareto optimal* [18, 22, 67] if no other strategy can achieve a higher satisfaction probability for one objective without reducing the satisfaction probability for another objective. The satisfaction probability vectors for all Pareto optimal policies forms the Pareto curve.

Unfortunately, the Pareto curve for ω-regular objectives cannot be exactly computed in polynomial time [22, Theorem 2.1]. On the positive side, since Pareto curves are convex for multiple

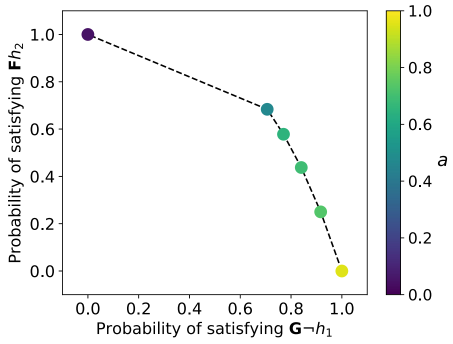

Fig. 1. Double bet example with two objectives: (1) The first coin is never heads, and (2) the second coin is eventually heads. These properties can be expressed in Linear temporal logic (LTL) [5] as **globally** (i.e., at every step, denoted by the temporal modality $\mathbf{G}$) avoid first coin landing heads ($\mathbf{G}\neg h_1$), and **finally** (i.e., at some time in the future, denoted by the temporal modality $\mathbf{F}$) the second coin lands head ($\mathbf{F}h_2$). The figure shows the learned Pareto curve from various weights, where a is the weight for the first objective.

ω-regular objectives [22], one can employ the standard approach of approximating them by optimising the weighted sum of various objectives for an even spread of weights [18]. We will follow this approach and enable reinforcement learning of weighted-sums of ω-regular objectives. Throughout the article, we assume that the weight vector is positive. Minimising the satisfaction of a particular objective can be done by taking the negation of the property.

Example 1.1 (Double Bet: Pareto Curve). Consider a simple scenario where an agent has two biased coins that turn up heads with probability 1/12 and 1/4, respectively. The agent can set the number of times to flip the coins, up to four times, or the agent can decide to set both coins to heads. There are two competing objectives: Ensure that the first coin is never heads, and ensure that the second coin is eventually heads. We consider weightings between these two properties where the first is weighted by $0 < a < 1$ and the second is weighted by $1 - a$. Figure 1 shows the maximal probabilities of satisfaction obtainable for each property as learned by RL with our method by sweeping through values of a. Initially, when $a = 1$, the agent decides to flip the coins 0 times, satisfying the first property, but neglecting the second. As a decreases to zero, the agent decides to flip the coins more times. Eventually, the agent decides to set the two coins to heads, satisfying the second property, but neglecting the first. The colored dots in Figure 1 are the critical points where the strategy changes. ◀

1.2 Lexicographic Preferences

Instead of specifying a particular weight for each objective, it can be more natural to specify the preference order of the objectives. Lexicographic preference is a well-studied comparison criterion in utility theory [26, 52, 72], formal methods [7, 11, 12, 16], and reinforcement learning [29, 31, 72]. Under a lexicographic preference over multiple objectives, a decision maker prefers any amount of satisfaction of a higher-order objective over any amount of satisfaction of a lower-ordered one. We study RL where the learning objective concerns a lexicographic optimisation of ω-regular objectives. That is, we are given k different ω-regular specifications, ordered by importance. We optimise the probabilities of these specifications as follows. An optimal strategy has to maximise the probability that the first objective is achieved. The strategy then also has to maximise the probability that the second objective holds among those strategies that achieve the maximum probability for the first objective. The next priority is to maximise the third objective, and so forth. Lexicographic preference is useful when, for instance, one wants to guarantee critical requirements first,

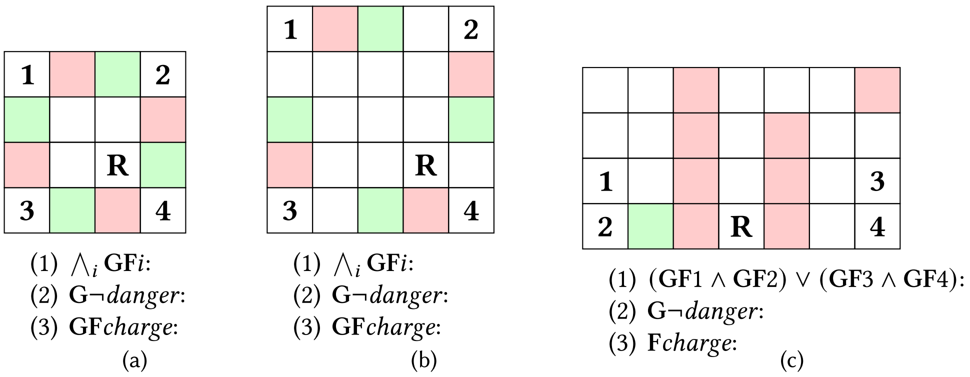

Fig. 2. **Robot** Case Studies (R 4×4, R 5×5, and R 4×7): Grid-world Examples with Multiple ω-Regular Objectives in Robot Navigation. The LTL property **GF**p is often referred to as *infinitely often* p.

functional ones second, and only then focus on the non-functional ones. Let us consider three grid-world scenarios to illustrate optimisation under lexicographic ω-regular objectives.

Example 1.2 (Motivating Lexicographic Objectives). Consider a robot navigating a grid world, as illustrated in Figure 2, with three scenarios of varying sizes. At each step, the robot chooses one of four directions, in which it moves zero, one, or two steps. If there is enough space, then the probability of moving two steps in the chosen direction is **p**, while the probability of moving only one step is 1−**p**. If there is only room for one step, then the robot moves a single step deterministically. If there is no space to move, then the robot remains stationary. In Figure 2, the initial position of the robot is indicated by **R**. The red fields represent hazards for the robot, while the green fields allow the robot to recharge its battery. Additionally, there are four fields labelled **1** to **4**.

For each of the scenarios (a)–(c), we have three different objectives expressed as LTL formulas [5]. The objectives are numbered (1) to (3), with (1) representing the highest priority objective and (3) representing the lowest priority objective.

(a) In this scenario, the first priority of the robot is to visit all four numbered fields infinitely often, the second priority is never to enter a dangerous area, and the third priority is to visit some charging field infinitely often. It is possible to fulfill the primary objective with probability 1. However, the secondary objective can then not be fulfilled with any positive probability: Due to its uncertain movements, the robot always has a chance to step into a dangerous field when trying to achieve the primary objective. The third priority, however, can again be fulfilled with probability 1.
(b) The objectives are the same as for scenario (a). However, because of the larger grid, the robot can stay out of danger (**G**¬*danger*) while satisfying its primary objective ($\bigwedge_i$ **GF**i); it can meet all three objectives with probability 1.
(c) The primary objective is to visit fields **1** and **2** infinitely often *or* to visit fields **3** and **4** infinitely often. The secondary objective is to avoid dangerous fields. The tertiary one is to eventually visit a charging field. Here, the robot can again fulfill its primary objective with probability 1. The secondary one can only be fulfilled with probability $1 - (\mathbf{p} \cdot (1 - \mathbf{p}))$: To fulfill its primary objective and then the secondary one, the best the robot can do is to move upwards and then to the right. If, by moving to the right, the robot first moves one field but then two fields, then it will run into a dangerous field. When maximising the primary and secondary objectives, the tertiary objective can then only be fulfilled with probability $\mathbf{p} \cdot (1 - \mathbf{p})$, because trying to reach the charging field cannot be done without

entering a dangerous field with at least probability $1 - \mathbf{p}$ (the chance of not "jumping" over the left barrier of red fields). Therefore, the tertiary objective is only pursued when the secondary objective has been missed.

The priorities play a key role here: For scenario (c), if the safety objective $\mathbf{G}\neg\text{danger}$ were the most important, then the robot would be able to completely avoid unsafe fields, fulfilling this objective with probability 1. However, the probability to fulfil the recurrence objective $(\mathbf{GF}1 \wedge \mathbf{GF}2) \vee (\mathbf{GF}3 \wedge \mathbf{GF}4)$ would be $\mathbf{p}$, because the robot can only reach the lower-right corner of the grid with probability $\mathbf{p}$ if it is to avoid danger at all cost. One can also consider giving property 1 the highest priority, followed by properties 2 and 3 with the same priority. In this case, the optimal strategy is for the robot to go to the lower-left corner of the grid. This results in the satisfaction of properties 1 and 3 with probability 1, and the satisfaction of property 2 with probability $1 - \mathbf{p}$. The cumulative satisfaction $1 + (1 - \mathbf{p})$ of properties 2 and 3 is higher under this strategy than the cumulative satisfaction of value 1 obtained if the robot went for the lower-right corner instead. ◄

1.3 Contributions

We study the reward translation problem for reinforcement learning of multiple ω-regular objectives with weighted and lexicographic preferences. We assume that each objective is given as a **Good-for-MDPs (GFM)** Büchi automaton [37]: A semantic condition on nondeterministic Büchi automata that requires that the optimal satisfaction probability of the automaton over an MDP is equal to the optimal probability of visiting accepting end-components [21] on the product of the MDP and the automaton. While suitable limit-deterministic Büchi automata [34, 61] are the most well-known classes of GFM automata, other characterisations (slim automata [37]) also exist.

In this work, we extend the limit-reachability theorem, a method introduced in our prior research [35, 36]. Our extension allows for the transformation of various combinations of Büchi conditions into different rewards, resulting in a reduction to a weighted reachability problem. By leveraging this reduction, optimal strategies can be computed over the resulting MDP (a finite MDP with Markovian scalar reward) using standard, off-the-shelf RL algorithms. It is important to note that, in practice, we do not perform the composition of the MDP with the automata and the transformation into rewards before the RL process begins. Instead, we employ an on-the-fly composition strategy to avoid the expensive construction of parts of the product automaton that would not be visited during RL.

Extension over the conference version. In the conference version [38] of this article, the focus was on lexicographic ω-regular objectives. However, this extended version introduces support for weighted ω-regular objectives, allowing for the approximation of the Pareto curve. As a result, combinations of lexicographic and weighted ω-regular objectives can now be considered. Our implementation of both reward schemes is available in Mungojerrie [39] (plv.colorado.edu/mungojerrie), an open-source tool designed for testing reward schemes for ω-regular objectives in finite-state MDPs. We provide evidence of the effectiveness of our reward schemes through various examples.

2 PRELIMINARIES

We write $\mathbb{B} = \{0, 1\}$ for the set of Boolean values, $\mathbb{N}$ for the set of natural numbers, and $\mathbb{R}$ for the set of real numbers. For $\ell, u \in \mathbb{R}$ with $\ell \leq u$, we define the closed interval $[\ell, u]$ as the set $\{n \in \mathbb{R} : \ell \leq n \leq u\}$, the open interval $]\ell, u[$ as the set $\{n \in \mathbb{R} : \ell < n < u\}$, and the intervals $[\ell, u[$ and $]\ell, u]$ as the sets $\{n \in \mathbb{R} : \ell \leq n < u\}$ and $\{n \in \mathbb{R} : \ell < n \leq u\}$, respectively. An *$\omega$-word* w on an alphabet Σ is a function $w \colon \mathbb{N} \to \Sigma$. The set of ω-words on Σ is written Σ^ω and a subset of Σ^ω is an *ω-language.*

A *discrete probability distribution* over a finite set S is a function $d\colon S \to [0,1]$ such that $\sum_{s \in S} d(s) = 1$. Let $\mathcal{D}(S)$ denote the set of all discrete distributions over S. A distribution $d \in \mathcal{D}(S)$ is a *point distribution* if $d(s) = 1$ for some $s \in S$. For $d \in \mathcal{D}(S)$, we write *supp*(d) for $\{s \in S\colon d(s) > 0\}$.

2.1 Markov Decision Processes: The Environment

A **Markov Decision Process (MDP)** $\mathcal{M}$ is a tuple $\langle S, s_0, A, T, \Sigma, L\rangle$ where S is a finite set of states, s_0 is a designated initial state, A is a finite set of *actions*, $T : S \times A \to \mathcal{D}(S)$ is the *probabilistic transition (partial) function*, Σ is the set of observations and $L : S \times A \times S \to \Sigma$ is the observation *labelling function* of the set of transitions. A *Markov chain* is an MDP where the set of actions is a singleton, meaning that there is only one possible action at each state.

We say that an action $a \in A$ is available (or enabled) in a state $s \in S$ if $T(s,a)$ is defined. For a state $s \in S$, the set $A(s) \subseteq A$ denotes the set of actions available in s. For states $s, s' \in S$ and $a \in A(s)$, we have that $T(s,a)(s')$ equals $\Pr(s'|s,a)$. A *run* of $\mathcal{M}$ is an ω-word $s_0, a_1, s_1, \ldots \in S \times (A \times S)^\omega$ such that $\Pr(s_{i+1}|s_i, a_{i+1}) > 0$ for all $i \geq 0$. A finite run is a finite such sequence. For a *run* $r = s_0, a_1, s_1, \ldots$, we define the corresponding labeled run as $L(r) = L(s_0, a_1, s_1), L(s_1, a_2, s_2), \ldots \in \Sigma^\omega$. We write *Runs*$(\mathcal{M})$ and *FRuns*$(\mathcal{M})$ for the set of runs and finite runs, respectively, of $\mathcal{M}$. Similarly, we write $\mathit{Runs}_s(\mathcal{M})$ and $\mathit{FRuns}_s(\mathcal{M})$ for the set of runs and finite runs of $\mathcal{M}$ starting from state s. When the MDP is clear from the context, we drop the argument $\mathcal{M}$.

A strategy in $\mathcal{M}$ is a function $\mu : \mathit{FRuns} \to \mathcal{D}(A)$ such that, for all finite runs r, we have $\mathit{supp}(\mu(r)) \subseteq A(\mathsf{last}(r))$, where $\mathsf{last}(r)$ is the last state of r. Let $\mathit{Runs}^\mu_s(\mathcal{M})$ denote the subset of runs $\mathit{Runs}_s(\mathcal{M})$ that correspond to strategy μ and initial state s. We say that a strategy μ is

- *pure*, if $\mu(r)$ assigns probability 1 to just one action in $A(\mathsf{last}(r))$ for all runs $r \in \mathit{FRuns}$;
- *stationary*, if $\mathsf{last}(r) = \mathsf{last}(r')$ implies $\mu(r) = \mu(r')$ for all finite runs $r, r' \in \mathit{FRuns}$;
- *positional*, if it is both pure and stationary; and
- *finite-state*, if there exists an equivalence relation $\sim$ on *FRuns* with finitely many equivalence classes, such that $\mu(r) = \mu(r')$ for all finite runs $r \sim r'$.

The behaviour of an MDP $\mathcal{M}$ under a strategy μ with starting state s is defined on a probability space $(\mathit{Runs}^\mu_s, \mathcal{F}^\mu_s, \Pr^\mu_s)$ over the set of infinite runs of $\mathcal{M}$ under μ from s. Given a random variable over the set of infinite runs $f : \mathit{Runs} \to \mathbb{R}$, we write $\mathbb{E}^\mu_s\{f\}$ for the expectation of f over the runs of $\mathcal{M}$ from state s that follow strategy μ. For any MDP $\mathcal{M}$ and stationary strategy μ, let $\mathcal{M}_\mu$ be the Markov chain resulting from choosing the actions in $\mathcal{M}$ according to μ.

2.2 ω-Regular Languages: The Requirements

A *nondeterministic Büchi automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma\rangle$, where Σ is a finite *alphabet*, Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\Delta \subseteq Q \times \Sigma \times Q$ is the set of transitions, and $\Gamma \subseteq Q \times \Sigma \times Q$ is the transition-based *acceptance condition*. An automaton $\mathcal{A}$ is *deterministic* if $(q, \sigma, q'), (q, \sigma, q'') \in \Delta$ implies $q' = q''$ and is *complete* if, for all $\sigma \in \Sigma$ and $q \in Q$, there is a transition $(q, \sigma, q') \in \Delta$.

A *run* r of $\mathcal{A}$ on $w \in \Sigma^\omega$ is an ω-word $r_0, w_0, r_1, w_1, \ldots$ in $(Q \times \Sigma)^\omega$ such that $r_0 = q_0$ and, for $i > 0$, it is $(r_{i-1}, w_{i-1}, r_i) \in \Delta$. A word has exactly one run in a deterministic, complete automaton. We write $\mathsf{inf}(r)$ for the set of transitions that appear infinitely often in the run r. A run r of $\mathcal{A}$ is *accepting* if $\mathsf{inf}(r) \cap \Gamma \neq \emptyset$. The *language*, denoted as $L_{\mathcal{A}}$, of automaton $\mathcal{A}$ (or *recognised* by $\mathcal{A}$) refers to the subset of words in Σ^ω that have accepting runs in $\mathcal{A}$. A language is considered *ω-regular* if it is accepted by some (nondeterministic) Büchi automaton.

Linear Time Logic (LTL) is a temporal logic whose formulae describe a subset of the ω-regular languages. It is often used to specify objectives in human-readable form. Given a set of atomic propositions AP, a is an LTL formula for each $a \in AP$. Moreover, if φ and ψ are LTL formulae, then so are $\neg\varphi$, $\varphi \vee \psi$, $\mathbf{X}\varphi$, $\psi \,\mathbf{U}\, \varphi$. Additional operators are defined as abbreviations: $\top \overset{\text{def}}{=} a \vee \neg a$; $\bot \overset{\text{def}}{=} \neg\top$; $\varphi \wedge \psi \overset{\text{def}}{=} \neg(\neg\varphi \vee \neg\psi)$; $\varphi \rightarrow \psi \overset{\text{def}}{=} \neg\varphi \vee \psi$; $\mathbf{F}\varphi \overset{\text{def}}{=} \top \,\mathbf{U}\, \varphi$; and $\mathbf{G}\varphi \overset{\text{def}}{=} \neg\mathbf{F}\neg\varphi$. We write $w \models \varphi$ if ω-word w over 2^{AP} satisfies LTL formula φ. The satisfaction relation is defined inductively [5, 53]. If $w = w_0 w_1 \ldots$, and $w^i = w_i w_{i+1} \ldots$, then $w \models a$ if $a \in w_0$; $w \models \neg\varphi$ if $w \not\models \varphi$; $w \models \varphi \vee \psi$ if $w \models \varphi$ or $w \models \psi$; $w \models \mathbf{X}\varphi$ if $w^1 \ldots \models \varphi$; $w \models \psi \,\mathbf{U}\, \varphi$ if, for some i, $w^i \models \varphi$ and for all $j < i$, $w^j \models \psi$.

2.3 Syntactic and Semantic Satisfaction Probabilities

Given an MDP $\mathcal{M}$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, we want to compute an optimal strategy, i.e., a strategy μ maximising the probability that the runs of $\mathcal{M}$ under μ belong to the language of $\mathcal{A}$. We define the *semantic satisfaction* probability for $\mathcal{A}$ and a strategy μ from state s as

$$\begin{aligned} \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu) &= \Pr{}^{\mu}_{s}\left\{ r \in Runs^{\mu}_{s}(\mathcal{M}) : L(r) \in L_{\mathcal{A}} \right\} \text{ and} \\ \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s) &= \sup_{\mu}\left(\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu) \right). \end{aligned}$$

A strategy μ_* is optimal for $\mathcal{A}$ if $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu_*) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$.

When using automata for the analysis of MDPs, we need a syntactic variant of the acceptance condition. Given an MDP $\mathcal{M} = \langle S, s_0, A, T, \Sigma, L \rangle$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, the *product* $\mathcal{M} \times \mathcal{A} = \langle S \times Q, (s_0, q_0), A \times Q, T^{\times}, \Gamma^{\times} \rangle$ is an MDP augmented with an initial state (s_0, q_0) and accepting transitions $\Gamma^{\times}$. The function $T^{\times} : (S \times Q) \times (A \times Q) \rightharpoonup \mathcal{D}(S \times Q)$ is defined by

$$T^{\times}((s,q),(a,q'))((s',q')) = \begin{cases} T(s,a)(s') & \text{if } (q, L(s,a,s'), q') \in \Delta \\ 0 & \text{otherwise.} \end{cases}$$

Finally, $\Gamma^{\times} \subseteq (S \times Q) \times (A \times Q) \times (S \times Q)$ is defined by $((s,q),(a,q'),(s',q')) \in \Gamma^{\times}$ if, and only if, $(q, L(s,a,s'), q') \in \Gamma$ and $T(s,a)(s') > 0$. A strategy $\mu^{\times}$ on the product defines a strategy μ on the MDP with the same value, and vice versa. Note that for a stationary $\mu^{\times}$, the strategy μ may need memory. We define the *syntactic satisfaction* probabilities as

$$\begin{aligned} \mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s,q),\mu^{\times}) &= \Pr{}^{\mu^{\times}}_{(s,q)}\left\{ r \in Runs^{\mu^{\times}}_{(s,q)}(\mathcal{M} \times \mathcal{A}) : \inf(r) \cap \Gamma^{\times} \neq \emptyset \right\} \\ \mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) &= \sup_{\mu^{\times}}\left(\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s,q_0),\mu^{\times}) \right). \end{aligned}$$

Note that $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ holds for a deterministic $\mathcal{A}$. In general, $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) \leq \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ holds, but equality is not guaranteed, because optimal resolution of nondeterministic choices may require access to future events.

An automaton $\mathcal{A}$ is ***good for MDPs* (GFM)**, if $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s_0) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$ holds for all MDPs $\mathcal{M}$ [37]. For an automaton to match $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$, its nondeterminism is restricted not to rely heavily on the future; rather, it must be possible to resolve the nondeterminism on-the-fly. In this article, we only consider GFM automata, which have this ability. Note that every LTL property and, more generally, every ω-regular objective can be expressed as a suitable GFM automaton [37].

An *end-component* of an MDP is a set $C \subseteq S$ that satisfies the following conditions: For every state $s \in C$, there exists an action $a_s \in A(s)$ such that $\{s' \mid T(s,a_s)(s') > 0\} \subseteq C$, and the graph formed by the vertices in C and the edges in

$$E = \{(s,s') \mid s \in C \text{ and } T(s,a_s)(s') > 0\}$$

is strongly connected. An end-component is *accepting* if at least one of such edges correspond to an accepting transition and is *maximal* if there does not exist an end-component $C' \supset C$. It is well-known (see, e.g., Reference [21]) that for every strategy the union of the end-components is visited with probability 1 and once an end-component, C', is entered there is a pure finite-state strategy that visits its every edge infinitely many times while never leaving C'. For ω-regular objectives, optimal satisfaction probabilities and strategies can be computed using graph-theoretic techniques [21] over the product structure to maximise the probability of reaching accepting end-components.

2.4 Reinforcement Learning

A *rewardful* MDP is a pair $(\mathcal{M}, \rho)$, where $\mathcal{M}$ is an MDP and $\rho \colon S \times A \to \mathbb{R}$ is a reward function. A rewardful MDP $(\mathcal{M}, \rho)$ under a strategy μ determines a sequence of random rewards $\rho(X_{i-1}, Y_i)_{i \geq 1}$, where X_i and Y_i are the random variables denoting the ith state and action, respectively. Depending upon the problem of interest, different aggregator functions may be of interest. The *reachability* objective $\mathrm{Reach}_T^{\mathcal{M}}(s, \mu)$ (with $T \subseteq S$) is defined as

$$\mathrm{Reach}_T^{\mathcal{M}}(s, \mu) = \mathrm{Pr}_s^{\mu} \left\{ s, a_1, s_1, \ldots \in \mathit{Runs}_s^{\mu}(\mathcal{M}) \colon \exists\, i \text{ such that } s_i \in T \right\}.$$

For a given discount factor $\lambda \in [0, 1[$, the *discounted reward* objective $\mathrm{Disct}_\lambda^{\mathcal{M}}(s, \mu)$ is defined as

$$\mathrm{Disct}_\lambda^{\mathcal{M}}(s, \mu) = \lim_{N \to \infty} \mathbb{E}_s^{\mu} \left\{ \sum_{1 \leq i \leq N} \lambda^{i-1} \rho(X_{i-1}, Y_i) \right\}.$$

For the aggregator $\mathrm{Agg}^{\mathcal{M}}$ where $\mathrm{Agg}^{\mathcal{M}} \in \{\mathrm{Reach}_T^{\mathcal{M}}, \mathrm{Disct}_\lambda^{\mathcal{M}}\}$, we define the optimal reward $\mathrm{Agg}_*^{\mathcal{M}}(s)$ as the supremum of $\mathrm{Agg}_\sigma^{\mathcal{M}}(s)$ over all strategies $\sigma \in \Sigma_{\mathcal{M}}$, starting from an initial state s. A strategy $\sigma \in \Sigma_{\mathcal{M}}$ is considered optimal for $\mathrm{Agg}^{\mathcal{M}}$ if $\mathrm{Agg}_\sigma^{\mathcal{M}}(s) = \mathrm{Agg}_*^{\mathcal{M}}(s)$ for all states $s \in S$. For an MDP $\mathcal{M}$ and an aggregator $\mathrm{Agg}^{\mathcal{M}} \in \{\mathrm{Reach}(T)^{\mathcal{M}}, \mathrm{Disct}(\lambda)^{\mathcal{M}}\}$, the optimal reward and an optimal strategy can be computed using value iteration, policy iteration, or, in polynomial time, using linear programming [24, 58]. However, when the transition and reward structure of the MDP is unknown, such techniques are not applicable. In the case of MDPs with unknown transition and reward structure, ***reinforcement learning*** **(RL)** [65] provides a framework to compute optimal strategies through repeated interactions with the environment. One classical RL algorithm is the Q-learning algorithm developed by Watkins [70].

The problem of computing an optimal strategy for the reachability probability objective can be reduced to the problem of computing an optimal strategy for the discounted objective under some large discount factor [24]. For this reason, we restrict our attention to algorithms for computing optimal strategies for the discounted performance objective. The optimal discounted value $\mathrm{Disct}_{\lambda,*}^{\mathcal{M}} = V : S \to \mathbb{R}$ can be characterized [58] using the following equations:

$$V(s) = \max_{a \in A(s)} \left\{ \rho(s, a) + \lambda \sum_{s' \in S} p(s' \mid s, a) \cdot V(s') \right\}. \tag{1}$$

Being a *contraction*, the unique fixed point of the following *value improvement operator* $\Phi : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$, defined as

$$\Phi : F \mapsto \max_{a \in A(s)} \left\{ \rho(s, a) + \lambda \sum_{s' \in S} p(s' \mid s, a) \cdot F(s') \right\},$$

gives the solution for the discounted optimality Equations (1). From the Banach fixed point theorem [6], the following sequence of iterates $V_{t+1}(s) = \max_{a \in A(s)} \{\rho(s, a) + \lambda \sum_{s' \in S} p(s' \mid s, a) \cdot V_t(s')\}$, starting from an arbitrary value function $V_0 \in [S \to \mathbb{R}]$, converges to the optimal value $\mathrm{Disct}(\lambda)_*^{\mathcal{M}}$. This sequence of iterates are often represented in the following equivalent form by using $Q_t(s, a)$

as the estimate for the value of a state-action pair at step t of the iteration:

$$Q_{t+1}(s,a) = \rho(s,a) + \lambda \sum_{s' \in S} p(s' \mid s,a) \cdot V_t(s') \quad \text{and} \quad V_{t+1}(s) = \max_{a \in A(s)} Q_{t+1}(s,a).$$

Let $Q_*(s,a) = \lim_{t\to\infty} Q_t(s,a)$ for all $s,a \in S \times A$. We refer to such value associated with the state-action pair as the Q-value or the "quality" of the pair.

The Q-learning algorithm assumes that instead of knowing the MDP beforehand, we are given data samples of the form $(s_t, a_t, r_{t+1}, s_{t+1}) \in S \times A \times \mathbb{R} \times S$ for $t = 0, 1, 2, \ldots$ such that $p(s_{t+1} \mid s_t, a) > 0$ and $\rho(s_t, a_t) = r_{t+1}$. The Q-learning algorithm iteratively applies the following update to the state-action value function:

$$Q_{t+1}(s,a) = \begin{cases} (1-\alpha_t) Q_t(s,a) + \alpha_t (r_{t+1} + \lambda \cdot \max_{a' \in A(s)} Q_t(s_{t+1}, a')), & \text{if } (s,a) = (s_t, a_t), \\ Q_t(s,a), & \text{otherwise,} \end{cases}$$

where $0 \le \alpha_t < 1$ is the learning-rate at the step $t \ge 0$.

Theorem 2.1 (Q-learning [69]). *For bounded rewards $|r_t| \le B$ and learning rates $0 \le \alpha_t < 1$ that satisfy the conditions:*

$$\sum_{t=0}^{\infty} \alpha_t = \infty \text{ and } \sum_{t=0}^{\infty} \alpha_t^2 < \infty,$$

we have that $Q_t(x,a) \to Q_*(s,a)$ as $t \to \infty$ for all $(s,a) \in S \times A$ almost surely.

3 MULTI-OBJECTIVE ω-REGULAR REINFORCEMENT LEARNING

For an unknown MDP $\mathcal{M}$ with initial state s_0 and multiple ω-regular learning objectives $(\varphi_1, \ldots, \varphi_k)$, our goal is to design a faithful and effective reward scheme under both lexicographic and weighted preferences. In the rest of the article, we assume that the properties $(\varphi_1, \ldots, \varphi_k)$ are available as GFM Büchi automata $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$. The challenge is to characterise a reward function over the product space of $\mathcal{M}$ and $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$ (independent of the probabilistic transition structure of $\mathcal{M}$) in such a way that any strategy maximising the reward function maximises the satisfaction probabilities for the objectives under the appropriate (lexicographic or weighted) preference.

3.1 Lexicographic Preference

Without loss of generality, we assume that the lexicographic preference order of the decision maker is $\varphi_1 > \cdots > \varphi_k$, i.e., the decision maker cares about φ_1 most and φ_k least. For two k-dimensional vectors $v = (v_1, \ldots, v_k)$ and $v' = (v'_1, \ldots, v'_k)$, we say that v is larger in the *lexicographic order* than v', denoted by $v > v'$, if there exists $1 \le i \le k$ such that $v_i > v'_i$ and $v_j = v'_j$ for all $j < i$. We write $v \ge v'$ if $v > v'$ or $v = v'$.

For an MDP $\mathcal{M}$ with initial state s_0 and ω-regular objectives $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$, the lexicographic value $\mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0, \mu)$ of a strategy μ and the optimal lexicographic value $\mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0)$ are defined as

$$\begin{aligned} \mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0, \mu) &= \left(\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}_1}(s_0, \mu), \ldots, \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}_k}(s_0, \mu)\right), \text{ and} \\ \mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0) &= \sup_{\mu} \mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0, \mu). \end{aligned}$$

We say that μ_* is a lexicographic optimal policy if $\mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0, \mu_*) = \mathsf{PSem}^{\mathcal{M},\text{lex}}_{\mathcal{A}_1,\ldots\mathcal{A}_k}(s_0)$.

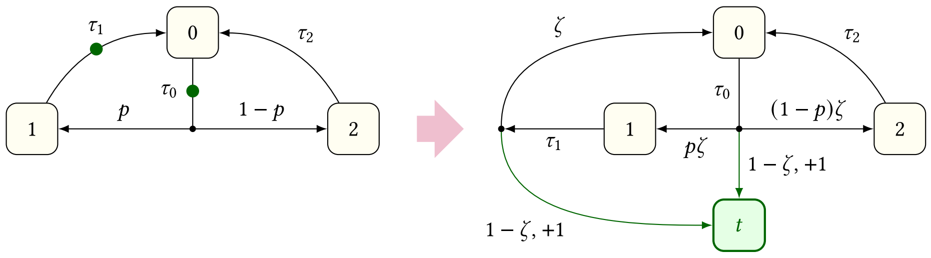

Fig. 3. Reward translation from ω-regular objectives to reachability reward.

3.2 Weighted Preference

For an MDP $\mathcal{M}$ with initial state s_0, the ω-regular objectives $(\mathcal{A}_1, \dots, \mathcal{A}_k)$, their relative weights as a vector $w = (w_1, \dots, w_k) \in \mathbb{R}^k_{\geq 0}$, we define weighted-value $\mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0, \mu)$ of a strategy μ and the weighted value $\mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0)$ of the MDP $\mathcal{M}$ as

$$\begin{aligned}\mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0, \mu) &= \sum_{i=1}^{k} w_i \cdot \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}_i}(s_0, \mu), \text{ and}\\ \mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0) &= \sup_{\mu} \mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0, \mu).\end{aligned}$$

We say that μ_* is a weighted-optimal policy if $\mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0, \mu_*) = \mathsf{PSem}^{\mathcal{M},w}_{\mathcal{A}_1,\dots\mathcal{A}_k}(s_0)$.

3.3 Faithful and Effective Reward Translation: Single-objective Case

Bridging the gap between ω-regular specifications and RL involves translating learning objectives into scalar rewards. This translation should ensure that an RL algorithm, focused on maximising scalar rewards, generates a policy that maximises the probability of satisfying the specification. We refer to this translation requirement as *faithfulness*. Additionally, the translation should be *effective*, meaning that the formulated reward should facilitate the reliable and efficient learning of optimal policies by mainstream RL algorithms, such as Q-learning [69].

For the single-objective setting, Figure 3 sketches the reward translation scheme (from GFM Büchi learning objective to a reachability aggregator) from Reference [35]. In this translation, maximising the chance to realize an ω-regular objective given by a GFM Büchi automaton $\mathcal{A}$ for an MDP $\mathcal{M}$ is reduced to maximising the chance to meet the reachability objective in the augmented MDP $\mathcal{R}^\zeta$, for $\zeta \in]0, 1[$, obtained from the product MDP $\mathcal{M} \times \mathcal{A}$ by

- adding a new target state t (either as a sink with a self-loop or as a point where the computation stops; we choose here the latter view) and by
- making the target t a destination of each accepting transition τ of $\mathcal{M} \times \mathcal{A}$ with probability $1 - \zeta$ and multiplying the original probabilities of all other destinations of an accepting transition τ by ζ.

Theorem 3.1 (Limit Reachability Theorem [35]). *For every MDP $\mathcal{M}$ and GFM automaton $\mathcal{A}$, the following holds:*

(1) For every $\zeta \in]0, 1[$, the MDPs $\mathcal{R}^\zeta$ and $\mathcal{M} \times \mathcal{A}$ have the same set of strategies.

(2) For a positional strategy μ, the chance of reaching the target t in $\mathcal{R}^\zeta_\mu$ is 1 if, and only if, the chance of satisfying the Büchi objective in $(\mathcal{M} \times \mathcal{A})_\mu$ is 1, i.e., $\mathsf{Reach}^{\mathcal{R}^\zeta}_t((s_0, q_0), \mu) = 1 \iff \mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s_0, q_0), \mu) = 1$.

(3) *There is a $\zeta_0 \in]0,1[$ such that, for all $\zeta \in [\zeta_0, 1[$, an optimal reachability strategy μ for $\mathcal{R}^\zeta$ is an optimal strategy for the Büchi objective in $\mathcal{M} \times \mathcal{A}$.*

To solve the reachability problem (an undiscounted reward maximisation problem) using Q-learning, one can exploit the concept of Blackwell-optimal strategies. Given an MDP $\mathcal{M}$, we say that a strategy μ is Blackwell-optimal if there exists a $\lambda_0 \in]0,1[$ such that μ is λ-discount optimal for all $\lambda \in]\lambda_0, 1[$. Moreover, if $\mathcal{M}$ has n states and all transition probabilities are rational with numerator and denominator bounded from above by M, then λ_0 is bounded from above by $1 - ((n!)^2 2^{2n+3} M^{2n^2})^{-1}$ [2, 42, 50]. The following theorem enables the application of Q-learning for discounted reward problem for total-reward when total rewards are bounded.

Theorem 3.2 (Blackwell-Optimality [49]). *Let $\mathcal{M}$ be an MDP and $\rho : S \times A \to \mathbb{R}$ be a reward function such that for every strategy μ of $\mathcal{M}$ the expected total reward is finite; then, every Blackwell-optimal strategy is total-reward optimal.*

Since the aforementioned reward scheme can be reduced to total reward objectives with bounded expected total reward, Q-learning can be applied with discount factor left as a hyperparameter.

3.4 Problem Definition

The key technical problem of this article concerns the design of faithful and effective reward schemes for multi-objective ω-regular specifications with lexicographic and weighted preferences. We reduce multiple Büchi objectives under both the lexicographic and weighted preferences to weighted reachability objectives defined via a product automaton constructed from the GFM automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, and some other auxiliary information. We introduce an intermediate objective, the weighted-Büchi automaton, and show reductions to it from lexicographic and weighted ω-regular objectives in Section 4. We then show how to reduce a weighted-Büchi automaton to weighted reachability, which can then be learned with RL, in Section 5.

4 MULTIPLE ω-REGULAR SPECIFICATIONS TO WEIGHTED BÜCHI AUTOMATA

This section presents a reduction technique that transforms lexicographic and weighted ω-regular objectives into a weighted-Büchi automaton. The reduction involves constructing a product automaton $\mathcal{P}$ using the GFM automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, along with additional auxiliary information. We start by focusing on lexicographic ω-regular objectives.

4.1 From Lexicographic Preference to Lexicographic-Büchi Automaton

In a first step, we discuss reductions from lexicographic ω-regular objectives, given as GFM Büchi automata, to a GFM automaton with lexicographic Büchi condition. We call such automata lexicographic-Büchi.

Lexicographic-Büchi Automata. A lexicographic-Büchi automaton is an ω-automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$ along with a valuation function $v : \Gamma \to \mathbb{B}^k$, which maps each accepting transition to a k dimensional Boolean vector different from the $\vec{0}$ vector. In our reduction the dimensions will correspond to the Büchi objectives $\mathcal{A}_1, \ldots, \mathcal{A}_k$ and the valuation function will indicate corresponding acceptance transitions. Let us first provide semantics for lexicographic-Büchi acceptance conditions. For convenience in the proofs, we provide two: The *independent* semantics and the *infimum* semantics. We will show that we obtain the correct results, irrespective of which semantics we use, for the translations we suggest: Both semantics provide the same value for the extended product automaton we define.

- The *infimum* semantics assigns to any run, r, the value according to the "worst accepting transition" seen infinitely many times in r. Namely, to the set $I_A = \mathsf{inf}(r) \cap \Gamma$ of accepting transitions visited infinitely many times along r, it assigns the $\vec{0}$ vector if I_A is empty, and otherwise the lexicographically minimal vector in $\{v(t) \mid t \in I_A\}$.
- The *independent* semantics treats all Büchi conditions independently: Using it, a run would be assigned a Boolean vector $(b_1, b_2, \ldots, b_k)$, where $b_i = 1$ if there is a transition $t \in I_A$ such that the ith component of $v(t)$ is 1.

Key Reduction. We assume that the individual ω-regular objectives are given by k GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$. From these automata, we discuss a construction to an equivalent extended product automaton $\mathcal{P}$, where equivalence means that, for all finite-state strategies, the value that can be obtained using the k GFM automata for the properties, and the infimum semantics and the independent semantics provide the same results.

Definition 4.1 (Product Automaton $\mathcal{P}$). Given GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, with $\mathcal{A}_i = (\Sigma, Q^i, q_0^i, \Delta^i, \Gamma^i)$, we define their product automaton $\mathcal{P} = (\Sigma, Q, q_0, \Delta, \Gamma, v)$, where:

- $Q = \bigtimes_{i=1}^{k} (Q^i \times \mathbb{B})$
- $q_0 = (q_0^1, 0; q_0^2, 0; \ldots; q_0^k, 0)$
- $\Delta = \Gamma \cup \Delta'$, where the non-accepting transitions Δ' are defined independently for the k components: For the ith component,
  - $((q, 0), \sigma, (q', 0))$ is possible iff $(q, \sigma, q') \in \Delta^i \setminus \Gamma^i$ (i.e., iff (q, σ, q') is a non-accepting transition of $\mathcal{A}_i$),
  - $((q, 0), \sigma, (q', 1))$ is possible iff $(q, \sigma, q') \in \Gamma^i$ (i.e., iff, (q, σ, q') is an accepting transition of $\mathcal{A}_i$),
  - $((q, 1), \sigma, (q', 1))$ is possible iff $(q, \sigma, q') \in \Delta^i$ (i.e., iff (q, σ, q') is a transition of $\mathcal{A}_i$), and
- for all transitions $(q_1, b_1; \ldots; q_k, b_k; \sigma; q_1', b_1'; \ldots; q_k', b_k') \in \Delta'$ with $\sum_{i=1}^{k} b_k' \neq 0$, Γ contains a transition $t = (q_1, b_1; \ldots; q_k, b_k; \sigma; q_1', 0; \ldots; q_k', 0)$ (obtained by replacing all Boolean values in the target state by 0), with $v(t) = (b_1', b_2', \ldots, b_k')$; Γ contains no further transitions.

That is, Δ' simply collects the information, which of the individual accepting transitions has been seen since the last transition from Γ has been taken. Taking a transition from Γ then "cashes in" on these transitions, while resetting the tracked values to 0. As a minor optimisation, we remove the states $\bigtimes_{i=0}^{k-1} Q^i \times \{1\}$ together with the transitions that lead to them. This can be done as there is never a point in delaying to cash in on these transitions. Offering such additional choices that should never be taken would likely impede, rather than help, learning.

Correctness. We now state and prove the correctness of the reduction.

Theorem 4.2. *Given an MDP $\mathcal{M}$ and k ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, it holds that maximising these k objectives with lexicographic order, and maximising them with the product automaton $\mathcal{P}$ defined in Definition 4.1 with infimum or independent semantics provides the same result.*

Proof. It follows from Reference [22] that pure finite-memory strategies suffice for MDPs with lexicographic Büchi objectives. We therefore fix an arbitrary finite-memory strategy μ for the control of the MDP $\mathcal{M}$ with lexicographic Büchi objectives, obtaining a Markov chain $\mathcal{M}_\mu$. Note that this is only a control for the MDP, not of the witness automaton $\mathcal{P}$.

We first turn any pure strategy for $\mathcal{P}$ with independent semantics into a strategy for the individual $\mathcal{A}_i$, which yields the same expected vector for every run (and thus the same expected probability vector). This is quite simple: Every individual $\mathcal{A}_i$ can simply behave like its component in $\mathcal{P}$. On every run, if the ith component of the lexicographic vector is 1, then $\mathcal{A}_i$ has seen infinitely many accepting transitions.

Next, we turn k individual pure finite-memory strategies for the individual $\mathcal{A}_i$ into a strategy for $\mathcal{P}$ and evaluate it with the infimum semantics. For this, the automaton essentially follows the component strategies, and only has to additionally decide when to "cash in." $\mathcal{P}$ will make this choice whenever all individual automata have reached an end-component on the product of $\mathcal{M}_\mu$ and the automaton, and, for all $\mathcal{A}_i$ that are in an accepting end-component, the Boolean store in the ith component is set to 1, or would have been set in this move. Note that this means that $v(t)$ in this case indicates all those components, for which the individual $\mathcal{A}_i$ is in an accepting end-component whenever an accepting transition occurs. In situations where none of the $\mathcal{A}_i$ are in an accepting end-component, we do not utilise accepting transitions. Apart from this, accepting transitions are used only when all Boolean values would otherwise be 1 (as cashing in becomes obligatory in such cases).

With this strategy, the valuation can only differ from the individual valuations of the $\mathcal{A}_i$ if either (at least) one of the $\mathcal{A}_i$-s never reaches an end-component, or if it eventually reaches an accepting end-component, but only visits accepting transitions finitely often. As both of these events have probability 0, the expected vectors are the same for the individual $\mathcal{A}_i$ and for $\mathcal{P}$ with this strategy. Finally, for every strategy the expected value for the independent semantics is at least as high as the value for the infimum semantics. □

4.2 From Lexicographic-Büchi Objective to Weighted-Büchi Automaton

In this section, we observe that the previous theorem translates smoothly to a scalar version of the previous translation.

Weighted-Büchi Automata. A weighted-Büchi automaton is an ω-automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$ along with a weight function $w : \Gamma \to \mathbb{R}$, which maps each accepting transition to a positive weight. As in the lexicographic-Büchi condition, the value of a run is 0 if no accepting transition occurs infinitely often. If accepting transitions do occur infinitely often, and they do occur in the order $t_1, t_2, t_3, \ldots \in \Gamma^\omega$, then the value of the run is $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(t_i)$.

For a linear function $f : \mathbb{R}^k \to \mathbb{R}$ with only positive coefficients (i.e., linear functions that grow strictly monotonically in each dimension), the weighted-Büchi automaton $\mathcal{P}_f = (\Sigma, Q, q_0, \Delta, \Gamma, f \circ v)$ is such that $\Sigma, Q, q_0, \Delta, \Gamma$, and v are identical to the product automaton $\mathcal{P}$ from Definition 4.1.

Theorem 4.3. *Given an MDP $\mathcal{M}$ and k lexicographic ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, it holds that maximising these k objectives, weighted by some $f : \mathbb{R}^k \to \mathbb{R}$ with only positive coefficients, and maximising them with the automaton $\mathcal{P}_f$ provides the same result.*

Proof. The proof of Theorem 4.2 merely needs to be extended by the observation that, for every run r, with value b^+ in the independent semantics, b^- in the infimum semantics, and w_f in the weighted semantics, $f(b^+) \geq w_f \geq f(b^-)$ holds. Recalling that the same expected vectors can be obtained using the individual $\mathcal{A}_i$, $\mathcal{P}$ with independent semantics, and $\mathcal{P}$ with infimum semantics, all these maximisations provide the same result. □

Lemma 4.4. *Let μ be any optimal finite-memory strategy for $\mathcal{M} \times \mathcal{P}_f$. Then $(M \times \mathcal{P}_f)_\mu$ never has two transitions $t, t' \in \Gamma$ with $v(t) \neq v(t')$ in an end-component reachable from its initial state.*

Proof. Assuming such transitions t and t', the strategy can be improved by playing an adjusted strategy that mimics the strategy in the end-component (once reached), except that it only plays an accepting transition when all Boolean values that occur in this end-component in the independent semantics, are set. As this increases the expected reward, it contradicts the optimality of the end-component. □

Corollary 4.5. *There exists optimal pure memoryless strategies for $\mathcal{M} \times \mathcal{P}_f$.*

We complete this reduction by showing that, for carefully chosen f, optimising the expected reward provides an optimal policy for $\mathcal{P}$ (for both semantics).

Theorem 4.6. *For a given MDP $\mathcal{M}$ and k GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, there is a linear function f such that an optimal pure strategy for $\mathcal{M} \times \mathcal{P}_f$ is also optimal on $\mathcal{M} \times \mathcal{P}$. (Note that $\mathcal{M} \times \mathcal{P}$ and $\mathcal{M} \times \mathcal{P}_f$ have the same states and strategies, and that Lemma 4.4 entails that the value for both semantics of $\mathcal{P}$ is the same.)*

Proof. Since the set of positional strategies is finite, there is a minimal difference $p_{\min} > 0$ between the probabilities to achieve an individual objective by any two positional strategies with a different probability of meeting this objective.

Let $f_k = 1$, pick any $f_i \geq (1 + 1/p_{\min}) f_{i+1}$ for all $i < k$, and let $\overrightarrow{f} = (f_1, f_2, \ldots, f_k)$. Now, consider any two positional strategies that obtain probability vectors of satisfying the properties $\overrightarrow{p} = (p_1, p_2, \ldots, p_k)$ and $\overrightarrow{p}' = (p'_1, p'_2, \ldots, p'_k)$, respectively, such that $\overrightarrow{p} > \overrightarrow{p}'$. We claim that the value of $\overrightarrow{p} \cdot \overrightarrow{f}^T$ is at least $p_{\min}$ higher than $\overrightarrow{p}' \cdot \overrightarrow{f}^T$. This is because, assuming that the ith position is the first one where $\overrightarrow{p}$ and $\overrightarrow{p}'$ differ, we get

$$\begin{aligned}
\overrightarrow{p} \cdot \overrightarrow{f}^T - \overrightarrow{p}' \cdot \overrightarrow{f}^T &\geq p_{\min} f_i - \sum_{j>i} f_j \geq p_{\min}(1 + 1/p_{\min}) f_{i+1} - \sum_{j>i} f_j \\
&= p_{\min} f_{i+1} - \sum_{j>i+1} f_j \geq \cdots \geq p_{\min} f_k = p_{\min}.
\end{aligned}$$

Moreover, if $\sum_{i=1}^{k} f_i \cdot \varepsilon < p_{\min}$, then using the weights of $\overrightarrow{f}$ for the reachability will guarantee that a strategy with a better performance obtains a better value and, in particular, only optimal strategies can obtain the highest value. □

4.3 From Weighted Preference to Weighted-Büchi Automaton

We now discuss the reduction of k GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ with preference given by weight vector $w = (w_1, \ldots, w_k)$ to a weighted-Büchi automaton $\mathcal{P}_f$. This reduction is direct. The automaton $\mathcal{P}_f = (\Sigma, Q, q_0, \Delta, \Gamma, f \circ v)$ is such that Σ, Q, q_0, Δ, Γ, and v are identical to the product automaton $\mathcal{P}$ from Definition 4.1, and $f(v) = w^T v$.

Theorem 4.7. *Given an MDP $\mathcal{M}$ and k ω-regular objectives given as GFM Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ with weighted preference given by weight vector $w = (w_1, \ldots, w_k)$, it holds that maximising the weighted preference objective, and maximising them with the weighted-Büchi automaton $\mathcal{P}_f$ provide the same result.*

Proof. The proof of Theorem 4.2 can be extended to establish this result. As mentioned in Reference [22], pure finite-memory strategies are sufficient for MDPs with weighted preference Büchi objectives. When translating an optimal pure finite-memory policy for the weighted preference objective to the automaton $\mathcal{P}_f$, the decision to "cash in" is done by waiting for the end-components on the product to be reached, and selecting for $\mathcal{A}_i$ that are in that accepting end-component. An optimal pure strategy on $\mathcal{P}_f$ can be translated back for the individual $\mathcal{A}_i$ by having it behave like its respective component in $\mathcal{P}_f$. □

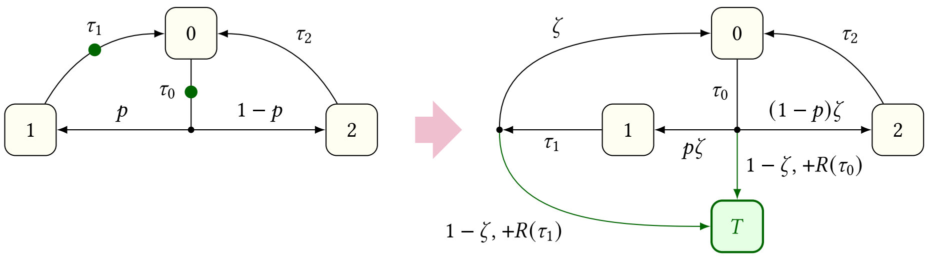

Fig. 4. Adding transitions to the target (T) in the augmented product MDP. The only edges that give non-zero reward are marked with a transition dependent $+R(\cdot)$ reward. In the original translation [35], all such rewards were equal to 1.

4.4 Combining Lexicographic and Weighted Preferences

Using the results from this section, one can combine a lexicographic ordering between levels of importance, while—within each level—allowing for fixed weights between properties.

Definition 4.8 (Lexicographic-Weighted ω-regular RL). Given an MDP $\mathcal{M}$ with unknown transition structure and ℓ sets of k_i (for $i = 1,\ldots,\ell$) GFM Büchi automata $\mathcal{A}_{1,1},\ldots,\mathcal{A}_{1,k_1}$; … ; $\mathcal{A}_{\ell,1},\ldots,\mathcal{A}_{\ell,k_\ell}$ accepting ω-regular objectives $\varphi_{1,1},\ldots,\varphi_{1,k_1}$; … ; $\varphi_{\ell,1},\ldots,\varphi_{\ell,k_\ell}$, compute a strategy optimal for the *lexicographic weighted ω-regular objective* $(\mathcal{A}_{1,1},\ldots,\mathcal{A}_{1,k_1}$; … ; $\mathcal{A}_{\ell,1},\ldots,\mathcal{A}_{\ell,k_\ell})$, that is, a strategy that maximises according to the lexicographic order of the vector $(\sum_{i=j}^{k_1} p_{1,j}w_{1,j}$; … ; $\sum_{j=1}^{k_\ell} p_{\ell,j}w_{\ell,j})$, where $p_{i,j} = \mathrm{PSem}^{\mathcal{M}}_{\mathcal{A}_{i,j}}(s_0)$ is the probability that $\mathcal{M}$ satisfies $\varphi_{i,j}$ and $w_{i,j} > 0$ is a weight that describes the relative importance of $\varphi_{i,j}$ within the ith level.

The above problem can be solved with the methods developed so far by selecting suitable weights for each automaton. As before, this can be done by scaling the weights by sufficiently large factors to ensure a lexicographic separation is achieved. The proofs from the previous sections extend fairly easily to this case.

5 FROM WEIGHTED-BÜCHI AUTOMATA TO WEIGHTED REACHABILITY

Generalizing the construction for GFM Büchi automata from References [35, 37], we replace the *fixed* payoff of $+R(\cdot) = 1$ used in the gadget (cf. Figure 4) by a *transition dependent* payoff $f(v(t))$. The gadget is otherwise unchanged: It takes the original transition with probability ζ, and moves, with a probability of $1-\zeta$, to a sink state T—providing a payoff of $f(v(t))$—where the run ends. The payout on these transitions is the only reward that exists in this game, while ζ is a hyperparameter. This modification transforms the problem into a generalised reachability scenario, where the payout occurs only in the last step.

Theorem 5.1. *Given an MDP $\mathcal{M}$ and a weighted-Büchi automaton $\mathcal{P}_f$ there is a $\zeta_0 < 1$ such that, for all $\zeta \in [\zeta_0, 1)$, the optimal strategies obtained from replacing the accepting transitions in $\mathcal{M} \times \mathcal{P}_f$ by the gadget, are optimal for $\mathcal{M} \times \mathcal{P}_f$.*

Proof. We start with expanding the observation from Lemma 4.4 about end-components in optimal solutions to the weighted reachability objective obtained using this gadget: In both MDPs under consideration, $\mathcal{M} \times \mathcal{P}_f$ and the variation where accepting transitions are replaced by the gadget, there cannot be two $t, t' \in \Gamma$ with $v(t) \neq v(t')$—if there were, the expected payoff could be improved as described in the proof of Lemma 4.4.

This leaves the expected difference to be purely down to the part *before* reaching an end-component. Moreover, for every positional strategy, the expected value of the undiscounted payoff

Table 1. Multiobjective Q-learning Results

Name	prop. ord.	states	prod.	prob.	time	f	w	ζ	ε	α	tol	ep-l	ep-n
`Bet`	1, 2	25	400	0.71,0.68	3.25	—	(0.5,0.5)	0.95	0.2	$0.1^\star$	0		60k
`Bet`	1, 2	25	400	0.84,0.44	8.11	—	(0.7,0.3)	0.95	0.2	$0.07^\star$	0		150k
`R 4×4`	1, 2, 3	16	1024	1,0,1	6.37		—	0.7					65k
`R 4×4`	2, 1, 3	16	1024	1,0,1	1.95		—	0.7		0.03			75k
`R 5×5`	1, 2, 3	25	1600	1,1,1	12.49		—	0.5	0.2	$0.4^\star$	0	250	200k
`R 4×7`	1, 2, 3	28	3584	1,0.75,0.25	23.18	20	—	0.9	0.2	$0.9^\star$	0	400	100k
`R 4×7`	2, 1, 3	28	3584	1,0.5,0	10.62	20	—	0.7	0.2	0.15		200	200k
`R 4×7`	1, 2, 3	28	3584	1,0.5,1	11.90	—	(20,1,1)	0.7	0.2	$0.9^\star$	0	400	150k
`Virus`	1, 2	809	58248	1,0	41.43		—	0.97	0.5	0.2		50	150k
`Virus`	2, 1	809	58248	1,0.25	191.97		—	0.97	0.5	$0.9^\star$		50	700k
`UAV`	1, 2	11,448	732672	1,1	93.57	20	—		0.2			40	120k
`Bridges`	1, . . . , 7	19	5318784	—	3.55	—	(1,. . .,1)	0.9		0.2			30k
`Tracking`	1, 2	41,472	497664	1,1	97.35		—		0.3	0.1	0	100	300k

Blank entries indicate that the default values were used. The default values are: $f = 10$, $\zeta = 0.99$, $\epsilon = 0.1$, $\alpha = 0.1$, tol = 0.01, ep-l = 30, and ep-n = 20,000. The w column indicates the weights used in a non-lexicographic weighting. Parameters that were linearly decayed to zero over training are indicated with $\star$. For `Bridges` the sum of the probabilities of satisfaction of the seven properties is 6. Times are in seconds.

is between the value obtained by removing the payoff for the gadgets not in an end-component, and increasing the payoff for these gadgets to the maximal payoff. As the expected difference between these two extremes approaches 0 as ζ approaches 1, for sufficiently large $\zeta < 1$, optimal strategies for weighted reachability in the model with gadgets also serve as optimal strategies for the mean payoff objective of $\mathcal{M} \times \mathcal{P}_f$. □

We note that weighted reachability lacks the contraction property [58, 65], which makes it theoretically unsuitable for Q-learning. Learners often wrap reachability (and undiscounted payoff) into a discounted version thereof. This adds another parameter γ, which should be chosen significantly closer to 1 than ζ [e.g., $1 - (1 - \zeta)^2$].

6 EXPERIMENTAL RESULTS

We implemented the construction described in Sections 4 and 5 in Mungojerrie [39], a C++ tool that reads MDPs described in the PRISM language [47] and ω-regular automata written in the HOA format [3]. In our implementation, we have introduced a component known as the "tracker," which keeps track of the accepting edges observed for each property. Our implementation is done on-the-fly, where we keep track of the states of the MDP, the automata, and the tracker, separately and compose them together at each timestep. The agent has additional actions to "cash in" and to control any nondeterminism in the automata.

6.1 Experimental Setup

We conducted a series of experiments using Q-learning on various case studies, and the results are presented in Table 1. The table provides details about each example, including the prioritisation order of properties, the number of states in the MDP and the product, the probability of satisfaction for each property under the learned strategy, and the execution time in seconds.

We also report the values of the parameter f (which encodes the linear function for $\overrightarrow{f} = (f, 1)$ from Section 4.2 when we have two objectives, and for $\overrightarrow{f} = (f^2, f, 1)$ when we have three), the weight vector w used if the weights used are not of this form, the parameter ζ, the exploration rate ε, and the learning rate α. The discount factor γ was 0.999 for all the examples in Table 1.

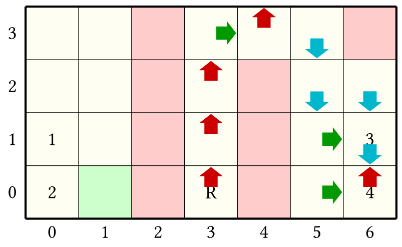

Fig. 5. Learned strategy on the 4×7 Robot grid with property prioritisation order 2, 1, 3. We show only reachable states and project the states in the product to the MDP. The "cash in" action is not shown for clarity.

The table also shows the relative difference under which action values are considered the same during model checking of the learned strategy (tol), the maximum number of timesteps between accepting edges before the episode is reset (ep-l), and the number of training episodes (ep-n). Parameters were tuned manually to minimise training time. All experiments were performed on a machine running Ubuntu with an Intel i7-8750H processor and 16 GB of RAM. The runtime for each experiment remained below 4 min, and the memory usage did not exceed 1 GB.

6.2 Case Studies

Next, we provide a short discussion for each case study in the table.

Double Bet (`Bet`). For the double bet example in the introduction, we use Q-learning with $\zeta = 0.9$, $\varepsilon = 0.2$, $\alpha = 0.06$ linearly decayed to 0 over training, tol = 0, $\gamma = 0.995$, ep-l = 30, and ep-n = $500k$ with various values for a. The objectives in LTL are:

(v_1) $\mathbf{G}\neg h_1$: Coin 1 is never heads.
(v_2) $\mathbf{F}h_2$: Coin 2 is eventually heads.

Figure 1 shows the learned Pareto curve showing the maximal probabilities of satisfaction obtainable for each property.

Robot (`R 4×4, R 5×5, and R 4×7`). For the robot example from the introduction in Figure 2 with $\mathbf{p} = 0.5$, we have considered instances for the three scenarios discussed there, with different priorities of them. On examples `R 4×4` and `R 5×5`, we initialise learning episodes randomly within the model to deepen exploration to achieve better performance. On the `R 4×7` example with property 1 given the highest priority and properties 2 and 3 given the same priority, we give a weighting of $(f, 1, 1)$ with $f = 20$. The results are in line with the analysis provided in the introduction. Figure 5 shows an optimal strategy learned for the 4×7 Robot grid with property prioritisation order 2, 1, 3.

To avoid clutter, we show only states that are reachable under the learned strategy and do not show the "cash in" action. Additionally, we project states in the product to the MDP. The robot starts in Row 0, Column 3 and moves up the column. Then, in Row 3, Column 3 it attempts to go across. With probability 0.5, it gets stuck in Row 3, Column 4. There are no safe actions from this field except to move north—keeping the robot in the same field. If it gets across to Row 3, Column 5, then it moves down and repeatedly visits the fields labelled 3 and 4.

Computer Virus (`Virus`). This case study, based on Reference [48], focuses on the spread of a virus in a computer network. The structure of the model is depicted in Figure 6, where circles represent network nodes and lines represent network connections through which the virus can spread. In this scenario, an attack has a probability of $1 - \mathbf{p}_{\text{detect}} = 0.5$ of bypassing the firewall.

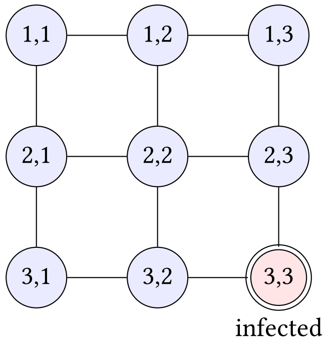

Fig. 6. Network model for virus spread in computer networks.

If an attack successfully bypasses the firewall, then it can infect the network with a probability of $\mathbf{p}_{\text{infect}} = 0.5$. The control of the attacks is centralised, and there is always a no-operation action available. The instance coordinating the attack has the following objectives:

(v_1) $\mathbf{F}s_{3,2} = 2 \bigwedge \mathbf{G}((s_{3,2} = 2 \wedge s_{3,1} \neq 2) \implies \mathbf{XX}s_{3,1} = 2)$: Eventually, node $(3,2)$ gets infected; and when $(3,2)$ is infected and $(3,1)$ is not, $(3,1)$ is infected within 2 steps.

(v_2) $\mathbf{F}s_{1,1} = 2 \bigwedge \mathbf{G}(s_{2,2} \neq 2 \wedge s_{2,3} \neq 2)$: Eventually node $(1,1)$ gets infected while nodes $(2,2)$ and $(2,3)$ never get infected.

When the prioritisation order is 1, 2, the optimal strategy crosses the barrier formed by nodes $(2,2)$ and $(2,3)$ to infect node $(3,1)$ before node $(3,2)$. When the prioritisation order is 2, 1, then the optimal strategy respects the barrier formed by nodes $(2,2)$ and $(2,3)$, and follows the path through node $(3,1)$ to $(1,1)$. This reduces the probability of satisfying v_1 from 1 to 0.25. This problem is particularly challenging, because it requires discovering a long sequence of actions and the properties interfere with each other during learning.

Human-in-the-Loop UAV Mission Planning (UAV). This model (originally from Reference [25]), considers the control of an unmanned aerial vehicle (UAV) interacting with a human operator. The UAV operates on a network of connections. In this network, waypoints (w_i) are specified as well as restricted operation zones (roz_i). In specifications, waypoints serve as places that shall be visited (once or repeatedly), while restricted operation zones shall be avoided. In our experiments, we have used the following properties.

(u_1) $\bigwedge_i \mathbf{G}\neg roz_i$: UAV never visits a restricted operation zone.

(u_2) $\mathbf{F}w_1 \wedge \mathbf{F}w_2 \wedge \mathbf{F}w_6$: UAV eventually visits w_1, w_2, and w_6.

Seven Bridges of Königsberg (Bridges). We also consider the classic problem of the seven bridges of Königsberg [23], in which one seeks a path that crosses each bridge exactly once. It has been shown that the configuration of the bridges limits one to cross at most six bridges exactly once. The model is deterministic, and we have seven properties of the form

(v_i) $\mathbf{F}b_i \wedge \mathbf{G}(b_i \rightarrow \mathbf{XG} \neq b_i)$: Cross bridge i exactly once.

where b_i indicates if one is on bridge i. Instead of applying a preference for each property (bridge), we give each an equal weighting of 1. In this setting, payoff is maximised when the sum of the probability of satisfaction of all properties is maximised, e.g., the agent maximises the expected number of bridges crossed exactly once. The RL agent successfully finds strategies to cross six bridges exactly once, the maximum number possible.

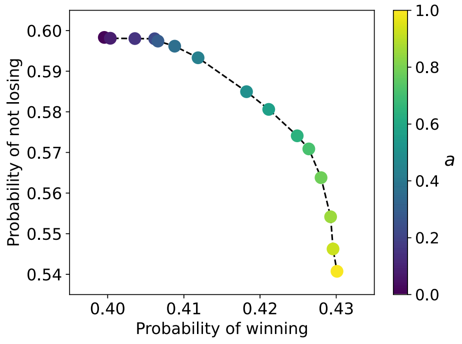

Fig. 7. Approximate Pareto curve for the blackjack example, computed via learning.

Tracking (`Tracking`). In this model, a spotlight needs to track the random movements of an object in a grid, while being confined to the grid's border cells. The two specifications are

(v_1) $\mathbf{FG}b$: Eventually dwell in the border cells b forever,
(v_2) $\mathbf{GF}h$: Have the spotlight hit the object infinitely often.

This combination of objectives is satisfied by many strategies, which accounts for the relative speed with which a strategy is learned in this case.

Blackjack (`Blackjack`). This case study is a slightly simplified game of blackjack. We examine the combination of two objectives:

(v_1) $\mathbf{F}w$: Win the game,
(v_2) $\mathbf{F}\neg l$: Do not lose the game.

Different strategies maximise the probability of winning and the probability of not losing (which includes drawing the game). By assigning weights to these objectives, one may explore a spectrum of strategies from the most cautious to the most aggressive.

We approximate the Pareto curve by using Q-learning with $\zeta = 0.5$, $\varepsilon = 0.9$, $\alpha = 0.3$ decayed linearly to 0 over training, tol $= 0$, $\gamma = 0.995$, ep-l $= 20$, and ep-n $=$ 1,000k. We weight the objective for winning with a and the objective for drawing with $1 - a$, for 15 equally spaced values of a between 0.01 and 0.99. The MDP in this example has 3,829 states and the product has 61,264 states. Learning took approximately 25 s per point.

6.3 Discussion

The experimental results demonstrate the feasibility of our reward translation approach for handling multi-objective ω-regular specifications in tabular learning settings. Table 1 shows that the running time for each example is only weakly correlated with the number of states in that example. Instead, we find that the running time is more closely related to the difficulty of discovering the correct strategy. In learning optimal strategies for multiple objectives, our technique is able to discover unintuitive strategies (Figure 5) and learn approximate Pareto curves (Figures 1 and 7).

7 RELATED WORK

MDP Optimisation. The study of optimal control problems for MDPs under various performance objectives is a well-established area of research, as discussed in Reference [58]. For a given MDP and a specific performance objective such as total reward, discounted reward, or average reward, the optimal expected cost can be characterised using Bellman equations. Furthermore, an optimal strategy can be computed using dynamic programming techniques like value iteration, policy iteration, or linear programming [58]. Chatterjee, Majumadar, and Henzinger [17] considered MDPs with multiple discounted reward objectives. In the presence of multiple objectives, the trade-offs between different objectives can be characterised as Pareto curves. The authors of Reference [17] showed that every Pareto optimal point can be achieved by a memoryless strategy and the Pareto curve can be approximated in polynomial time. Moreover, the problem of checking the existence of a strategy that realises a value vector can be decided in polynomial time. These multi-objective optimisation problems were studied in the context of multiple long-run average objectives by Chatterjee [15]. He showed that the Pareto curve can be approximated in polynomial time in the size of the MDP for irreducible MDPs and in polynomial space in the size of the MDP for general MDPs. Additionally, the problem of checking the existence of a strategy that guarantees values for different objectives to be equal to a given vector is in polynomial time for irreducible MDPs and in NP for general MDPs.

Probabilistic Model Checking. Verification of stochastic systems against ω-regular requirements has received considerable attention [5, 47]. For systems modelled as MDPs and requirements expressed using ω-regular specifications, the key verification problem "probabilistic model-checking" is to compute optimal satisfaction probabilities and strategies. The probabilistic model checking problem can be solved [21] using graph-theoretic techniques (by computing so-called accepting end-component and then maximising the probability to reach states in such components) over the product of MDPs and ω-automata. Etessami et al. [22] were the first to study the multi-objective model-checking problem for MDPs with ω-regular objectives. Given probability intervals for the satisfaction of various properties, they developed a polynomial-time (in the size of the MDP) algorithm to decide the existence of such a strategy. They also showed that, in general, such strategies may require both randomisation and memory. That article also studies the approximation of the Pareto curve with respect to a set of ω-regular properties in time polynomial in the size of the MDP. Forejt et al. [27] studied quantitative multi-objective optimisation over MDPs that combines ω-regular and quantitative objectives. Those algorithms are implemented in the probabilistic model checker PRISM [47].

Reward Translation in RL. RL has recently been applied to finding optimal control for ω-regular objectives [10, 28, 35, 37, 40, 41, 46, 60], but all of theses papers deal with a single objective. Recently, Reference [9] considered using RL to perform the lexicographic maximisation of a safety property, followed by an ω-regular objective, followed by a discounted reward objective. The approach of Reference [9] (and, more recently, of Reference [64]) uses multiple tables from Q-learning to gate the allowed actions for optimising a particular objective based on higher priority objectives. This does not allow the use of multiple ω-regular objectives, since multiple ω-regular objectives may require additional memory, which we supply in our construction. In this way, the results of Reference [9] are complementary to ours. Our method allows an arbitrary number of ω-regular objectives with arbitrary (potentially non-lexicographic) weights that can be combined into a single reward. By treating this combination as a single objective at the highest priority, one can use the method of Reference [9] to add a discounted reward objective at the lowest priority. The resulting policy lexicographically optimises the weighted/lexicographic ω-regular objectives followed by the discounted reward objective, which follows from the results of Reference [9].

Safety in RL. Correct-by-construction synthesis, as described in Reference [5], is an approach to the design of safety-critical systems that emphasises the integration of formal proof-of-correctness with the automatic refinement of formal specifications. This approach considers the environment as adversarial and employs tools from competitive game theory to design theoretically optimal, if over-cautious, systems. In contrast, RL provides an alternative perspective on the environment, treating it as a stochastic player with an unknown strategy. RL employs adaptive sampling-based techniques to converge toward an optimal policy while adapting to changes in the environment. However, RL relies on the Markovian assumption and lacks guarantees on system safety during the learning process. Shielding, as introduced in References [1, 45], combines the benefits of correct-by-construction synthesis and RL. It integrates guarantees from correct-by-construction synthesis with the adaptive nature of RL. Although lexicographically ranked objectives often prioritise safety components in specifications, it's important to note that our work does not specifically address the concept of safe learning. Safe learning focuses on developing learning approaches that avoid violations of safety objectives even during the training phase, as explored in References [1, 14, 30, 43, 57, 63].

8 CONCLUSION

This article generalises a recent work on applying reinforcement learning to finding optimal control for MDPs with ω-regular objectives to address the problem of controlling MDPs with multiple ω-regular objectives with lexicographic and weighted preferences. Starting with good-for-MDPs automata, the extension is surprisingly simple: It suffices to add a "tracker" to the product of the individual automata, which memorises which of the individual Büchi conditions have occurred and to allow the automaton to "cash in" on these occurrences (while resetting the memory). How valuable the return is depends on the objectives, for which a good event (the passing of an individual Büchi transition) has been witnessed, where the main objective attracts a high reward and each consecutive objective obtains a small fraction of the reward assigned to the previous more important one. Such a reward structure can be proven to work for reinforcement learning in essentially the same way as it has been proven to work for the simpler case of having a single objective (which is also a special case of our results).

This reduction is beautifully straightforward, and our experimental results show that it is also effective. This might look surprising: After all, the correctness proofs rely heavily on well chosen hyperparameters. Yet, in practice, the techniques have yet again shown to be robust: We could learn correct strategies reliably and efficiently.

REFERENCES

[1] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2669–2678.

[2] D. Andersson and P. B. Miltersen 2009. The complexity of solving stochastic games on graphs. In *Algorithms and Computation*. 112–121.

[3] T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. 2015. The Hanoi ω-automata format. In *Proceedings of the International Conference on Computer Aided Verification (CAV'15)*. 479–486. LNCS 9206.

[4] Ch. Baier and M. Größer. 2005. Recognizing ω-regular languages with probabilistic automata. In *Proceedings of the Conference on Logic in Computer Science (LICS'05)*. 137–146.

[5] Ch. Baier and J.-P. Katoen. 2008. *Principles of Model Checking*. MIT Press.

[6] Stefan Banach. 1922. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae* 3, 1 (1922), 133–181. http://eudml.org/doc/213289.

[7] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. 2009. Better quality in synthesis through quantitative objectives. In *Proceedings of the International Conference on Computer Aided Verification (CAV'09)*. Springer, 140–156.

[8] Y.-Lan Boureau and Peter Dayan. 2011. Opponency revisited: Competition and cooperation between dopamine and serotonin. *Neuropsychopharmacology* 36, 1 (2011), 74–97.

[9] Alper Kamil Bozkurt, Yu Wang, and Miroslav Pajic. 2021. Model-free learning of safe yet effective controllers. Retrieved from https://arXiv:2103.14600.

[10] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'20)*. 10349–10355. https://doi.org/10.1109/ICRA40945.2020.9196796

[11] Véronique Bruyere, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. 2017. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Info. Comput.* 254 (2017), 259–295.

[12] Véronique Bruyère, Quentin Hautem, and Jean-François Raskin. 2017. Parameterized complexity of games with monotonically ordered ω-regular objectives. Retrieved from https://arXiv:1707.05968.

[13] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. 2019. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the Joint Conference on Artificial Intelligence.* 6065–6073.

[14] S. Carr, S. Junges, N. Jansen, and U. Topcu. 2022. Safe reinforcement learning via shielding under partial observability. Retrieved from https://arxiv.org/pdf/2204.00755.pdf.

[15] Krishnendu Chatterjee. 2007. Markov decision processes with multiple long-run average objectives. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, V. Arvind and Sanjiva Prasad (Eds.). Springer, Berlin, 473–484.

[16] Krishnendu Chatterjee, Joost-Pieter Katoen, Maximilian Weininger, and Tobias Winkler. 2020. Stochastic games with lexicographic reachability-safety objectives. In *Proceedings of the International Conference on Computer Aided Verification.* Springer, 398–420.

[17] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. 2006. Markov decision processes with multiple objectives. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science.* Springer, 325–336.

[18] Indraneel Das and John E. Dennis. 1997. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Struct. Optimiz.* 14, 1 (1997), 63–69.

[19] Nathaniel D. Daw. 2003. *Reinforcement Learning Models of the Dopamine System and their Behavioral Implications.* Carnegie Mellon University.

[20] Nathaniel D. Daw, Sham Kakade, and Peter Dayan. 2002. Opponent interactions between serotonin and dopamine. *Neural Netw.* 15, 4-6 (2002), 603–616.

[21] L. de Alfaro. 1998. *Formal Verification of Probabilistic Systems.* Ph.D. Dissertation. Stanford University.

[22] K. Etessami, M. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. 2007. Multi-objective model checking of Markov decision processes. In *Tools and Algorithms for the Construction and Analysis of Systems*, Orna Grumberg and Michael Huth (Eds.). Springer, Berlin, 50–65.

[23] Leonhard Euler. 1956. The seven bridges of Königsberg. *World Math.* 1 (1956), 573–580.

[24] E. A. Feinberg and A. Shwartz (Eds.). 2002. *Handbook of Markov Decision Processes.* Springer.

[25] Lu Feng, Clemens Wiltsche, Laura R. Humphrey, and Ufuk Topcu. 2015. Controller synthesis for autonomous systems interacting with human operators. In *Proceedings of the ACM/IEEE 6th International Conference on Cyber-Physical Systems (ICCPS'15)*, Alexandre M. Bayen and Michael S. Branicky (Eds.). ACM, 70–79. https://doi.org/10.1145/2735960.2735973

[26] Peter C. Fishburn. 1974. Exceptional paper–Lexicographic orders, utilities, and decision rules: A survey. *Manage. Sci.* 20, 11 (1974), 1442–1471.

[27] Vojtech Forejt, Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. 2011. Quantitative multi-objective verification for probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, Parosh Aziz Abdulla and K. Rustan M. Leino (Eds.). Springer, Berlin, 112–127.

[28] J. Fu and U. Topcu. 2014. Probably approximately correct MDP learning and control with temporal logic constraints. In *Proceedings of Robotics: Science and Systems—A Robotics Conference (RSS'14).*

[29] Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári. 1998. Multi-criteria reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML'98)*, Vol. 98. Citeseer, 197–205.

[30] J. Garcia and F. Fernández. 2015. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* 16 (2015), 1437–1480.

[31] Alessandro Giuseppi and Antonio Pietrabissa. 2020. Chance-constrained control with lexicographic deep reinforcement learning. *IEEE Control Syst. Lett.* 4, 3 (2020), 755–760.

[32] Paul W. Glimcher. 2011. Understanding dopamine and reinforcement learning: The dopamine reward prediction error hypothesis. *Proc. Natl. Acad. Sci. U.S.A.* 108, supplement 3 (2011), 15647–15654. https://doi.org/10.1073/pnas.1014269108

[33] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning.* MIT Press.

[34] E. M. Hahn, G. Li, S. Schewe, A. Turrini, and L. Zhang. 2015. Lazy probabilistic model checking without determinisation. In *Proceedings of the International Conference on Concurrency Theory (CONCUR'15).* 354–367.

[35] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. 2019. ω-Regular objectives in model-free reinforcement learning. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19).* 395–412. LNCS 11427.

[36] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2020. Faithful and effective reward schemes for model-free reinforcement learning of ω-regular objectives. In *Proceedings of the*

18th International Symposium on Automated Technology for Verification and Analysis (ATVA'20) (Lecture Notes in Computer Science), Dang Van Hung and Oleg Sokolsky (Eds.), Vol. 12302. Springer, 108–124. https://doi.org/10.1007/978-3-030-59152-6_6

[37] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. 2020. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'20).*

[38] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2021. Model-free reinforcement learning for lexicographic ω-regular objectives. In *Proceedings of the International Symposium on Formal Methods*. Springer, 142–159.

[39] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2023. Mungojerrie: Reinforcement learning of linear-time objectives. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'23).* Retrieved from https://plv.colorado.edu/wwwmungojerrie/.

[40] M. Hasanbeig, A. Abate, and D. Kroening. 2018. Logically-correct reinforcement learning. Retrieved from http://arxiv.org/abs/1801.08099.

[41] M. Hasanbeig, A. Abate, and D. Kroening. 2019. Certified reinforcement learning with logic guidance. Retrieved from https://arXiv:1902.00778.

[42] A. Hordijk and A. A. Yushkevich. 2002. *Handbook of Markov Decision Processes: Methods and Applications*. Springer, 231–267.

[43] N. Jansen, B. Könighofer, S. Junges, A. Serban, and R. Bloem. 2020. Safe reinforcement learning using probabilistic shields. In *Proceedings of the International Conference on Concurrency Theory (CONCUR'20).* 3:1–3:16.

[44] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. 2019. Model-based reinforcement learning for atari. Retrieved from https://arXiv:1903.00374.

[45] Bettina Könighofer, Florian Lorber, Nils Jansen, and Roderick Bloem. 2020. Shield synthesis for reinforcement learning. In *Proceedings of the International Symposium on Leveraging Applications of Formal Methods*. Springer, 290–306.

[46] Jan Kretínský, Guillermo A. Pérez, and Jean-François Raskin. 2018. Learning-based mean-payoff optimization in an unknown MDP under ω-regular constraints. In *Proceedings of the 29th International Conference on Concurrency Theory (CONCUR'18) (LIPIcs)*, Sven Schewe and Lijun Zhang (Eds.), Vol. 118. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 8:1–8:18. https://doi.org/10.4230/LIPIcs.CONCUR.2018.8

[47] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the Conference on Computer Aided Verification (CAV'11).* 585–591. LNCS 6806.

[48] M. Kwiatkowska, G. Norman, D. Parker, and M.G. Vigliotti. 2009. Probabilistic mobile ambients. *Theor. Comput. Sci.* 410, 12–13 (2009), 1272–1303.

[49] M. E. Lewis. 2002. Bias optimality. In *Handbook of Markov Decision Processes*, E. A. Feinberg and A. Shwartz (Eds.). Springer, 89–111.

[50] T. M. Liggett and S. A. Lippman. 1969. Short notes: Stochastic games with perfect information and time average payoff. *SIAM Rev.* 11, 4 (1969), 604–607.

[51] Chunming Liu, Xin Xu, and Dewen Hu. 2014. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Trans. Syst., Man, Cybernet.: Syst.* 45, 3 (2014), 385–398.

[52] Michael Mandler. 2021. The lexicographic method in preference theory. *Econ. Theory* 71, 2 (2021), 553–577.

[53] Z. Manna and A. Pnueli. 1991. *The Temporal Logic of Reactive and Concurrent Systems Specification*. Springer.

[54] MITtr18. 10 Breakthrough Technologies 2017. Retrieved from https://www.technologyreview.com/10-breakthrough-technologies/2017/. Date accessed: 07-24-2022.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[56] Nvidia20. NVIDIA: Paving the Way for Smarter, Safer Autonomous Vehicles. Retrieved from https://www.nvidia.com/en-us/industries/transportation/. Date accessed: 07-07-2020.

[57] M. Pecka and T. Svoboda. 2014. Safe exploration techniques for reinforcement learning—An overview. In *Proceedings of the International Conference on Modelling and Simulation for Autonomous Systems (MESAS'14).* 357–375.

[58] M. L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

[59] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *J. Artific. Intell. Res.* 48 (2013), 67–113.

[60] D. Sadigh, E. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. 2014. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *Proceedings of the IEEE Conference on Decision and Control (CDC'14).* 1091–1096.

[61] S. Sickert and J. Křetínský. 2016. MoChiBA: Probabilistic LTL model checking using limit-deterministic Büchi automata. In *Automated Technology for Verification and Analysis*. 130–137. LNCS 9938.

[62] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (Jan. 2016), 484–489.

[63] T. D. Simão, N. Jansen, and M. T. J. Spaan. 2021. AlwaysSafe: Reinforcement learning without safety constraint violations during training. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems: AAMAS*. 1226–1235.

[64] J. Skalse, L. Hammond, C. Griffin, and A. Abate. 2022. Lexicographic multi-objective reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'22)*. 3430–3436.

[65] R. S. Sutton and A. G. Barto. 2018. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

[66] Gerald Tesauro, Rajarshi Das, Hoi Chan, Jeffrey Kephart, David Levine, Freeman Rawson, and Charles Lefurgy. 2007. Managing power consumption and performance of computing systems using reinforcement learning. *Adv. Neural Info. Process. Syst.* 20 (2007).

[67] Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *J. Mach. Learn. Res.* 15, 1 (2014), 3483–3512.

[68] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, and Petko Georgiev. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[69] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3-4 (1992), 279–292.

[70] C. J. C. H. Watkins. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation. King's College, Cambridge, UK.

[71] Wayve18. Wayve: Learning to Drive in a Day with Reinforcement Learning. Retrieved from https://wayve.ai/blog/learning-to-drive-in-a-day-with-reinforcement-learning. Date accessed: 11-05-2018.

[72] Kyle Hollins Wray and Shlomo Zilberstein. 2015. Multi-objective POMDPs with lexicographic reward preferences. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*.

Received 26 July 2022; revised 19 May 2023; accepted 11 June 2023