A PRACTICAL GUIDE TO MULTI-OBJECTIVE REINFORCEMENT LEARNING AND PLANNING

A PREPRINT

Conor F. Hayes*
National University of Ireland Galway
Galway, Ireland
c.hayes13@nuigalway.ie

Roxana Rădulescu*
Vrije Universiteit Brussel
Brussels, Belgium
roxana.radulescu@vub.be

Eugenio Bargiacchi
Vrije Universiteit Brussel
Brussels, Belgium

Johan Källström
Linköping University
Linköping, Sweden

Matthew Macfarlane
University of Amsterdam
the Netherlands

Mathieu Reymond
Vrije Universiteit Brussel
Brussels, Belgium

Timothy Verstraeten
Vrije Universiteit Brussel
Brussels, Belgium

Luisa M. Zintgraf
University of Oxford
Oxford, United Kingdom

Richard Dazeley
Deakin University
Geelong, Australia

Fredrik Heintz
Linköping University
Linköping, Sweden

Enda Howley
National University of Ireland Galway
Galway, Ireland

Athirai A. Irissappane
University of Washington
Tacoma, USA

Patrick Mannion
National University of Ireland Galway
Galway, Ireland

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium

Gabriel Ramos
Universidade do Vale do Rio dos Sinos
São Leopoldo, RS, Brazil

Marcello Restelli
Politecnico di Milano
Milan, Italy

Peter Vamplew
Federation University Australia
Ballarat, Australia

Diederik M. Roijers
Vrije Universiteit Brussel, Belgium &
HU University of Applied Sciences Utrecht, the Netherlands

ABSTRACT

Real-world decision-making tasks are generally complex, requiring trade-offs between multiple, often conflicting, objectives. Despite this, the majority of research in reinforcement learning and decision-theoretic planning either assumes only a single objective, or that multiple objectives can be adequately handled via a simple linear combination. Such approaches may oversimplify the underlying problem and hence produce suboptimal results. This paper serves as a guide to the application of multi-objective methods to difficult problems, and is aimed at researchers who are already familiar with single-objective reinforcement learning and planning methods who wish to adopt a multi-objective perspective on their research, as well as practitioners who encounter multi-objective decision problems in practice. It identifies the factors that may influence the nature of the desired solution, and illustrates by example how these influence the design of multi-objective decision-making systems for complex problems.

Keywords Multi-objective decision making · Multi-objective reinforcement learning · Multi-objective planning · Multi-objective multi-agent systems

*These authors contributed equally to this work.

1 Introduction

In most real-world decision problems, we care about more than one aspect. For example, if we have a water reservoir with a hydro-electric power plant we may care about maximising energy production, while minimising irrigation deficits as well as minimising the risk of flooding [Reddy and Kumar, 2006, Pianosi et al., 2013, Castelletti et al., 2013]. In medical treatment, we may want to maximise the effectiveness of the treatment, while minimising a variety of side effects [Jalalimanesh et al., 2017, Laber et al., 2014, Lizotte et al., 2010]. In other words, most real-world decision problems are inherently multi-objective.

While most decision problems actually have multiple objectives, most algorithms dealing with agents that need to interact with decision problems focus on optimising a single objective [Sutton and Barto, 2018]. To nonetheless deal with the multiple objectives of the real world, a common approach to creating decision-theoretic agents is to combine all the important aspects together into a single, scalar, additive reward function. This typically involves an iterative process of assigning numerical rewards or penalties to events that can occur in the environment. For example, in the water reservoir setting, we may put a large penalty on a flood occurring, a positive reward on the power output for each time step, and a negative reward for each time step in which the irrigation demand is not met. Then, the single-objective planning or learning agent is turned on, the resulting policy observed, and then the reward function is re-engineered if the behaviour is not satisfactory. This iterative process is then repeated until the behaviour is acceptable to the designer. We argue that this workflow is problematic for several reasons, which we will discuss in detail one by one: (a) it is a semi-blind manual process, (b) it prevents people who should take the decisions from making well-informed trade-offs, putting an undue burden on engineers to understand the decision-problem at hand, (c) it damages the explainability of the decision-making process, and (d) it cannot handle all types of preferences that users and human decision makers might actually have. Finally, (e) preferences between objectives may change over time, and a single-objective agent will have to be retrained or updated when this happens.

Firstly (a), if we engineer a scalar reward function through an iterative process until we reach acceptable behaviour, we try out multiple reward functions, each of which is a *scalarisation* of the actual objectives. However, we do not systematically inspect all possible reward functions. In other words, we may meet our minimal threshold for acceptable behaviours, but we only observe a subset of all possible scalarisations. Therefore, although an acceptable solution may be found, it can be arbitrarily far away from optimal utility – the use we would have received if we could have systematically examined all possible solutions. This automatically brings us to the second point (b). As the reward function is something that needs to be engineered beforehand, we are only guessing as to the effects this might have on the policy. For example, when trying to train an agent in a power production system, we may wish to double the average power output. However, even if the objectives are linearly weighted in the reward function, it is not as simple as just doubling the reward associated with the power output aspect of performance, as the relationship between the reward weights and the actual objective outcomes may well be non-linear [Van Moffaert et al., 2014]. If, on the other hand, we would be able to inspect all *possibly optimal* policies – and their values offering different trade-offs between the objectives – we could have decided in a well-informed manner on the outcomes, rather than making an educated guess at the scalarisation a priori. This educated guessing is also putting decision power where it does not belong: with the engineers. When an engineer creates a scalar reward function, they are simultaneously making assumptions about the preferences of the actual decision makers (e.g., a government in case of the water reservoir) and making guesses about the behavioural changes resulting from changes to the scalar reward function. This is not a responsibility that can be left to AI engineers – at least not in decision problems that are of significant importance.

Another issue with scalar reward functions is the lack of (a posteriori) explainability (c). If we ask "why did the robot collide with and destroy the vase?", we could try to input an alternative decision, such as swerving away from the vase. An agent with a single all-encompassing objective that has learnt a scalar value function will then, for example, tell us there was a 3.451 reduction in value for this other policy, which provides little insight.

If instead, the agent could have told us that in the objective of damage to property the probability of damaging the vase would have dropped to practically 0, but the probability of running into the family dog increased by 0.5% (a different objective), this would give us insight into what went wrong. We may also disagree for different reasons: we may think that the agent has overestimated the risk of colliding with the dog, which would be an error in the value-estimate for that objective. We might also think that a 0.5% increase in the likelihood of bumping into to the dog would be so small that it would have been acceptable – especially if the robot bumping into the dog would probably have been an inconvenience for the dog, but not an actual danger to it – if the robot could have definitely avoided destroying the vase. This would have been an error in the utility function we assign to different outcomes. In other words, not taking an explicitly multi-objective approach can rob us of essential information that we might need to evaluate or understand our agents.

Furthermore (d), not all human preferences can be handled by scalar additive reward functions [Roijers et al., 2013]. When a user's preferences ought to be modelled with a non-linear rather than a linear utility function, a priori scalarisation becomes mathematically impossible within many reinforcement learning frameworks, as scalarisation would break the additivity of the reward function. For some domains, this might still be acceptable, as the resulting loss of optimality may not have a major impact. However, in important domains where ethical or moral issues become apparent, single-objective approaches require explicitly combining these factors together with other objectives (such as economic outcomes) in a way that may be unacceptable to many people [Wallach and Allen, 2008]. Similarly, designing single-objective rewards may be difficult or impossible for scenarios where we wish to ensure fair or equitable outcomes for multiple participants [Vamplew et al., 2018, Siddique et al., 2020].

Finally (e), humans are known to change their minds from time to time. Therefore, preferences between trade-offs in the different objectives may well change over time. An explicitly multi-objective system can train agents to be able to handle such preference changes, thereby preempting the need to discover a new policy whenever such changes occur. This increases the applicability of multi-objective decision-making agents, as agents do not need to be taken out of operation to be updated and they can simply switch policy to match the new user preferences. We note that this type of change is different from the issue of non-stationary dynamics of the problem which can occur in both single-objective and multi-objective problems; here the multi-objective Markov decision process (Section 3) itself is stationary, but the external preferences have changed.

An insight into the difference between single-objective and multi-objective approaches to an application can be gained by comparing two different studies applying RL to wet clutch engagement [Van Vaerenbergh et al., 2012, Brys et al., 2013]. The task is to control the piston in a wet clutch so as to produce a fast and smooth engagement, by minimising both the time to engagement and the torque loss. The initial study uses a scalar reward with discounting which implicitly captures both aspects of the desired behaviour and achieves acceptable results [Van Vaerenbergh et al., 2012]. However, the subsequent study examines the policies arising from several different utility functions and parameterisations of those functions and demonstrates that some of these are superior to those reported in the original work [Brys et al., 2013].

By now, we hope we have convinced you, the reader, that taking an explicitly multi-objective approach to planning and learning may be essential to deploying AI in decision problems. To provide further motivation, as well as showcase some difficulties that can arise when modelling problems with multiple objectives, we will provide examples of such multi-objective decision problems in Section 2. We then proceed with formalising multi-objective problems (Section 3) and recommend an approach to systematically deal with multi-objective decision problems that puts the user's utility front-and-centre throughout the entire process (Section 4). In Section 5 we outline which factors should be taken into account in the process from identifying a multi-objective decision problem to deploying a policy for it in practice. We describe the effects of these factors on both this process and on the solution concepts. We then proceed to describe the relationships between multi-objective decision problems and other known decision problems (Section 6), and briefly survey both algorithmic approaches (Section 7) and the metrics for evaluating the solutions produced by these algorithms (Section 8). To help researchers get started in the field, we include a worked-out example of a multi-objective decision problem, a water management problem with multiple objectives, in Section 9, furthermore, we added a Jupyter notebook [Kluyver et al., 2016] with these worked-out examples as supplementary material. Finally, we conclude the article and discuss open research challenges in Section 10.

Our purpose with this article is to provide an introduction to multi-objective decision making and guide the reader through getting started with modelling and solving such decision problems. This article differs from existing surveys in the literature that aim to provide a comprehensive overview of methods and theory, in that it is designed to be a guide for practitioners and researchers, highlighting the issues that need to be considered and addressed when applying multi-objective agents to practical problems. As a follow-on reading, we recommend the more technical survey provided by Roijers et al. [2013].

2 Motivating examples of modelling complex problems with multi-objective approaches

This section presents examples of complex decision-making situations where multi-objective approaches play a role. These examples motivate some of the aspects discussed in later sections.

2.1 Planning a journey

Consider you need to travel from your house to a given destination. Deciding on the modes of transportation typically involves a number of objectives, such as minimising travel time and cost whereas maximising comfort and reliability [Ortúzar and Willumsen, 2011, Ramos et al., 2020, Mannion et al., 2016]. For instance, car trips may be faster and

more comfortable than subway ones, at the cost of being more expensive and less reliable (at least in cities that easily get congested due to e.g. an accident). Moreover, given the competitive nature of traffic, your objectives are usually affected by other users, which increases the uncertainties associated with your decision. In spite of such uncertainties, if you can express your preferences over these different objectives as a linear combination, then you can make your decision using conventional optimisation approaches. However, if (as is often the case) you cannot articulate your preferences explicitly in a single formula, or you actually can, but this formula is non-linear, you have a genuine multi-objective problem, which requires a multi-objective approach (see details in Section 5.3).

In order to select the best multi-objective approach, different factors come into play. If you execute this journey every day, you might be interested in balancing your objectives on average over a longer period. However, your intention might also be to balance the objectives during each of the single journeys, which would require a different approach. Both views would result in one policy that tells you how to plan your journey. Nonetheless, at some occasions, you might want to balance the objectives differently because you have an important meeting or you have someone accompanying you on your journey. If you want to be prepared for this, you can apply a method that provides you with a variety of policies, each of which is optimising a different combination of the objectives involved. In this situation, you could easily adjust each single trip based on your current needs. In contrast, conventional optimisation approaches would need to recompute the policy from scratch in order to handle such changes.

2.2 Water management

Water reservoir operations need to handle multiple competing objectives related to significant socio-economic impacts [Castelletti et al., 2008]. By regulating a system of dam gates placed at the outlet of a lake you can modulate the water release and the level of the lake. On the one hand, you will need to supply water to downstream users to meet their agricultural needs. To achieve this, you need to store water during the winter and spring in order to release it during the irrigation season. On the other hand, stakeholders on the shores are interested in keeping the lake level within a certain range to avoid floods and support recreational activities or environmental services. Increasing the lake storage to avoid irrigation deficits means increasing the risk of flooding and therefore some compromise needs to be established. The regulation problem is complicated by the presence of other objectives that interact with the two above: hydropower production, flood mitigation for downstream users, lake navigability, and many others [Reddy and Kumar, 2006, Pianosi et al., 2013, Castelletti et al., 2013]. A multi-objective analysis is a fundamental tool for the human operator and for the representatives of the various stakeholders to properly evaluate the possible trade-offs among the several conflicting objectives and to support their decisions.

2.3 Military purchasing

The manufacture and purchasing of military equipment requires long term dynamic planning [Nguyena and Caoa, 2017]. Each type of equipment takes a varying degree of time to manufacture. For example, a truck may only need a week while a submarine may need more than ten years. Furthermore, the time and cost in setting up a manufacturing pipeline will require items to be produced in larger numbers. Governments need to make decisions now based on a prediction of the types of environments and operations they expect to deploy forces to in the future. These environments typically require unique combinations of equipment to maximise their ability to achieve the outcomes required. Determining this optimum combination of equipment required for operations ten to fifteen years into the future is a multi-objective planning problem – weighing-up various factors such as the cost, effectiveness, versatility, and protection provided to personnel. In practice, this becomes a problem with many objectives, when considering details such as the selected features of each piece of equipment. For instance, Beliakov et al. [2019] discusses some seventeen objectives (related to survivability, lethality, mobility, and knowledge) to be considered when simply purchasing a single tank.

Furthermore, in the real world, any initial decision made must also be constantly altered over subsequent years. These alterations may be instigated by a change in: government; national priorities; international dynamics; technology; expected operational environments; and, types of operations. No government can make a decision now and expect it is still optimal in fifteen years. Therefore, new plans are developed periodically that align with new predictions. These new predictions can be represented as selecting a new policy from a pre-computed set of solutions. However, governments must be very careful about when to continue; when to cancel; and when to switch manufacturing and purchasing orders. Changing policy directions can incur substantial financial penalties due to ramp-up and down costs; create periods of unbalanced forces during the switching period; require extra personnel training costs; etc. Therefore, a solution to this situation must be able to ensure that an optimal policy is maintained across objectives during the process of changing from one policy to another. This type of dynamic planning situation across multi-objective problems occurs frequently in real-world strategic decision making domains, such as government, business, manufacturing, etc. Hence, the development of robust solutions could support many decision makers.

2.4 Wind farm control

The design of traditional wind turbine control systems is typically focused on two objectives. On the one hand, a wind turbine needs to maximise its power production. On the other hand, maximising power output leads to higher fatigue loads (i.e., the stress induced on the turbine's components), and thus impacts their overall lifetime. Therefore, a trade-off needs to be made between power output and accumulated damage.

Single-turbine control and design has been well-explored in the literature [Abdullah et al., 2012, Menezes et al., 2018]. However, as multiple wind turbines are often geographically grouped into wind farms to reduce capital costs, the turbines become dependent on each other due to the wake effect. This effect occurs when upstream turbines extract energy from wind, leaving a cone of reduced available wind for downstream turbines, harming their productivity. One method to tackle this issue is through wake redirection control, where upstream turbines are purposely misaligned with the incoming wind vector in order to deflect wake away from the downstream turbines [Verstraeten et al., 2020]. However, while misaligned rotors may lead to a higher farm-wide power production, it induces higher loads on the turbine's components. To tackle these non-linearities and complexities that originate from the wake effect, the use of data-driven optimisation methods is necessary to yield optimal wind farm control strategies.

Finding a good balance between power production and loads is challenging. The link between control actions, the high-dimensional load spectrum and future costs is still an open problem [Verstraeten et al., 2019]. Therefore, although the relationship between control actions and maintenance costs is expected to be complex, a linear scalarisation of power production and loads is often employed (e.g., [van Dijk et al., 2016]), where the parameters are decided based on the expertise of operators. Preferably, the operators should receive a set of alternative control strategies to investigate, covering the entire spectrum of objectives ranging from load-focused to power-focused strategies.

2.5 Other topics

In addition to the motivating examples discussed above, recent years have seen multi-objective learning and planning methods applied across a wide range of problem domains including: distributed computing [Qin et al., 2020, da Silva Veith et al., 2019], drug and molecule design [Zhou et al., 2019, Horwood and Noutahi, 2020], cybersecurity [Sun et al., 2018], simulation [Ravichandran et al., 2018], job shop scheduling [Méndez-Hernández et al., 2019], cognitive radio networks [Messikh and Zarour, 2018, Raj et al., 2020], satellite communications [Hu et al., 2020, Ferreira et al., 2019], recommender systems [Lacerda, 2017], power systems [Deng and Liu, 2018, Deng et al., 2020, Wang et al., 2019, Mello et al., 2020], building management [Zhang et al., 2019], traffic management [Jin and Ma, 2019], manufacturing [Govindaiah and Petty, 2019, Lepenioti et al., 2020, Dornheim and Link, 2018], bidding and pricing [Yang et al., 2020, Krasheninnikova et al., 2019], education [Rowe et al., 2018], and robotics [Huang et al., 2019]. The scope and variety of these applications supports our assertion that many important problems involve multiple objectives, and are best addressed using explicitly multi-objective methods.

3 Problem setting

First, let us introduce the basic multi-objective sequential decision problem. We formalise this as a *multi-objective Markov decision process* (MOMDP). We note that more complex models exist, such as a multi-objective partially observable Markov decision process [Soh and Demiris, 2011a,b, Wray and Zilberstein, 2015, Nian et al., 2020] and multi-objective multi-agent systems [Rădulescu et al., 2020a]. However, the MOMDP formalisation allows us to study many relevant aspects of multi-objective decision making problems, while also being simple to understand. We therefore use it as the basis for this article. In this section we will restrict discussion to single-agent MOMDPs and defer discussion of the more complex multi-agent situation until Section 7.2.6.

A MOMDP is represented by the tuple $\langle S, A, T, \gamma, \mu, \mathbf{R} \rangle$, where:

- S is the state space
- A is the action space
- $T\colon S \times A \times S \to [0, 1]$ is a probabilistic transition function
- $\gamma \in [0, 1)$ is a discount factor
- $\mu\colon S \to [0, 1]$ is a probability distribution over initial states
- $\mathbf{R}\colon S \times A \times S \to \mathbb{R}^d$ is a vector-valued reward function, specifying the immediate reward for each of the considered $d \geq 2$ objectives

The crucial difference between a single-objective MDP and a MOMDP is the vector-valued reward function $\mathbf{R}$, which expresses a numeric feedback signal for each of the considered objectives. This means that the length of the reward vector is equal to the number of objectives.

Like single-objective MDP, the state and action sets can in principle be discrete and finite. However, in many real-world problems the state-space is infinite. This happens as soon as some of the state variables describing a state—such as the water levels in a lake (Section 2.2)—are continuous. Moreover, even if the state space is discrete, it often is too large to enumerate as states may be described using images, e.g., cameras in an autonomous car. The action-space can also be infinite in size. For example, in wind farm control (see Section 2.4), actions correspond to a specific rotor orientation with respect to the incoming wind direction. This again is a continuous value. Infinite state- and action-spaces make the problem considerably harder, and necessitate the use of function approximators to estimate policies and their (vector) values.

3.1 Policies and value functions

In MOMDPs, the agent behaves according to a policy $\pi \in \Pi$, where Π is the set of all possible (and allowed) policies. A policy is a mapping $\pi : S \times A \to [0, 1]$, i.e., for any given state, an action is selected according to a certain probability distribution.

The value function of a policy π in a MOMDP is defined as:

$$\mathbf{V}^{\pi} = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{k+1} \mid \pi, \mu\right], \tag{1}$$

where $\mathbf{r}_{k+1} = \mathbf{R}(s_k, a_k, s_{k+1})$ is the reward received at timestep $k + 1$. In contrast to single-objective MDPs, the value function is also vector-valued, $\boldsymbol{V}^{\pi} \in \mathbb{R}^d$. We can also define the value of a state s, for any timestep t, when $s_t = s$:

$$\mathbf{V}^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1} \mid \pi, s_t = s\right]. \tag{2}$$

In single-objective settings, the value functions offer a complete ordering over the policy space, i.e., for any two policies π and π', $V^{\pi}(s)$ will either be greater than, equal to, or lower than $V^{\pi'}(s)$. This implies that finding the optimal policy π^* is equivalent to maximising the expected cumulative discounted reward. For a MOMDP this is not necessarily the case.

If we have access to a *utility function* (also called a *scalarisation function* in the literature) $u : \mathbb{R}^d \to \mathbb{R}$, mapping the multi-objective value of a policy to a scalar value,

$$V_u^{\pi} = u(\mathbf{V}^{\pi}), \tag{3}$$

then this would give us a total ordering over policies and reduce the MOMDP to a single-objective decision making problem. This however, is not always possible, feasible, or desirable as motivated in the introduction. We illustrate this further in Section 5.1.

Thus, when dealing with multi-objective value functions (Equation (2)), it is possible to encounter a situation in which $V_i^{\pi} > V_i^{\pi'}$ for objective i, while $V_j^{\pi} < V_j^{\pi'}$ for objective j. As a consequence, in MOMDPs, value functions only allow for a *partial* ordering over the policy space, so determining the optimal policy is no longer possible without additional information on how to consider or prioritise the objectives to order the policies.

Notice that the formulation of policies described in this section only allows for stationary policies, i.e., we condition only on the current state. While this may be sufficient for fully-observable, single-objective MDPs, White [1982] demonstrate that for multi-objective tasks it may be beneficial to allow policies to be non-stationary with respect to the current state (i.e., also conditioned on other variables, such as the sum of previously received rewards).

3.2 Solution sets

In single-objective RL problems, there exist a unique optimal value V^*, and there can be multiple optimal policies π^* that all have this value. The goal in single-objective RL is typically to learn an optimal policy.

In the multi-objective case however, without any additional information about the user's utility, there can now be multiple *possibly optimal* value vectors $\mathbf{V}$. We therefore need to reason about sets of possibly optimal value vectors and policies when thinking about solutions to MORL problems. In the following, we introduce several useful definitions

for possibly optimal policies and values. We start by defining the most general set of solutions, i.e., the undominated set. This is the set of policies and associated value vectors for which there is at least one utility function for which this policy is optimal (i.e., there is no other policy for this utility function that has strictly higher utility).

The concepts introduced in this section are defined in terms of policies. However, as each policy π has an associated value vector $\mathbf{V}^\pi$, throughout the survey we often relate value vectors to these concepts when the context is clear.

Definition 1 *The undominated set, $U(\Pi)$, is the subset of all possible policies Π and associated value vectors for which there exists a possible utility function u whose the scalarised value is maximal:*

$$U(\Pi) = \left\{ \pi \in \Pi \;\middle|\; \exists u, \forall \pi' \in \Pi : u(\mathbf{V}^\pi) \geq u(\mathbf{V}^{\pi'}) \right\}. \tag{4}$$

However, the undominated set may well contain excess policies. That is, policies that are optimal for a given (set of) utility function(s), but where other policies exist that have optimal utility for that/those utility function(s) as well. In that case, we do not need to retain all policies to retain optimal utility.

Definition 2 *A set $CS(\Pi)$ is a* coverage set *if it is a subset of $U(\Pi)$ and if, for every u, it contains a policy with maximal scalarised value, i.e.,*

$$CS(\Pi) \subseteq U(\Pi) \wedge \left(\forall u, \exists \pi \in CS(\Pi), \forall \pi' \in \Pi : u(\mathbf{V}^\pi) \geq u(\mathbf{V}^{\pi'}) \right). \tag{5}$$

As mentioned, there generally does not exist a total ordering over the values of possible policies in a MORL problem. We can, however, again reason about sets of *possibly optimal* policy values. For the most general case where u is any (potentially unknown) monotonically increasing utility function (i.e., including non-linear functions), we define the set of undominated values as follows.

Definition 3 *If the utility function u is any monotonically increasing function, then the* Pareto Front (PF) *is the undominated set [Roijers et al., 2013]:*

$$PF(\Pi) = \{ \pi \in \Pi \mid \nexists \pi' \in \Pi : \mathbf{V}^{\pi'} \succ_P \mathbf{V}^\pi \}, \tag{6}$$

where $\succ_P$ is the Pareto dominance relation,

$$\mathbf{V}^\pi \succ_P \mathbf{V}^{\pi'} \iff (\forall i : \mathbf{V}^\pi_i \geq \mathbf{V}^{\pi'}_i) \wedge (\exists i : \mathbf{V}^\pi_i > \mathbf{V}^{\pi'}_i). \tag{7}$$

In words, the Pareto Front is the set of non-dominated policies: for each policy in the Pareto Front, there exists no other policy with value that is equal or better in *all* objectives.

Note that for the Pareto front this means we only need to retain one of the policies that have the same value vector. A set of policies whose value functions correspond to the PF is called a *Pareto Coverage Set (PCS)*.

If the (a priori unknown) utility function is a positively-weighted linear sum, then the undominated set will be the policies corresponding to the convex hull (CH) of value functions V^π.

Definition 4 *A linear utility function computes the inner product of a weight vector $\mathbf{w}$ and a value vector $\mathbf{V}^\pi$*

$$u(\mathbf{V}^\pi) = \mathbf{w}^\top \mathbf{V}^\pi. \tag{8}$$

Each element of $\mathbf{w}$ specifies how much one unit of value for the corresponding objective contributes to the scalarised value. The elements of the weight vector $\mathbf{w}$ are all positive real numbers and constrained to sum to 1.

Definition 5 *The* convex hull (CH) *is the subset of Π for which there exists a $\mathbf{w}$ (for a linear u) for which the linearly scalarised value is maximal, i.e., it is the undominated set for linear utility functions:*

$$CH(\Pi) = \{ \pi \in \Pi \mid \exists \mathbf{w}, \forall \pi' \in \Pi : \mathbf{w}^\top \mathbf{V}^\pi \geq \mathbf{w}^\top \mathbf{V}^{\pi'} \}. \tag{9}$$

In words, the convex hull is the set of policies that maximise the weighted sum over objectives for some weight vector $\mathbf{w} \in \mathbb{R}^d$.

The Pareto Front and the Convex Hull often consist of infinitely many policies, especially when policies can be stochastic.

However, coverage sets can often be significantly smaller. This is particularly so for the *convex coverage set*.

Definition 6 *A set $CCS(\Pi)$ is a* convex coverage set *if it is a subset of $CH(\Pi)$ and if for every $\mathbf{w}$ it contains a policy whose linearly scalarised value is maximal, i.e., if:*

$$CCS(\Pi) \subseteq CH(\Pi) \wedge \left(\forall \mathbf{w}, \exists \pi \in CCS(\Pi), \forall \pi' \in \Pi : \mathbf{w}^\top \mathbf{V}^\pi \geq \mathbf{w}^\top \mathbf{V}^{\pi'}\right). \tag{10}$$

The CCS is not only important for linear utility functions. Specifically if we also allow stochastic policies in Π, a CCS is sufficient to construct a CS for all possible (non-linear) monotonically increasing utility functions as well, i.e., a PCS [Vamplew et al., 2009].

For deterministic stationary policies, the difference between the $CH(\Pi)$ and a $CCS(\Pi)$ is often small. Therefore, the terms are often used interchangeably. The key difference however is stochastic policies. Specifically, if the space of deterministic policies is discrete (i.e., there are a finite number of states for which a finite number of actions can be chosen) then there is always a finite CCS, even if stochastic policies are allowed. In contrast, the CH is typically infinite in this case. This is especially important because, as we have already mentioned, this finite CCS can be used as a basis to construct every policy in a PCS. For more detailed information on these sets, and how they interact with deterministic/stochastic policy spaces, please refer to [Roijers et al., 2013].

The choice of solution set is key to the efficiency of the algorithms used to solve multi-objective problems. This is because we have to compute all the policies in these sets. When these sets become too large, we may not be able to compute them anymore, and we need to solicit more information on how to handle or prioritise the objectives. We consider that this optimisation process should be driven by the utility obtained by the user from a proposed solution which can be derived using the utility function. We introduce this perspective and approach in the following section.

4 The utility-based approach

Considering the user utility first is key to the successful application of any AI in decision problems. In multi-objective problems, it is especially important, as the properties of the user's utility may drastically alter the desired solution, what methods are available, and even—in some cases [Rădulescu et al., 2020b]—whether stable solutions even exist. Following the recent literature on multi-objective RL, we therefore support the *utility-based* approach [Rădulescu et al., 2020b,a, Roijers et al., 2013, Roijers and Whiteson, 2017, Zintgraf et al., 2015].

The utility-based approach stands in contrast to the earlier axiomatic approach. In the axiomatic approach the optimal solution set to a multi-objective decision problem is assumed to be the Pareto front (see Definition 3). However, this set is typically too large, and may be prohibitively expensive to retrieve. Furthermore, as Vamplew et al. [2009] have shown, if stochastic polices are allowed, a much smaller solution set suffices to construct a Pareto-front, i.e., we can use stochastic mixtures between the policies in the deterministic stationary convex coverage set (CCS), which is much easier to compute, and allows for algorithms that exploit the properties of the CCS to retrieve the optimal policies, such as outer loop methods [Roijers, 2016] as discussed further in Section 7.2.3. Moreover, in practical applications, a lot more might be known about the utility function of the user, due to domain knowledge. Using an axiomatic approach would make it difficult to exploit this knowledge, and a lot of time and effort might be spent on computing an approximate solution which contains solutions with very low utility for the user/deployment.

The utility-based approach aims to derive the optimal solution set from the available knowledge about the utility function of the user, and which types of policies are allowed. This knowledge allows constraints to be placed on the solution set, reducing its size and thereby improving learning efficiency and making it easier for users or systems to select their preferred policy [Roijers et al., 2013]. The utility-based approach entails the following steps:

1. Collect all a priori available information regarding a user's utility.
2. Decide which type of policies (e.g., stochastic or only deterministic) are allowed.
3. Derive the optimal solution concept from the resulting information of the first two points.
4. Select or design a MORL algorithm that fits the solution concept. A variety of algorithms suited to differing solution concepts are reviewed in Section 7.
5. When multiple policies are required for the solution, design a method for the user to select the desired policy among these optimal policies.

We note that some of these steps can be done in parallel, as illustrated in the work flow diagram in Figure 1. Specifically, it is possible to gather information on the user's utility and decide on which types of policies to allow simultaneously (Steps 1 and 2). However, Steps 1 and 2 need to be completed to be able to derive the solution concept (Step 3), which in turn needs to be completed before being able to select or design an appropriate algorithm (Step 4), and design how the user can select policies (Step 5).

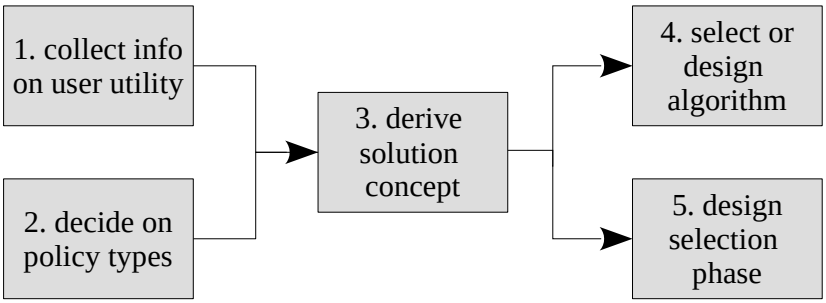


Figure 1: Work flow diagram for multi-objective reinforcement learning and planning.

In each of the steps to complete in this process, different factors will come into play. We will briefly discuss which factors must be considered at each step, while referring to later sections for a more detailed discussion.

In Step 1, we aim to collect as much a priori available information about the user's preferences as possible. This information will help us determine the class of utility functions which we should employ. For example, if we know that all objectives correspond to units of goods that we need to buy or sell on an open market, the utility function will typically be linear (i.e., a sum of prices per unit, times the amount of units we need to buy and can sell).

Another key distinction we have to make here concerns the application of the utility function for deriving the user's utility [Roijers et al., 2013, 2018b, Rădulescu et al., 2020b,a]. Specifically, if the utility is derived from single outcomes of performing the policy, we need to apply the utility function to the returns, and then optimise the expected utility of the returns. This is the so-called Expected Scalarised Returns (ESR) criterion. For example, in a medical treatment planning setting, the patients will derive their utility from their specific treatment outcomes. Conversely, if the utility is derived from the average returns over multiple runs we should take the expectation first, and optimise the utility of expected returns. This is called the Scalarised Expected Returns (SER) criterion. For a detailed discussion on whether to apply ESR or SER, please refer to Section 5.3.

In Step 2, we need to decide what types of policies are allowed. This is important, as in contrast to single-objective problems, stochastic policies can be strictly better than deterministic policies [Vamplew et al., 2009, White and Kim, 1980, White, 1982]. However, this does not mean that we should always allow them. For example, in a medical treatment planning setting, the patients would probably object to random selection of different medicines. Furthermore, we need to decide whether to allow non-stationary policies or not [White, 1982]. For a detailed discussion on policy types, please refer to Section 5.2.3.

Using the information from Step 1 and 2, we need to derive the appropriate solution concept (Step 3). For example, if the utility function is unknown at learning time, but known to be linear, any type of policy is allowed. We need a set of policies that contains at least one optimal policy for every possible set of linear weights. An example where this situation would arise would be where the linear weights correspond to fluctuating market prices of different commodities.

In Step 4, we need to either select an existing algorithm from the literature or design one that is suitable for the user's requirements. The choice of algorithm depends on the solution concept selected in Step 3; one of the main distinctions is between single-policy and multi-policy algorithms (see Section 7.2). If the user utility function is completely known a priori and is not likely to change over time, a single-policy algorithm is appropriate. Conversely, if the utility function is unknown or subject to change a multi-policy algorithm is more suitable.

In Step 5, the goal is to help the user select a policy from a solution set produced by the algorithm selected in Step 4, that comes as close as possible to optimal user utility. This might be relatively straightforward if this set is small enough to show all possible policy value vectors to the user. If the set is large, or even continuous, more intricate methods are needed. For example, Zintgraf et al. [2018] use Gaussian processes to model the utility function, and use relative preferences queries posed to the user to train this model. Furthermore, they use targeted priors and additional (virtual) data to exploit the fact that utility functions in multi-objective decision problems are monotonic in all objectives.

Together, these steps form a complete pipeline to set up a multi-objective reinforcement learning or planning system.

5 Factors influencing the design of multi-objective systems

Multiple factors exist in multi-objective problem domains which do not need to be considered for single-objective problems, and these can have important implications for the design of a multi-objective agent. In this section, we identify and describe these factors, and explain the impact they may have on the design.

5.1 Scenarios requiring a multi-objective approach

Some researchers would argue that modelling problems as multi-objective is unnecessary and that all rewards can be represented as a single scalar signal. This implies that it is always possible to convert a MOMDP to a MDP. In order for this conversion to take place, an a priori scalarisation function is required. However, Roijers et al. [2013] show that in certain situations it may be impossible, infeasible or undesirable to perform this conversion. Roijers et al. [2013] present three scenarios in which this can occur as illustrated in (a), (b) and (c) in Figure 2. Additionally, we propose three new motivating scenarios: the interactive decision support scenario (d), the dynamic utility function scenario (e), and the review and adjust scenario (f). Figure 2 shows that each scenario consists of a planning or learning phase in which either a single policy or a solution set of multiple policies is found, and an execution phase in which a single policy is executed, and, in some scenarios, a selection phase in which the policy to be executed is selected.

In the **unknown utility function scenario** (a) [Rădulescu et al., 2020b], a priori scalarisation is undesirable as the utility function is unknown at the time when planning or learning occurs. There is too much uncertainty around the utility that could be received. In this scenario it is preferable to compute a coverage set of policies so as to be able to respond quickly whenever more information is available. In the wind farm control example (Section 2.4), there are two conflicting objectives. The goal is to maximise power output while minimising the required maintenance costs caused by the stress of operation. Specifying the exact preferences for these objectives is difficult since certain circumstances such as storms, the wake effect, and grid instability can affect the lifespan of turbine components. Since the link between these effects and preventive control measures is insufficiently understood, it is important to learn a set of optimal solutions.

In the **decision support scenario** (b), the user's preferences are unknown or difficult to specify. Working with this uncertainty makes it infeasible, if not impossible, to use a priori scalarisation as the user's utility function is unknown. The decision support scenario is almost identical to the unknown utility function scenario. The only difference is during the selection phase, where a set of policies is presented to a user who selects a policy based on their preferences. In the water management example (Section 2.2), the optimal solution for managing a water reservoir depends on many stakeholders and their multiple conflicting objectives. Each stakeholder has their own preferences as to how the water should be managed, with each objective having an effect on different aspects of the businesses operating around the lake as well as the livelihood of those living nearby. Capturing accurate preferences for all stakeholders while taking into account the trade-offs across all objectives would be difficult, if not impossible. Instead, it would be better to learn a set of optimal policies and then make decisions regarding what policy to follow once a collective decision can be made by a local council or government.

The difference between the two scenarios above lies in the selection phase. The first scenario includes a utility revelation step where the utility function is made explicit. In the second scenario on the other hand, the decision relies on the user(s), and the utility function remains implicit in the decision taken. As defining a utility function explicitly is hard (if not infeasible), user selection typically employs the decision support scenario (b).

In the **known utility function scenario** (c), the user's preferences are known. Working with known preferences, we can assume the user's utility function are known at the time of learning or planning, making scalarisation both possible and feasible. However, it can still be undesirable to do so because performing a priori scalarisation can lead to an intractable problem [Roijers et al., 2013, Rădulescu et al., 2020b]. In the wind farm control example (Section 2.4), based on their preferences a user may want to maximise power output while minimising the stress on the turbine's components. Since the user's preferences are known, it is possible to learn a single policy that optimises the user's preferences.

In the **interactive decision support scenario** (d), the agent has to learn about both the preferences of the user and the environment [Roijers et al., 2018a]. Applying a priori scalarisation in this scenario can be both undesirable and infeasible as the utility function of the user may be unknown or may be uncertain. During learning, the agent can elicit preferences from the user, removing uncertainty about the user's utility function. In the planning a journey example (Section 2.1), a user may not be able to accurately specify their preferences. While cost and travel time are preferences that may be easily specified, objectives such as comfort and reliability may be difficult to specify. When planning a journey other trade-offs such as taking a direct route over switching trains or having a lay over may have an impact on the user's utility. At various times during the learning phase a user could be presented with different potential solutions

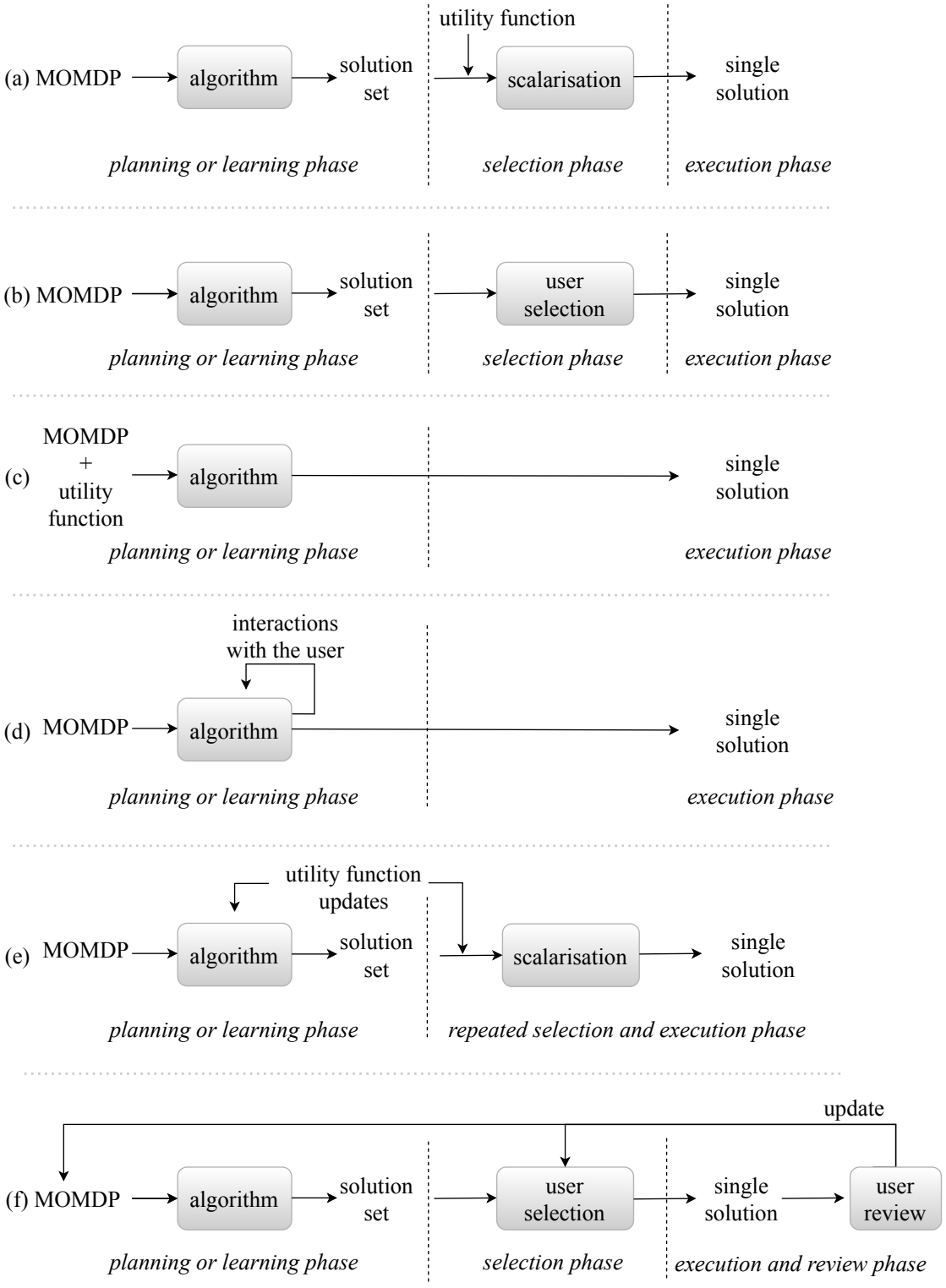


Figure 2: The six motivating scenarios for MOMDPs: (a) the unknown utility function scenario, (b) the decision support scenario, (c) the known utility function scenario, (d) the interactive decision support scenario, (e) the dynamic utility function scenario, and (f) the review and adjust scenario.

and rank the solutions in order of preference. This will enable the system to get a more accurate representation of the users preferences and learn an optimal solution for the users.

In the **dynamic utility function scenario** (e), the user's preferences for certain objectives change over time [Natarajan and Tadepalli, 2005]. In this scenario applying a priori scalarisation would be undesirable. Given that a user's preferences can change over time, it would be optimal for the algorithm to learn a finite number of policies over time and choose an appropriate non-dominated policy for any utility function and improve upon it. Although there is an infinite amount of utility functions, they can be covered by a finite number of policies [Natarajan and Tadepalli, 2005]. In the military purchasing example (Section 2.3), current governments must make decisions about military purchasing, but as governments change over time so do the preferences of each government towards military spending. Using a system that can learn optimal policies for changing preferences is the desired approach for this example. While it would be possible to learn a single policy for the initial utility function and then dynamically adapt this as the user's utility function changes, this would incur a period of sub-optimal behaviour as the agent adapts, which need not occur if the agent has learned in advance a suitable set of solutions. When governments change it is crucial that optimal policies are still followed despite the change in preferences.

In the **review and adjust scenario** (f), a user may be uncertain about their preferences over objectives and their preferences over objectives could change over time. Applying a priori scalarisation in this scenario is unfeasible as there is too much uncertainty around the utility function of the user. In this scenario, learning a coverage set of policies is optimal. Once a coverage set has been learned a user can then select the policy which accurately reflects their preferences. Before execution, the user can review their chosen solution. If the user's preferences have changed, the user can adjust their selected solution to accurately reflect their updated preferences.

The review process can also update the MOMDP which can alter the set of solutions learned. This may for example occur when a new objective is identified, that was previously missed. For example, imagine an agent is used to control traffic in a part of a city. Initially, the pollution levels are seen as a single objective. However, after inspecting a map of the pollution levels resulting from the policies, it turns out pollution levels around a school is relatively high, while a school is actually an area where it ought to be low. In such a case, the pollution objective should be refined, i.e., split into two objectives: one overall, and one for sensitive/key areas such as schools (and e.g., hospitals).

In the planning a journey example from Section 2.1, a user may not be certain about their preferences for comfort and cost, among other objectives. In this scenario a set of optimal solutions is learned and the user selects the solution which accurately reflects their preferences. However, before execution if the user's preferences have changed, the user can review their chosen solution and select an alternative solution which accurately reflects their updated preferences. The system can then be updated to reflect the newly obtained information about the user's preferences.

5.2 Problem taxonomy

Roijers et al. [2013] outline a problem taxonomy which discusses what constitutes an optimal solution for a MOMDP. The taxonomy is based on the utility based approach, where the agent's ultimate goal is to maximise user utility [Rădulescu et al., 2020b]. In the previous section, we highlighted three new motivating scenarios and we have updated the problem taxonomy diagram from Roijers et al. [2013] to include the extended scenarios. The taxonomy in Table 1 outlines how each factor can lead to different solution concepts. It is important to carefully consider each factor in the taxonomy before choosing a solution concept. The factors of the problem taxonomy are covered extensively in [Roijers et al., 2013], but we will briefly outline each factor below.

5.2.1 Single versus multiple policies

Whether or not an algorithm learns a single or multiple policies depends on which of the motivating scenarios holds from Section 5.1. For example, in the unknown utility function and decision support scenarios the agent needs to learn multiple policies. In both of these scenarios the utility function of the user is unknown at the time of learning or planning, and therefore the agent must return a set of optimal policies. In the known utility function scenario the user's utility function is known at the time of learning or planning and therefore returning multiple policies is not necessary.

In the planning a journey example (Section 2.1), a user may or may not know their exact preferences about getting to their destination. A user may be unsure about how they would like to get to their destination or how much they are willing to spend on the journey. In this case we are in the unknown utility function scenario (a) [Roijers et al., 2013] and learning a coverage set of policies is required. In contrast, a user may want to arrive to their destination at a specific time using a specific mode of transport, and may have a fixed budget. Since the user's preferences are known we are in the known utility function scenario (c) [Roijers et al., 2013] and a single policy which represents the user's preferences can be learned.

	single policy (known utility function, interactive decision support)		*multiple policies (unknown utility function, decision support, dynamic utility function, review and adjust)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarisation	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing utility function	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Table 1: Taxonomy of multi-objective decision problems and corresponding solution sets. See Section 3.2 for the definitions of the solution sets.

5.2.2 Linear versus monotonically increasing utility functions

The nature of the utility function has a significant role to play in what constitutes an optimal solution in a MOMDP and which of the motivating scenarios holds. When the utility function is linear the weighted sum for each value of the objectives is computed. In the known utility function scenario, the utility function is known at the time of learning or planning. The utility function can be applied to each reward vector in the MOMDP and an optimal solution can be found. But linear utility functions may not be suitable when trying to express a user's preferences. If a user's preferences are non-linear, a linear utility function is unable to accurately represent these preferences.

A monotonically increasing utility function adheres to the constraint that if a policy increases for one or more of its objectives without decreasing any of the objectives, the scalarised value also increases. A monotonically increasing utility function is able to represents both linear (with non-zero positive weights) and non-linear user preferences. For example, in the unknown utility function scenario, the agent must learn a set of policies. When the utility function is unknown at the time of learning or planning, Pareto dominance can be used to determine a set of non-dominated solutions. Since the utility function is monotonically increasing, policies that are Pareto dominant will be preferred by the user.

In the wind farm control example (Section 2.4) there are two objectives: to maximise power and to reduce fatigue loads on the turbine. In the real world, a user will likely have non-linear preferences over these objectives. If these preferences are known (known utility function scenario) at the time of learning or planning it is crucial they are represented using a non-linear utility function. If the user's preferences are represented using a linear utility function a sub-optimal solution will be learned. A linear utility function cannot accurately represent non-linear preferences [Roijers et al., 2013]. In this case, learning a sub-optimal solution could negatively impact a wind turbines performance. If a wind turbine is not operating optimally, stress on the turbine's components would be increased, which impacts the components lifespan and increases maintenance costs. However, if a non-linear utility function is used to represent the user's preferences then it is possible to learn an optimal solution.

5.2.3 Deterministic versus stochastic policies

Whether to restrict the agent to policies that are deterministic or to allow stochastic policies has a significant impact on what an optimal solution is in a MOMDP. When the utility function is linear, we can translate the MOMDP to a single-objective MDP. In an MDP, only deterministic stationary policies apply as the optimal obtainable value is reachable with deterministic stationary policies. This is true for all linear utility functions. But when the utility function is monotonically increasing and non-linear the situation is much more complex.

For example, in the known utility function scenario, the utility function is known during the learning or planning phase. If the utility function is a linear representation of a user's preferences, we can then translate the MOMDP to a single-objective MDP where only deterministic stationary policies hold. As another example, in the unknown utility function scenario where the utility function is assumed to be non-linear and only deterministic policies are allowed a

coverage set of Pareto dominant policies must be learned. In this scenario, non-stationary policies can Pareto dominate stationary policies [White, 1982], therefore the Pareto coverage set must include non-stationary policies [Roijers et al., 2013].

In the water management example (Section 2.2) there are certain scenarios where stochastic policies should not be considered whatsoever. A stochastic policy where there is a chance the dam gates are opened and all the water in the reservoir is drained should not be considered, even if other outcomes of the policy are optimal. This stochastic policy would have catastrophic outcomes for the nearby town. If the utility function is non-linear and known at the time of learning or planning (known utility function scenario (c)) it would be optimal to learn one deterministic non-stationary policy. In this case, devastating outcomes like the scenario already mentioned would be avoided.

5.3 Scalarised expected returns and expected scalarised returns

In contrast to single-objective reinforcement learning, in multi-objective reinforcement learning (MORL) different optimality criteria exist [Roijers et al., 2013]. Optimising under each criterion can lead to significantly different policies being learned [Rădulescu et al., 2020b], where the criterion chosen for optimisation depends on how the policies are used in practice. The two optimisation criteria are known as the scalarised expected returns (SER) and the expected scalarised returns (ESR).

The SER criterion is the most commonly used optimisation criterion in multi-objective RL and planning [Roijers et al., 2015a]. The SER criterion is calculated by first computing the expected vector returns of a policy and then applying the utility function to this expectation,

$$V_u^\pi = u\left(\mathbb{E}\left[\sum_{i=0}^{\infty}\gamma^t \mathbf{r}_i \,\middle|\, \pi, s_0\right]\right). \tag{11}$$

For SER the utility of a user is derived from multiple executions of a policy. SER is the optimal optimisation criterion in scenarios where the user is concerned about achieving an optimal utility over time. For SER, a coverage set is defined as a set of optimal solutions for all possible utility functions.

However, many scenarios exist where only a single execution of a policy may be relevant to a user. In scenarios where a single execution of a policy is used to derive the utility of a user, optimising under the ESR criterion is optimal [Roijers et al., 2018b]. For example, in a medical setting a patient may have only one opportunity to select a treatment. Under the ESR criterion the utility function is applied to the returns and the expectation is then computed,

$$V_u^\pi = \mathbb{E}\left[u\left(\sum_{i=0}^{\infty}\gamma^t \mathbf{r}_i\right) \,\middle|\, \pi, s_0\right]. \tag{12}$$

For a linear utility function there is no difference in the policies learned for SER and ESR. However, for a non-linear utility function the policies learned under SER and ESR are significantly different [Rădulescu et al., 2020a]. Many RL methods cannot be combined with the ESR criterion because non-linear utility functions in MOMDPs do not distribute across the sum of immediate and future returns which invalidates the Bellman equation [Roijers et al., 2018b],

$$\max_\pi \mathbb{E}\left[u\left(\mathbf{R}_t^- + \sum_{i=t}^{\infty}\gamma^i \mathbf{r}_i\right) \,\middle|\, \pi, s_t\right] \neq u(\mathbf{R}_t^-) + \max_\pi \mathbb{E}\left[u\left(\sum_{i=t}^{\infty}\gamma^i \mathbf{r}_i\right) \,\middle|\, \pi, s_t\right], \tag{13}$$

where $\mathbf{R_t^-} = \sum_{i=0}^{t-1}\gamma^i \mathbf{r}_i$. ESR is the most commonly used optimality criterion in the game theory literature and literature on multi-objective games [Rădulescu et al., 2020a]. However, the ESR criterion has been extensively understudied in the RL literature. To study the ESR criterion further it is essential that new methods are formulated. It is important to note, for the ESR criterion a coverage set has yet to be defined and this is an open area for research.

In the planning a journey example (Section 2.1), we can consider a known non-linear utility function for a user planning their daily commute to work. If the user's employer is flexible about what time the user can start work, then the user can sometimes be late as long as the user is usually on time. Under the SER criterion a policy is optimised on the duration of journey objective. Since this policy is executed everyday it is acceptable for the user to be late some days because the days when the user is early compensate. However, if the user's employer is strict and requires the user to be on time each day or be subject to a fine, it is crucial that the user can plan a daily journey where they arrive on time. In this case optimising under the ESR criterion is optimal since every policy execution must ensure that the user arrives to work on time.

6 The relationship with other problems

Parallels exist between some aspects of multi-objective sequential decision-making tasks and the classes of problems considered by other areas of reinforcement learning and planning research. In this section we identify some of the key areas of overlap between fields where we believe there is potential for beneficial exchange of ideas and techniques. We also pinpoint several pitfalls where the application of methods to multi-objective problems may not be consistent with the utility-based paradigm.

6.1 Partially observable MDPs

A key observation made already in the 1980s, is that if one assumes linear utility functions, POMDPs are a superclass of MOMDPs [White and Kim, 1980]. To see this, imagine there would be a "true objective" and the linear weights of the utility function would form a "belief" over what the true objective would be. This is a special type of POMDP, where there will never be any observations concerning what the "true objective" is – because after all, it does not actually exist.

Of course multi-objective problems and partially observable problems have significantly different interpretations. However, the fact that POMDPs form a superclass of multi-objective MDPs under linear utility has important consequences for researchers and practitioners alike. Firstly, a lot of theoretical properties are inherited from POMDPs. This means that a lot of theorems do not have to be proven anew for MOMDPs under linear utility. So if you are wondering whether a certain property holds, it is prudent to consult the POMDP literature as well. Secondly, it means that methods that have been invented originally for POMDPs, can often be adapted for usage in MOMDPs [Roijers, 2016]. While doing so, it is key to note that the number of objectives in a MOMDP correspond to the number of states in a POMDP (i.e., the dimensionality of the belief- and α-vectors) [Roijers et al., 2015a]. This means that methods that did not work well in a POMDP context because they scale poorly in the number of states, might be very useful in a MOMDP context. A good example of this is Optimistic Linear Support (OLS) [Mossalam et al., 2016, Roijers, 2016, Roijers et al., 2015a], which was based on Cheng's linear support for POMDPs [Cheng, 1988]. Finally, it might mean that some algorithmic improvements may be applicable to both MOMDPs and POMDPs (such as [Roijers et al., 2018c]).

6.2 Multi-objective as multi-agent problems

Objectives are not agents. Some papers—in our opinion abusively—cast single-agent multi-objective problems as multi-agent problems, with each agent representing a competing objective [Li and Czarnecki, 2019, Méndez-Hernández et al., 2019]. Then, either through voting rules [Tozer et al., 2017] or (Nash) equilibria [Duan et al., 2014, Economides et al., 1991, Lee, 2012, Li et al., 2012], a policy is selected. This mechanism however, has no guarantees with respect to user utility. It is unclear whether this "compromise solution" represents a desired trade-off or not. Specifically, the concepts of voting rules and Nash equilibria have been designed to find trade-offs between the individual utilities of agents. This is different from trade-offs for an individual agent between objectives, as objectives can be more or less important and may have non-linear interactions in the utility function. A voting rule or Nash equilibrium is not able to capture such subtleties and can therefore function as no more than an unfounded heuristic. In fact it is well-known that Nash equilibria can be Pareto-dominated [Cohen, 1998, Dubey and Rogawski, 1990].

But altruistic agents can see other agents as objectives. On the other hand, if we consider an agent that is explicitly altruistic, i.e., it cares about the other agents in its environment, such an agent could see the utility of these other agents as objectives, and therefore this should be modelled as a multi-objective problem. As such, it is also possible to consider varying levels of altruism. Aoki et al. [2004] for example, consider a multi-stage flow system with multiple agents. Each agent is a service centre, and is represented as a different objective by the other agents. They use a distributed reinforcement learning framework and propose a bi-directional decision making mechanism to address the resulting multi-objective problem.

Modelling other agents as objectives enables explicitly imposing fairness between these objectives, i.e., the utilities of the agents. A loose condition for fairness is that a joint policy $\boldsymbol{\pi}$ is not so-called Lorenz dominated. Lorenz domination is based on the so-called Lorenz vector [Perny et al., 2013]. The Lorenz vector $\mathbf{L}(\mathbf{V}^{\boldsymbol{\pi}})$ of a vector $\mathbf{V}^{\boldsymbol{\pi}}$ is defined as:

$$\left(v_{(1)},\ v_{(1)} + v_{(2)},\ \ldots,\ \sum_{i=1}^{N} v_{(i)} \right), \tag{14}$$

where $v_{(1)} \leq v_{(2)} \leq \ldots \leq v_{(N)}$ correspond to the values in the vector $\mathbf{V}^{\boldsymbol{\pi}}$ sorted in increasing order. Imposing that the Lorenz vector of a policy is undominated leads to the Lorenz optimal set as coverage set. A vector $\mathbf{V}^{\boldsymbol{\pi}}$ Lorenz

dominates ($\succ_L$) a vector $\mathbf{V}^{\boldsymbol{\pi}'}$ when:

$$\mathbf{V}^{\boldsymbol{\pi}} \succ_L \mathbf{V}^{\boldsymbol{\pi}'} \Leftrightarrow \mathbf{L}(\mathbf{V}^{\boldsymbol{\pi}}) \succ_P \mathbf{L}(\mathbf{V}^{\boldsymbol{\pi}'}), \tag{15}$$

i.e., when the Lorenz vector of $\mathbf{V}^{\boldsymbol{\pi}}$ Pareto dominates the Lorenz vector of $\mathbf{V}^{\boldsymbol{\pi}'}$. A Lorenz optimal set can then be input for a negotiation of which policy to execute in practice [Rădulescu et al., 2020a]. Lorenz optimal sets have been studied in the context of different problem domains [Galand and Lust, 2015, Moghaddam et al., 2011, Perny et al., 2013].

In short, objectives do not typically correspond to the interests of single agents as agents will care about multiple objectives. However, altruistic agents may see the interests of other agents as objectives, and therefore aim to come up with fair solutions.

6.3 Multi- and auxiliary task RL

A highly related problem that has recently gained traction in the RL literature is that of *auxiliary tasks* and *multi-task RL*. For example, Schaul et al. [2015] define multiple goals, which are typically a subset of the states. They then learn a universal value function approximation (UVFA) network, that learns a value with respect to these different goals. From a MORL perspective, UVFA is thus an instance of MORL, with the restriction that goals are closely associated with states, and that the utility function may only select one of these goals to be *the goal* at the moment. To move to a more general MORL setting, a goal should be generalised to a specific a priori known (parameterised) utility function and as such there is no clear relation between the goal (i.e., the importance of each objective) and the state. This would be an issue in multi-policy settings as (a) it is not clear how many specific utility functions would be needed, and (b) non-linear utility functions would not be supported. For the dynamic weights setting with linear utility functions, UVFA has been adapted to the MORL setting as a baseline algorithm [Abels et al., 2019], and shown to perform worse than specific MORL algorithms, but better than more naive baselines.

In their work on successor features (SF), Barreto et al. [2017] decompose a scalar reward into a product of state features and task weights to enable transfer learning between tasks. Again, we observe that successor features are in fact a subclass of multi-objective problems with linear weights, i.e., where objectives can be associated with desirable state-features. Universal Successor Features Approximators [Borsa et al., 2019] and Universal Successor Representations [Ma et al., 2018] combine the benefits of SF and UVFA to further generalise across goals. It is important to note though that while state features and task/goals weights are analogous to the multi-objective reward and linear weight vectors, in MORL the decomposition between reward and weight vectors is typically given rather than learnt. This is because successor features are not observing the individual objectives and are only provided with a scalar reward function. One might think that this would make SF more widely applicable than MORL. However, it also restricts the usage of such methods to things that can be inferred from state features. But, more importantly, scalar reward functions are often engineered on the basis of real events, multiple sensor inputs, and endlessly tweaked on the basis of the actual objectives of the users and designers, as we discussed in Section 1. Hence, using successor features instead of MORL, would in many real-world problems come down to throwing away information first in order to construct a scalar reward function, to later partially infer it back from data. This is of course sub-optimal, and should be avoided if possible.

6.4 Human-aligned agents

As AI systems are increasingly being applied to important real-world tasks, interest has grown in ensuring that the behaviour of autonomous systems is aligned with our own objectives, so as to avoid harmful outcomes either at a general level or with regards to specific individuals. Research within this field focuses on ensuring that the decisions and behaviour of autonomous agents are safe, trustworthy,aligned interpretable, fair and unbiased. As these add additional considerations beyond maximising the agent's primary reward, there is a clear link to multi-objective approaches.

Strong parallels exist between multi-objective decision making and risk-sensitive or safety-aware decision making. An agent making decisions in the context of uncertain risks must aim not just to maximise its expected reward, but also to account for some measure of risk. This measure may be based on the variance of the reward, the worst-case outcome or the probability of entering known error states [Garcıa and Fernández, 2015]. As with the multi-objective methods discussed in this paper, the choice of optimal action for a risk-aware agent will be based on combining together the expected reward and risk measures for each action using some form of utility function. Therefore it is not surprising that several authors have framed safe reinforcement learning as a multi-objective problem. In [Geibel and Wysotzki, 2005], [Geibel, 2006] and [Horie et al., 2019], MORL was applied to develop risk-aware agents, where the risk-related reward is based on the probability of the agent visiting an error state. Meanwhile, Elfwing and Seymour [Elfwing and Seymour, 2017] argue, based on biological evidence, that computational agents may behave more safely if they learn separate values for rewards and punishments.

As well as a growing interest in safe AI, recent years have also seen an increasing focus on the issues of explainability and interpretability of autonomous systems, as these factors are important for building trust with human users, and in ensuring transparency and lack of bias. It has been argued that a reward which has been decomposed from a scalar into its component terms provides benefits from the perspective of explaining decisions [Juozapaitis et al., 2019], and so several recent papers have explored multi-objective approaches to explainable and interpretable RL agents [Noothigattu et al., 2018, Zhan and Cao, 2019, Cruz et al., 2019].

In many applications it is also important to ensure that the actions of an agent are fair with regards to multiple stakeholders—a solution which is optimal for many members of society but which significantly disadvantages a subset of the population may well be unacceptable. In some contexts this may involve the development of multi-agent systems as in our earlier discussion of altruistic agents. In other situations, a single agent may be considering multiple objectives where each objective corresponds to the desires of a particular individual stakeholder, or where each stakeholder may have different preferences over these objectives and the agent must balance the utility obtained by each stakeholder. Multi-objective approaches to fairness have been explored at an abstract level [Siddique et al., 2020], and also within specific applications [Hu et al., 2020, Perez et al., 2010].

In short, following Vamplew et al. [2018, 2021b], we argue that multi-objective agents provide a suitable mechanism for developing human-aligned artificial intelligence, addressing safety constraints as well as other alignment issues such as ethics or legal restrictions.

7 Survey of multi-objective reinforcement learning and planning algorithms

In this section we review the state-of-the-art in algorithms for multi-objective planning and reinforcement learning, relating these algorithms back to the design factors identified in Section 5. The aim is to aid in identifying which extant algorithms may be best suited for a particular application, based on the properties of that application.

7.1 Multi-objective planning algorithms

Research on planning approaches to MOMDPs has been established for much longer than work on reinforcement learning approaches, dating back to at least the early 1980s [White and Kim, 1980, Thomas, 1982, White, 1982]. White and Kim [White, 1982] adapted dynamic programming to develop an algorithm for finding Pareto set policies for infinite horizon discounted MOMDPs. However, as identified by Wiering and De Jong [2007], that approach has issues of computational feasibility and finds policies which are non-stationary. To address this, they developed the CON-MODP algorithm which invokes a consistency operator to ensure the stationarity of policies.

Bryce et al. [2007] demonstrated by example that, in the context of MOMDPs with stochastic state transitions, agents which aim to maximise the SER cannot rely on localised decision-making. The information available at any given state is insufficient to determine the optimal action under the SER formulation, and the agent must also take into account the actions which will be selected, and rewards which will be received, at all other states of the MOMDP. They develop the Multi-objective Looping AO^* ($MOLAO^*$) algorithm to address this issue.

The Convex Hull Value Iteration (CHVI) algorithm [Barrett and Narayanan, 2008] is amongst the most widely-cited works on MOMDP planning. Although it is frequently incorrectly described as a MORL method, it in fact extends Bellman's value iteration algorithm to estimate and store the convex hull of future rewards for each state-action pair. This allows CHVI to identify the coverage set of policies, but only under the assumption that the utility function is linear. Because of the linear utility function, CHVI is akin to planning in POMDPs (see also the relation with POMDPs in Section 6.1). This has recently been shown in a paper that improves both CHVI and POMDP value iteration methods by reusing information across linear programs in subsequent iterations of these methods [Roijers et al., 2018c].

Other planning methods have considered the possibility of specific non-linear definitions of utility. Perny and Wang [Perny and Weng, 2010] address the task of finding the single optimal policy given the goal of minimising the distance between the reward vector received and a target reference point in objective space. They show that the non-linear nature of this utility prevents direct adaptation of methods like dynamic programming which are based on the Bellman equation, and instead develop a non-linear programming solution for this task. Meanwhile, Wray et al. [2015] identify Lexicographic MDPs as a specific subset of MOMDPs, where there is a specified ordering over objectives. They develop methods based on value-iteration for solving such tasks, allowing the ordering of objectives to be state-dependent and incorporating the concept of slack, which allows some degree of loss in the primary objective in order to obtain gains in secondary objectives. This approach has also been extended to POMDPs [Wray and Zilberstein, 2015].

7.2 Multi-objective reinforcement learning algorithms

7.2.1 Stateless/bandit algorithms

Algorithms designed for the multi-armed bandit (MAB) domain endeavour to follow an optimal exploration/exploitation strategy for selecting between different actions (arms), so as to minimise the regret (the loss in reward from not selecting the, initially unknown, optimal action on every time-step). Several papers have examined the extension of MAB algorithms to multi-objective tasks, often by adopting the concept of multi-objective regret in which the agent aims to minimise the number of Pareto-dominated actions which are performed.

Several multi-objective variations to the well-known UCB1 algorithm are compared in [Drugan and Nowe, 2013], including linear and Chebyshev scalarisations, as well as a version based on Pareto dominance. The empirical results show that Pareto UCB1 outperforms the scalarised versions. Later, Yahyaa et al. [2014] demonstrated that a Pareto-based variant of the knowledge gradient algorithm could lead to further improvements in performance over Pareto UCB1.

Other work has examined multi-objective extensions to specialised forms of bandits. Van Moffaert and Nowé [Van Moffaert and Nowé, 2014] consider a multi-objective form of the χ-armed bandit, in which the set of arms is a measurable (potentially infinitely large) set of arms. They propose a modified form of the Hierarchical Optimistic Optimization (HOO) algorithm for this class of bandits. Likewise, Lacerda [Lacerda, 2017] examines multi-objective extensions of ranked bandits, in which the agent produces a ranking of arms rather than a single choice at each time-step. More recently, Turgay et al. [2018] extended the contextual MAB model to incorporate multiple objectives. Unlike conventional MABs, a contextual bandit incorporates some additional state or side-information, and so represents a compromise between stateless bandits and full-blown RL scenarios. Their Pareto Contextual Zooming (PCZ) algorithm aims to minimise the Pareto regret while also maintaining a fair distribution over the Pareto-optimal arms [Bouneffouf et al., 2020].

7.2.2 Single-policy algorithms

Perhaps the simplest and most widely-adopted approach to MORL is to extend existing single-objective model-free value-based methods, such as Q-learning, to handle multiple objectives. This extension requires two changes to the learning algorithm, i.e., the agent must store Q-values as vectors rather than scalars, and the scalarisation function designed to match the user's utility function must be used to identify the greedy-action to perform in any given state. This approach naturally gives rise to single-policy solutions to the multi-objective problem, as the underlying single-objective methods are designed to produce a single optimal solution.

Many applications of this approach have used a linear scalarisation function, either weighted or unweighted [Aissani et al., 2008, Shabani, 2009, Guo et al., 2009, Perez et al., 2009]. This is equivalent to transforming the MOMDP into a corresponding MDP, and so existing proofs of convergence apply [Roijers et al., 2013]. In some domains this will also be a suitable representation of the user's underlying utility (for example, in problems where the objectives are naturally expressed in monetary terms). However, in many cases this linear function will be inadequate to represent the user's true utility [Vamplew et al., 2008]. Therefore it will often be preferable to use a non-linear function instead [Gábor et al., 1998, Van Moffaert et al., 2013a,b, Issabekov and Vamplew, 2012]. Nevertheless, this violates the assumption of additive returns in the Bellman equation at the heart of these algorithms [Roijers et al., 2013], and therefore it may be necessary to condition the Q-values and the agent's choice of action on an *augmented state* formed by concatenating the environmental state with the summed rewards previously received by the agent [Geibel, 2006]. Additionally these approaches may fail to converge to the optimal policy in environments with stochastic state transitions [Vamplew et al., 2021a].

An alternative to these value-based approaches is to adopt a policy-search algorithm. These have the advantage that by optimising at policy level, they can directly optimise with regards to any utility function, including non-linear functions. In addition, they generally produce stochastic policies, which can be beneficial in the context of multiple objectives as discussed earlier in Section 5.2.3. For example, Pan et al. [2020] implement a mixture of long-term policy gradient and short-term planning to find single-policy solutions, while Siddique et al. [2020] develop multi-objective forms of the PPO and A2C policy search methods for the task of finding a single-policy which is fair with regards to all objectives, as measured by the Generalized Gini social welfare function. A substantial number of further multi-objective policy-search methods have been explored in the literature, but much of this work has been in the context of multi-policy approaches and/or deep RL, and so will be discussed further in the later sub-sections.

Under the ESR criterion (Section 5.3) a non-linear utility function is assumed. As already highlighted a non-linear utility function invalidates the assumed additive returns in the Bellman equation. In this case, new methods must be created to efficiently optimise the ESR criterion. Roijers et al. [2018b] implement an Expected Utility Policy Gradient

(EUPG) algorithm which uses Monte Carlo simulations to calculate the sum of the accrued returns and future returns. EUPG optimises over the full returns of an episode as the utility function is applied to the sum of the accrued returns and the future returns. Hayes et al. [2021a,b] propose an algorithm known as Distributional Monte Carlo Tree Search (DMCTS) which learns a posterior distribution over the utility of the returns of a full episode and achieves state-of-the-art performance under the ESR criterion.

7.2.3 Multi-policy approaches

Multi-policy approaches can be divided into two classes. *Outer loop* methods operate on series of single-objective problems, whereas *inner loop* methods consist of algorithms directly designed to produce multiple policies [Roijers and Whiteson, 2017].

The simplest outer loop methods iterate through a series of different parameter settings for a utility function, and re-run a single-policy MORL method for each setting (for example, [Parisi et al., 2014]). The efficiency of outer loop approaches can be improved in two ways. Re-using information from earlier runs rather than discarding this information can reduce learning time [Natarajan and Tadepalli, 2005, Parisi et al., 2017]. Secondly, naive searches through parameter space may re-learn the same policy multiple times, or require a small step-size to ensure all optimal policies are discovered [Roijers et al., 2015b]. More efficient adaptive search methods can reduce the number of iterations of the outer loop [Roijers et al., 2015a,b, Roijers, 2016].

Inner loop methods modify the underlying algorithm to directly identify and store multiple-policies in parallel rather than sequentially. Both Pareto-Q-Learning (PQL) [Van Moffaert and Nowé, 2014] and PQ-learning [Ruiz-Montiel et al., 2017] modify Q-learning to store multiple Pareto-optimal values for each state-action pair. Pruning of dominated values is used to eliminate dominated policies [Mandow and Pérez-de-la Cruz, 2018]. So far these methods are restricted to tabular representation of Q-values, limiting their broader applicability, although the Pareto DQN algorithm [Reymond and Nowé, 2019] provides an initial attempt to integrate PQL and deep RL methods. In the batch setting, Multi-Objective Fitted Q-Iteration (MOFQI) [Castelletti et al., 2012] extends the Fitted Q-Iteration algorithm [Ernst et al., 2005] to the multi-objective case by adding to the state the linear scalarisation weights. MOFQI learns with a single training process an approximation of the optimal Q-function for all the combinations of the scalarisation weights.

Multiple authors have developed inner-loop multi-policy methods based on multi-objective extensions of Monte Carlo Tree Search. The decision about which branch of the tree to expand at any point is determined based on either the hypervolume metric, or on a measure based on Pareto-dominance [Wang and Sebag, 2012, 2013, Perez et al., 2013, Chen and Liu, 2019, Weng et al., 2020].

Model-based methods have clear benefits in the context of multi-policy learning, as once a model of the environment has been learned, it can be used to derive the optimal policy for any utility function with no requirement for further interaction with the environment. Despite this, there has been surprisingly little research so far in model-based MORL. Wiering et al. [2014] provide an approach which learns all Pareto-optimal policies by first learning a model, and then applying the CON-MDP multi-objective dynamic programming algorithm [Wiering and De Jong, 2007]. However, this approach is limited to learning stationary, deterministic policies for deterministic environments. The approach of Yamaguchi et al. [2019] can be applied to stochastic environments. It learns a model which stores reward occurrence probability (ROP) vectors rather than Q-values, and then uses the inner product of the ROP vector and a given weight vector to find the expected reward for the optimal policy for that weight vector. This approach avoids the need to perform an extensive search of the weight space to identify optimal policies. However, it is limited to finding deterministic policies under linear scalarisation, and is designed for maximising the average reward rather than the cumulative discounted return.

In order to learn in domains with continuous state-action spaces and where the state is not fully observable, policy search or actor-critic algorithms are usually considered [Deisenroth et al., 2013]. In the literature, both outer loop [Parisi et al., 2014] and inner loop [Parisi et al., 2016, Giuliani et al., 2016, Parisi et al., 2017] approaches have been proposed to extend policy search methods to multi-objective problems. The approach of Parisi et al. [2017] is interesting in that it constructs a continuous rather than discrete approximation of the Pareto front.

Population-based evolutionary methods are well-suited to finding multiple policies, as each individual can represent a policy which is optimal for a different set of utility preferences. The field of multi-objective evolutionary optimisation is already very well established [Trivedi et al., 2016, Antonio and Coello, 2017, Falcón-Cardona and Coello, 2020], and several researchers have applied concepts from this area to MORL tasks. Evolutionary methods can either be applied directly [Cheng, 1988, Parisi et al., 2017], or combined in a hybrid algorithm with local hill-climbing [Soh and Demiris, 2011a], policy-gradient [Xu et al., 2020] or actor-critic methods [Chen et al., 2020].

7.2.4 Interactive approaches

The majority of MORL methods take either an *a priori* approach to policy selection where user's preferences must be specified prior to learning, or an *a posteriori* approach where a set of policies are learned and then presented to the user for selection. A third alternative is to allow the user to interactively specify their preferences during the learning process, as first proposed in [Vamplew et al., 2011, p. 63]. This allows the user to make a more informed decision based on the agent's discoveries about the nature of achievable trade-offs between objectives, while also allowing earlier convergence to the user's preferences which is important in online learning. An example is the Q-steering algorithm [Vamplew et al., 2015, 2017b]. The user specifies initial preferences in terms of a target point in objective space, and the agent learns a non-stationary mixture of linear-scalarised base policies which minimises the distance between the average reward and the target. A visualisation of the returns of the base policies can be provided to the user, who may then revise their choice of target. The agent can immediately adapt to such changes.

Some work builds on methods for single-objective reinforcement learning with human guidance, and extends those methods to multi-objective problems. Wanigasekara et al. [2019] propose an algorithm that learns user utility functions from observations of user-system interactions for multi-objective contextual bandit based personalized ranking of search results. Ikenaga and Arai [2018] propose to use inverse reinforcement learning for elicitation of user preferences in multi-objective sequential decision making, while Saisubramanian et al. [2020] use human feedback through random queries, approval, corrections, and demonstrations to learn policies that avoid negative side effects.

A systematic approach to simultaneous learning about the environment and the user was proposed by Roijers et al. [2017] for multi-objective multi-armed bandits. Specifically, the *interactive Thompson sampling (ITS)* algorithm uses queries to solicit preferences resulting from linear utility functions while interacting with the environment. For this, it employs Bayesian logistic regression to learn about the utility function, and uses the uncertainty estimates about the utility function to decide which queries to ask. The *Gaussian-process Utility Thompson Sampling (GUTS)* algorithm [Roijers et al., 2020] does the same for any continuous utility function by using Gaussian processes to model the utility function, and estimate the uncertainty about this function.

7.2.5 Scaling up to high-dimensional states

Single-policy and outer loop multi-policy methods can be extended to handle high-dimensional input data in much the same way as the corresponding single-objective algorithms on which they are based. For example, Tesauro et al. [2008] combined SARSA, non-linear utility, and small multilayer perceptrons to learn to control the power consumption and performance of computing clusters.

Deep reinforcement learning methods [Mnih et al., 2015, Lillicrap et al., 2015] have shown to scale beyond finite and discrete spaces to problem domains with high-dimensional, continuous state and action spaces. Using deep networks as non-linear function approximators for handling multi-objective optimization problems has been on the rise the past few years [Mossalam et al., 2016, Li et al., 2020, Abels et al., 2019, Tajmajer, 2018, Nguyen et al., 2020]. Most of these methods extend the single-objective DQN architecture [Mnih et al., 2015] and follow a single-policy or a multi-policy approach.

Mossalam et al. [2016] extended DQN to a multi-objective setting by learning an approximate coverage set of policies (multi-policy). Each policy is represented using a DQN whose output layer has $|\mathcal{A}| \times n$ nodes, where $|\mathcal{A}|$ represents the size of the action space and n represents the number of objectives. For better efficiency, the authors proposed to re-use the network weights of previously learnt policies for preferences that are similar to each other ($\boldsymbol{w'} \sim \boldsymbol{w}$). Abels et al. [2019] analyzed different architectures while extending DQN to a multi-objective setting: a multi-policy approach with multiple DQNs for different user preferences; and a single-policy approach with a single DQN that can generalize across different user preferences. In scenarios where user preferences change in real-time, the single-policy method was most effective. To improve sample efficiency and address bias to recently seen user preferences, they used a diverse experience replay buffer that contained experiences corresponding to different user preferences. Yang et al. [2019] also used a single-policy approach which generalized across different user preferences, however, they performed envelop updates by using a convex envelope of the solution frontier while updating network parameters. Such envelop updates lead to faster convergence when compared to scalarized updates for a given user preference, which are often sample inefficient, resulting in sub-optimal policies.

There has been some recent work on multi-objective deep reinforcement learning for continuous action spaces. Chen et al. [2019] combined a multi-objective extension of PPO [Schulman et al., 2017] with model-agnostic meta-learning (MAML) [Finn et al., 2017]. The proposed method first learns a meta-policy, which can then be fine-tuned in few iterations to find a set of Pareto optimal policies. Compared to learning each policy from scratch, the method was shown to improve performance in terms of training time and optimality of the resulting Pareto front. Xu et al. [2020] also used a multi-objective extension of PPO, combined with an evolutionary algorithm to guide learning in the most

promising direction. In each generation of evolution, data stored from previous iterations of MORL are used to fit a prediction model, which can help find the pairs of policies and scalarization weights that will improve the solution the most. Each selected policy-weight pair is then improved through MORL to produce offspring policies, which are used to create a new generation of policies. The final generation is divided into policy families by clustering, and policy parameters within each family are interpolated to produce a continuous approximation of the Pareto front. Abdelfattah et al. [2019] used a two-stage approach for learning in environments with non-stationary dynamics and continuous actions. In the first stage, a set of generic skills (e.g., *Move Forward*, *Turn Left*, and *Turn Around*) are learned. In the second stage, the learned skills are used in a hierarchical version of DDPG [Lillicrap et al., 2015] to produce a policy coverage set for the MOMDP. An intrinsically motivated RL algorithm is used to select which objective preferences to explore to improve the coverage set, and a policy bootstrapping mechanism is used to quickly adapt to changes in the environment dynamics.

While the above approaches use linear scalarization, Tajmajer [Tajmajer, 2018] proposed a non-linear action-selection mechanism by using n different DQNs corresponding to each objective which are combined using a separate output layer along with the user preferences. Deep reinforcement learning methods for multi-objective partially observable settings have also been proposed [Nian et al., 2020]. These approaches use action and observation histories along with user preferences as input to the neural network. In general, partially observable settings are much more complex when compared to fully observable settings in terms of training time as well as training stability.

7.2.6 Multi-agent algorithms

As explained in Section 4, in single-agent multi-objective problems, the shape of the utility function, in conjunction with the allowed policy space, can be used to derive the optimal solution set that a multi-objective decision-theoretic algorithm should produce. In multi-agent settings, the situation is more complex, as each individual agent can represent one or more distinct users (i.e., each agent can have a different utility function). For this reason, Rădulescu et al. [2020a] proposed a new taxonomy which classifies multi-objective multi-agent decision making (MOMADM) settings on the basis of both reward structures and utility functions, as shown in Figure 3. We note that the case of individual reward–team utility is equivalent to and treated as the individual reward–individual utility case, since the individual return vectors would still lead to different utility values for each agent, despite them having the same utility functions.

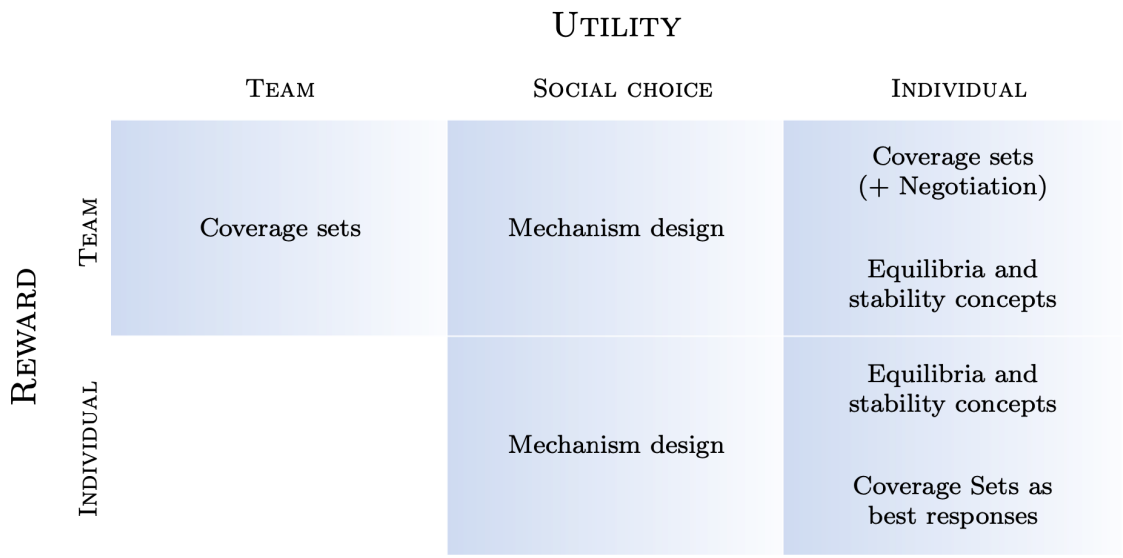


Figure 3: Multi-objective multi-agent decision making taxonomy and mapping of solution concepts [Rădulescu et al., 2020a].

In multi-objective multi-agent settings the agents' strategies are interrelated. For this reason, *solution concepts*, i.e., whether the agents in the system reach outcomes that are of interest, could be used to evaluate the algorithms' performance. We detail below the solution concepts identified by Rădulescu et al. [2020a] for the MOMADM setting and present a few algorithmic approaches that employ them.

Coverage sets. The team reward and team utility setting in MOMADM represents a fully cooperative scenario, where all agents share the same rewards and derived utility. Since there is only one true utility function in the execution phase, coverage sets represent the right solution concept for this case, with the same motivation as for single-agent

multi-objective decision making. Multi-objective coordination graphs (MOCoGs) represent one of the most studied models for cooperative multi-objective multi-agent systems. They exploit the fact that in multi-agent systems the rewards or agents can often be factorised into smaller components. Numerous algorithmic approaches focus on finding (approximate) Pareto coverage sets (3.2) like for example multi-objective bucket elimination (MOBE) [Rollón, 2008, Rollón and Larrosa, 2006], multi-objective Russian doll search [Rollon and Larrosa, 2007], multi-objective (AND/OR) branch-and-bound tree search [Marinescu, 2009, 2011, Rollon and Larrosa, 2008], Pareto local search [Inja et al., 2014], and multi-objective max-sum [Delle Fave et al., 2011]. Another frequently used model is the cooperative multi-objective stochastic game (MOSGs), where reinforcement learning or evolutionary algorithms were used to derive coverage sets (e.g., [Mannion et al., 2018, 2017, Yliniemi and Tumer, 2016]). Similar methods were proposed for the individual reward and utility setting, where a coverage set can also be a set of possible best responses to the behaviours of the other agents (e.g., [Avigad et al., 2011, Eisenstadt et al., 2015, Dusparic and Cahill, 2009]). In an individual reward–team utility setting, coverage sets could be used if all agents agree (e.g., through negotiation [Jonker et al., 2017]) upon which alternative joint policy from the coverage set to execute.

Equilibria and stability concepts. In the individual utility scenario, the utility derived by each agent from the received reward is different, regardless if this reward is the same or not for all the agents. Suitable solution concepts for dealing with decision making between self-interested agents are game theoretic equilibria (e.g., Nash equilibria [Nash, 1951], correlated equilibria [Aumann, 1987]). We find here works that study the idea of robust equilibria in multi-objective games [Qu et al., 2015, Yu and Liu, 2013] or how equilibria are affected by the use of the different optimisation criteria [Rădulescu et al., 2020b]. Furthermore, knowledge transfer [Taylor et al., 2014] and opponent modelling [Zhang et al., 2020] also becomes more important in this context.

When binding agreements among agents are possible, solution concepts from cooperative game theory can also apply to individual utility settings. Coalition formation can therefore become a central problem in these cases, i.e., finding (sub)groups of agents that are willing to make such a binding agreement with each other [Igarashi and Roijers, 2017].

Mechanism design. In game theory, the field of mechanism design takes the system's perspective for multi-agent decision problems. This implies taking as input both the original decision problem (where the agents have individual reward functions that are unknown to the other agents and the "owner" of the game), as well as a social welfare function. The aim is to design a system of additional payments that would (a) force the agents to be truthful about their individual utilities, and (b) lead to solutions that are (approximately) optimal under the social welfare function. In multi-objective settings, the situation is more complex, as the individually received rewards determine the utilities via individual, private utility functions. In general, it can be very challenging?, or even impossible to articulate these functions, so being "truthful" about one's utility might be infeasible from the get-go. Nevertheless, it is possible to design mechanisms for some multi-objective multi-agent problems if the individual utilities can be articulated (e.g., [Grandoni et al., 2010, Pla et al., 2012, Ramos et al., 2020]).

For an in-depth overview of solution concepts for multi-objective multi-agent decision making, the interested reader is referred to a recent survey by Rădulescu et al. [2020a].

8 Evaluating the performance of multi-objective decision making algorithms

Unlike in single-objective RL, there is not only one optimal solution in multi-objective problem settings. MORL algorithms therefore often produce solution *sets* (see Section 3.2). This complicates the evaluation and comparison procedure of MORL algorithms: When is one solution set better than the other? What properties should a solution set have, and how do we measure those?

In this section, we give an overview of existing evaluation metrics, starting with *axiomatic-based* ones (Section 8.1). These assume that the optimal solution is the true Pareto front (or convex hull), and try to compare to this in aspects like spread, coverage, or distance. However, these axiomatic metrics are often difficult to interpret from a user perspective. As argued in Section 4, the development of MORL solutions should be driven by the perspective on user utility. Similarly, *utility-based* evaluation metrics should be used when assessing MORL algorithms (Section 8.2).

After giving an overview on evaluation metrics and approaches, we briefly discuss discuss potential pitfalls when using value function approximations in MORL settings in Section 8.3. Section 8.4 gives an overview of existing benchmarks and their properties.

8.1 Axiomatic-based evaluation metrics

In this section we give an overview of axiomatic approaches to evaluating solutions to multi-objective decision making problems. Such approaches were widely-used in early literature in the field.

8.1.1 The hypervolume metric

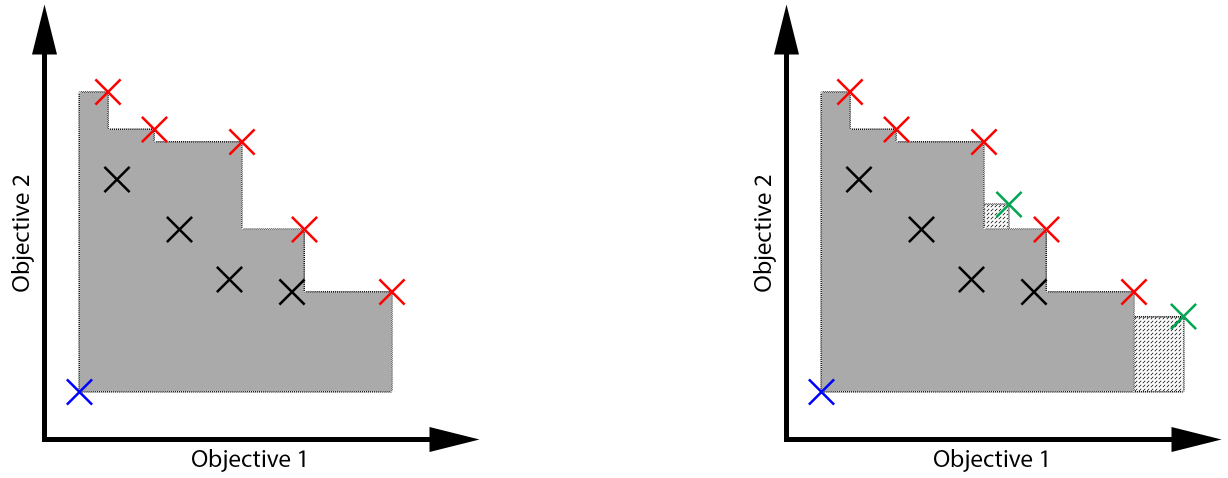


Figure 4: Left: A graphical illustration of the hypervolume for a 2-objective problem, where both objectives are to be maximised. Solutions in red form the undominated set, while solutions in black are said to be dominated. The shaded area denotes the hypervolume of the undominated set with respect to the reference point (shown in blue). Right: The effect of adding two new points (shown in green) to the undominated set.

The hypervolume metric has been widely used to evaluate the performance of multi-objective decision making algorithms (e.g., [Mannion et al., 2018, Vamplew et al., 2011, Van Moffaert et al., 2014, 2013a, Wang and Sebag, 2013, Yliniemi and Tumer, 2016]). The hypervolume metric measures the (hyper-)volume in value-space Pareto-dominated by the set of policies in an approximate coverage set. This correlates with (but is not equal to) the spread of a set of undominated solutions over the possible multi-objective solution space. For this reason, it has been used to compare the sets of solutions produced by multi-policy algorithms (or indeed single policy algorithms run multiple times with different scalarisation/utility function parameters). The accuracy of any set of solutions produced by an algorithm can be evaluated by comparing its hypervolume with that of the non-dominated set produced by a competing algorithm, or with that of the true Pareto front of the application domain (if known). In domains where the true Pareto front is known, the hypervolume represents an absolute maximum level of performance that may be achieved in terms of coverage of the set of solutions over the objective space:

$$\texttt{HyperVolume}(CS, \mathbf{V}_{\text{ref}}) = \bigcup_{\pi \in CS} \texttt{Volume}(\mathbf{V}_{\text{ref}}, \mathbf{V}^{\pi}), \tag{16}$$

where $\texttt{Volume}(\mathbf{V}_{\text{ref}}, \mathbf{V}^{\pi})$ is the volume of the hypercube spanned by the reference vector, $\mathbf{V}_{\text{ref}}$, and the vector in the CS, $\mathbf{V}^{\pi}$.

Figure 4 illustrates the hypervolume of a set of undominated solutions with respect to a given reference point, $\mathbf{V}_{\text{ref}}$, for a 2-objective maximisation problem, where both objectives are to be maximised. For convenience, a reference point in the multi-objective space is often used when calculating the hypervolume of a non-dominated set. This reference point may be chosen arbitrarily.

Although widely used in the literature, the hypervolume metric has a number of problems. The most significant of these is that hypervolume values are difficult to interpret, as they do not map to any real-world notion of value or utility. When comparing the hypervolume of two competing sets of solutions, the benefit of a certain increase or decrease in hypervolume is not readily apparent to the end user. Adding just one non-dominated solution at the extreme ends of the objective ranges could lead to a large increase in the hypervolume of a non-dominated set, even if this additional solution is of little interest to the end user. Conversely, adding a new solution that is close to other solutions in the non-dominated set can result in a minimal increase in hypervolume, even if the new solution is valuable to the end user. Finally, it is unlikely that the true set of non-dominated solutions will be known a priori for any non-trivial multi-objective decision making applications. This invalidates one of the main arguments for the use of the hypervolume metric, i.e., evaluating the coverage of a set of solutions with respect to a reference set. Furthermore, the hypervolume is only applicable to settings where every Pareto-non-dominated policy potentially contributes to the utility. This is not always the case. For example, when the utility function is known to be linear, the hypervolume is not applicable

as many policies that would contribute to the hypervolume are known to not improve utility (i.e., all concave regions in the Pareto front). For these reasons, we recommend the use of alternative metrics that better reflect the usefulness of the solutions produced by an algorithm (such as the user's utility).

8.1.2 Sparsity of coverage sets

The information that is contained by metrics like the hypervolume is rather limited. The only guarantee we have is that if the hypervolume is maximised (unless there are points that contribute 0 hypervolume at the edges that we have missed), then a Pareto Coverage Set has been recovered. This is of course not informative, especially during learning.

One key bit of critique is that if we have two approximate solution sets with equal – or approximate – hypervolume, then we should prefer the set which has more spread over the value space. In other words, the set that contains value vectors that are furthest apart from each other is the better one. From a utility-based perspective, this is also intuitive, as the user will pick the best vector from a solution set $\mathcal{S}$ according to:

$$\pi^* = \arg\max_{\pi \in \mathcal{S}} u(\mathbf{V}^\pi), \tag{17}$$

so it helps if the user has a larger variety of value vectors to select from.[2]

In multi-objective optimisation, this idea has been used to create algorithms that explicitly look for diverse solutions [Deb et al., 2002]. In MORL, this same idea has been used to diversify the experience replay buffer, in order to be able to adapt to different utility functions faster [Abels et al., 2019].

In addition to finding solution sets that are evenly spread over the value space (i.e., sets with high diversity), it is desirable that the solutions provide a dense coverage of the whole Pareto front (i.e., sets with high resolution). For this purpose, Xu et al. [2020] proposed to use sparsity as a metric for evaluation of Pareto front approximations. The proposed sparsity metric is defined as:

$$Sp(\mathcal{S}) = \frac{1}{|\mathcal{S}| - 1} \sum_{j=1}^{m} \sum_{i=1}^{|\mathcal{S}|-1} (\tilde{\mathcal{S}}_j(i) - \tilde{\mathcal{S}}_j(i+1))^2, \tag{18}$$

Here $\mathcal{S}$ is the Pareto front approximation for an environment with m objectives, and $\tilde{\mathcal{S}}_j(i)$ is the i-th value in the sorted list for the j-th objective values in $\mathcal{S}$. According to this metric, Pareto front approximations with lower sparsity are better.

8.1.3 The $\varepsilon-$metric

The ε metric [Zitzler et al., 2008] measures how closely a solution set $\mathcal{S}$ approximates the Pareto front PF. It has been widely used in multi-objective evolutionary optimisation [Zitzler et al., 2008] and reinforcement learning [Vamplew et al., 2017a]. There are two measures, the additive and the multiplicative ε-indicator.

The **additive** ε-indicator is given by

$$I_{\varepsilon^+} = \inf_{\varepsilon \in \mathbb{R}} \{\forall \mathbf{V}^\pi \in PF,\ \exists \mathbf{V}^{\pi'} \in \mathcal{S} : V_i^\pi \leq V_i^{\pi'} + \varepsilon, \forall i \in \{1, \ldots, n\}\}, \tag{19}$$

where n is the number of objectives. where $\mathbf{V}^\pi \in \mathbb{R}^d$ is the d-dimensional value of policy π (see Section 3). In words, a solution set $\mathcal{S}$ is an ε-approximate Pareto front according to this metric if *for each* value vector $\mathbf{V}^\pi$ on the Pareto front PF, there *exists at least one* value vector $\mathbf{V}^{\pi'}$ in the solution set $\mathcal{S}$, such that for each objective d, the value in $\mathbf{V}^{\pi'}$ is *at most* ε smaller than the values in $\mathbf{V}^\pi$.

The (less commonly used) **multiplicative** ε-indicator is given by

$$I_{\varepsilon^*} = \inf_{\varepsilon \in \mathbb{R}} \{\forall \mathbf{V}^\pi \in PF,\ \exists \mathbf{V}^{\pi'} \in \mathcal{S} : V_i^\pi \leq V_i^{\pi'}(1 + \varepsilon), \forall i \in \{1, \ldots, n\}\}, \tag{20}$$

The difference to the additive indicator is in how the distance is calculated: here, each objective can at most be worse by a multiplicative factor of $1 + \varepsilon$, i.e., this scales with the magnitudes of the individual values (objectives with larger values allow a larger deviation).

The ε metric gives an indication of the factor by which an approximate solution set is worse than the Pareto Front, considering all objectives. It can also be used to prepare two arbitrary solution sets instead of a solution set and the

[2] Please note that this intuition implicitly assumes some form of continuity in the user's utility function, u.

Pareto front. Unlike the hypervolume metric, it can give an indication of *whether* one is better than the other (they might, however, be incomparable w.r.t. this metric).

We argue that the ε metric is more useful than the hypervolume, since it can directly be used to derive a utility for a given user, see Section 8.2.

8.1.4 Metrics from information retrieval

The Coverage Ratio metric is used in [Yang et al., 2019] as an evaluation metric for comparing different multi-objective algorithms. It is a measure of the count of policies recovered from a finite Coverage Set (CS) which is determined by a comparison between the set $\mathcal{S}$ of policies ϕ with value vectors $\mathbf{V}^{\pi} \in \mathbb{R}^d$ found by a MORL algorithm, and the value vectors corresponding to the policies in the (ground-truth) CS. The measure weights both the precision and recall of finding policies in the CS, the following definition is used when calculating precision and recall such that policies with value vectors within epsilon of the value vector of a policy in CS, are classed as in the CS.

$$\mathcal{S} \cap_{\epsilon} CS = \{\mathbf{V}^{\pi} \in \mathcal{S} \mid \exists \mathbf{V}^{\pi^*} \in CS : \|\mathbf{V}^{\pi} - \mathbf{V}^{\pi^*}\|_1 / \|\mathbf{V}^{\pi^*}\|_1 < \epsilon\} \tag{21}$$

The Coverage Ratio (also known as the F-score) is then calculated as the harmonic mean between the precision $= |\mathcal{S} \cap_{\epsilon} CS|/|\mathcal{S}|$ and recall $= |\mathcal{S} \cap_{\epsilon} CS|/|CS|$ measures.

$$CR(\mathcal{S}) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{22}$$

We argue that there are several issues with the Coverage Ratio metric. Firstly, the measure (Equation 22) implies that precision and recall are equally important which in reality is not the case. For example, if the utility function is linear, and $\mathcal{S}$ contains excess policies that have a value which is the weighted sum of two other value vectors. Such vectors need not be in the CCS (convex coverage set, see Equation 10) and decreases the precision. However, having this excess policy does not decrease the utility for any linear utility function. Conversely, as missing out a policy in the CS typically does decrease the utility for a whole range of utility functions.

Secondly, like the hypervolume metric, the Coverage Ratio does not account for the different levels of utility a user gains from different solutions. In this measure the presence of any solution from the CS is treated as of equal value to any other solution found also in the CS. However, unlike the hypervolume metric the Coverage Ratio is not correlated at all with the spread of the set of non-dominated solutions over the possible multi-objective space. Therefore, this measure fails to account for any utility the user gains from the spread of solutions but retains the undesirable properties of the hypervolume metric.

Thirdly, the ε-parameter that controls the threshold for when we consider a policy in the Convex Coverage Set is a parameter that needs to be chosen and can have a large impact on the Coverage Ratio. Specifically, if a value estimate $\tilde{\mathbf{V}}$ is within the hypercube surrounding a value in the CS, $\mathbf{V}_i \pm \varepsilon$, it is assumed to correspond to that value vectors in the CS. Setting ε arbitrarily high can lead to all solutions being treated in the CS, while setting it low could lead to an algorithm producing no solutions in the CS. When comparing different algorithms, the choice of ε could have a large impact on the final ranking and it is not a priori clear what a fair setting of ε would be.

Lastly introducing the notion of ε to the recall measure means that multiple counting of policies in the CS can occur as more than one value vector in F could be within ε distance of the same policy's value vector in the CS. Not only can an algorithm thus "recall" more than the ground truth number of policies, but more importantly, "recalling" two policies for the same ground truth policy can obfuscate the missing of another policy in the resulting value of the metric, which is of course highly undesirable.

8.2 Utility-based evaluation metrics

As argued in Section 4, the utility-based approach is preferable in most scenarios, since here the algorithms are designed and evaluated with respect to the utility that the solution can offer to the user. Accordingly, any evaluation metric should take this into account. Many of the axiomatic-based evaluation metrics are difficult to interpret in terms of user utility, and in addition they often require access to the true Pareto front. If it is possible to assess the utility of the user at time of deployment, then solution sets can be compared based on user utility.

For instance, the user's utility might correspond to revenue that a deployed solution achieves; in this case, the utility can be measured and compared directly.

For when this is not possible, Zintgraf et al. [2015] propose two utility-based evaluation methods, the *expected utility metric* (EUM) and *maximal utility loss* (MUL). Compared to many other metrics such as the hypervolume metric,

these are more suitable to compare different algorithms, since they are aimed at directly evaluating an agent's ability to maximize user utility, which is always our ultimate goal.

For a given solution set, the EUM is defined as the expected utility for a user from this solution set, under some prior distribution over user utility functions. Under the SER optimality criterion, this can be written as:

$$\text{EUM} = \mathbb{E}_{P_u}\left[\max_{\pi \in \mathcal{S}} u(\mathbf{V}^{\pi})\right], \tag{23}$$

where $\mathcal{S}$ the solution set outputted by an algorithm, u the utility function of the user, $\mathbf{V}^{\pi}$ the vector-value of the best policy from that set (according to u). The expectation is taken with respect to the distribution over utility functions P_u. This metric is useful in situations where we care about the agent's ability to do well across many different utility functions, e.g., because many policies from the solution set will be used over time, or because they will be used for different users. This metric does however require a good prior over possible scalarisation functions in order to meaningfully evaluate a given solution set.

The MUL measures the maximal loss in utility that occurs when taking a policy from a given solution set, instead of the full set of possibly optimal solutions. Under the SER optimality criterion, this can be written as:

$$\text{MUL} = \max_{u \in \mathcal{U}} \left(\max_{\pi^* \in \mathcal{S}^*} u(\mathbf{V}^{\pi^*}) - \max_{\pi \in \mathcal{S}} u(\mathbf{V}^{\pi}) \right), \tag{24}$$

where $\mathcal{S}^*$ is the true optimal solution set (PF or CH, or a very good approximation thereof), $\mathcal{S}$ the solution set outputted by an algorithm, $\mathbf{V}^{\pi}$ the vector-value of said policy, and u the utility function of the user, over which we take the maximisation with respect to the space of possible utility functions $\mathcal{U}$. Since it is often infeasible to compute the full set of optimal solutions in order to compute this metric and compare algorithms, a good reference set can be used (such as the union of multiple solution sets, e.g., the final solution sets of all algorithms evaluated in a comparison).

We note that MUL is bounded if an ε-bound can be given on the accuracy of the set $\mathcal{S}$ produced by the algorithms, and the utility function is guaranteed to be (Lipschitz)-continuous [Zintgraf et al., 2015].

8.3 A word of caution regarding value vector approximation

In multi-objective planning, it is often the case that for a given policy, π, we know the exact value vector, $\mathbf{V}^{\pi}$. When the human decision maker (or even another algorithm) selects such a policiy in the selection phase (see Section 5.1), we can thus trust these value vectors. This is key as we derive utility from these value vectors by applying the utility function to them (either implicitly or explicitly).

In multi-objective reinforcement learning it is tempting to think we have proper value vectors too, as many algorithms produce value vector estimates. In the literature these are often also denoted $\mathbf{V}$ or $\mathbf{Q}$. However, it is essential to note that these are stochastic estimates, that may well have both high variance, or even systematic biases [Hasselt, 2010]. This issue is exacerbated by the use of function approximators, such as neural networks, which may have their own added variance and/or biases. It would therefore be fairer to explicitly denote such value estimates as estimates by using tildes, $\tilde{\mathbf{V}}$ or $\tilde{\mathbf{Q}}$, for example, but this is not common practice.

In multi-objective RL, having inexact value estimates in the coverage set presented to human decision makers (or other algorithms), can lead to missing on two sides: firstly, if the value estimate of the actual best policy is off, that policy may not be selected. Furthermore, the value estimate of the policy that is selected may also be off, leading to a different utility than expected. Combined, these two sources of potential loss can severely affect the user utility.

In order to mitigate the issue of inexact value estimates, and maybe even more importantly, biases in value estimates, we recommend that the coverage sets presented in selection phases do not directly rely on the value vector estimates from the MORL algorithms. Instead, we recommend to extract the policies that constitute the coverage set, and run a separate and thorough policy evaluation, before selecting any policy to execute.

8.4 Benchmark problems for multi-objective decision making

Well established benchmark problems are important for evaluation of reinforcement learning algorithms, since even small variations in the experiment design may have a significant impact on the results. By using common benchmarks a fair comparison of different approaches can be ensured, and by evaluating algorithms on several benchmarks the generality of results can be studied. Table 2 presents an overview of frequently used MORL benchmarks with discrete states and actions, as well as more recent extensions and additions with high-dimensional states, partial observability, multiple agents, and continuous actions.

Table 2: Benchmarks for multi-objective reinforcement learning.

Benchmarks					
Name	Number of Objectives	Observation Space	Action Space	Pareto Front	Ref.
Deep Sea Treasure	2	Discrete	Discrete	Known	[Vamplew et al., 2011]
Deep Sea Treasure [a]	2	Continuous	Discrete	Known	[Mossalam et al., 2016]
Deep Sea Treasure [b]	2	Continuous	Discrete	Known	[Nian et al., 2020]
Deep Sea Treasure [c]	2-3	Continuous	Discrete	Known	[Hasan et al., 2019]
MO-Puddleworld	2	Discrete	Discrete	Known	[Vamplew et al., 2011]
MO-Mountain-Car	3	Discrete	Discrete	Known	[Vamplew et al., 2011]
Resource Gathering	3	Discrete	Discrete	Known	[Vamplew et al., 2011]
Linked Rings	2	Discrete	Discrete	Known	[Vamplew et al., 2017b]
Non Recurrent Rings	2	Discrete	Discrete	Known	[Vamplew et al., 2017b]
Space Exploration	2	Discrete	Discrete	Known	[Vamplew et al., 2017a]
Bonus World	3	Discrete	Discrete	Known	[Vamplew et al., 2017a]
MO Beach Problem [*]	2	Discrete	Discrete	Known	[Mannion et al., 2018]
Mine Cart	$\geq$2	Continuous	Discrete	Known [d]	[Abels et al., 2019]
HalfCheetah-v2	2	Continuous	Continuous	Unknown	[Xu et al., 2020]
Hopper-v2	2	Continuous	Continuous	Unknown	[Xu et al., 2020]
Swimmer-v2	2	Continuous	Continuous	Unknown	[Xu et al., 2020]
Ant-v2	2	Continuous	Continuous	Unknown	[Xu et al., 2020]
Walker2d-v2	2	Continuous	Continuous	Unknown	[Xu et al., 2020]
Humanoid-v2	2	Continuous	Continuous	Unknown	[Xu et al., 2020]
Hopper-v3	3	Continuous	Continuous	Unknown	[Xu et al., 2020]

[a] With image observations.
[b] With image observations and partial observability.
[c] With image observations and dynamic environment.
[d] Optimal solutions are known for the default configuration of the environment.
[*] Multi-agent environment.

For multi-objective decision problems other than MOMDPs, such as multi-objective coordination graphs [Marinescu, 2009, Roijers and Whiteson, 2017, Rollon and Larrosa, 2008] and multi-objective normal form games [Rădulescu et al., 2020b, Zhang et al., 2020], benchmarks are also few and spread out over different papers.

9 An illustrated example

In Section 2, we illustrated diverse problems that require multi-objective optimisation. One of them is the water management problem (Section 2.2). In that case, as we want to propose a diverse set of solutions to the decision maker—following the taxonomy defined in Section 5—we are in a multi-policy scenario. In this section, we concretely tackle the water management problem by applying a multi-objective algorithm, and comparing it with its single-objective counterpart, with both producing deterministic policies. We make no assumptions about the utility function of the decision maker, except that it is monotonically increasing. Thus, the solution set we aim to produce is the Pareto coverage set of deterministic non-stationary policies.

9.1 Setting

In this problem, the goal is to control a dam responsible for dispatching water downstream while avoiding flooding in the region. This environment is modelled as a one-dimensional, continuous state, representing the amount of water present in the reservoir. This is subject to change, depending on factors such as rain. At each timestep, the dam can release a specified water amount (a one-dimensional, continuous action) [3].

The dam is responsible for supplying water, and needs to meet the water demand. At the same time, it should be careful not to hold too much water, as this increases the risk of flooding upstream. These objectives are conflicting,

[3]The code used to illustrate this setting can be found at the following link: `https://gitlab.ai.vub.ac.be/mreymond/morl-guide`

since the inflow of water is on average insufficient to cope with the water demand. In order to increase the chance of meeting future demand, the reservoir needs to be filled, thus increasing the risk of flooding upstream.

9.2 Multi-objective natural evolution strategies

With an unknown utility function, which may be non-linear, we want to propose a set of alternatives to the decision makers by approximating the Pareto-front. Since we have no prior information about the preferences of the decision makers, we make no assumptions about the utility function.

The algorithm that is used to approximate the coverage set is Multi-Objective Natural Evolution Strategies (MONES) [Parisi et al., 2017]. In essence, a parametric policy is used, where each parameter is represented by a Gaussian distribution. Sampling these distributions results in an executable policy that can be applied on our environment. MONES optimises the mean and standard deviation of each parameter such that, whenever we sample from them, we obtain a policy that leads to a different point on the Pareto front. MONES is an iterative process, that repeats three steps:

1. Sample a population of policies from our parameters, and execute them on the environment;
2. Evaluate the quality of these policies using an indicator metric;
3. Perform a gradient step that improves this indicator using the natural gradient.

Our policy is represented as a small feedforward neural network, where each weight is sampled from its own Gaussian distribution. The network contains a single hidden layer of 50 neurons and, although these weights are correlated with each other, we assume each Gaussian distribution to be independent for the sake of simplicity.

To evaluate the performance of the policies, MONES requires an indicator metric. We closely follow [Parisi et al., 2017], where the metric used is a combination of non-dominance ranking and crowding distance. For non-dominance ranking, a rank of 0 is applied for all the non-dominated points of the discovered returns. By removing these solutions from the population, a new set of policies becomes non-dominated. We set the rank for these points to -1. This process is repeated until no points remain to be evaluated. At each iteration the rank is decreased by 1.

This rank is then combined with the crowding distance, which is a metric providing information about the diversity of a frontier:

- For all the points of the same rank we compute, for each dimension, the distance between its closest neighbours.
- This distance is normalised. Points close to each other will have a crowding distance close to 0, while points at the border of the frontier will have a distance close to 1.

Summing these 2 metrics together provides us with an indicator that encourages points to be on the Pareto front, and be as diverse as possible.

The MONES learner is trained for 30 iterations, sampling 50 policies every time. Each policy is executed 10 times. The average return of each policy is evaluated using the non-dominance/crowding-distance metric.

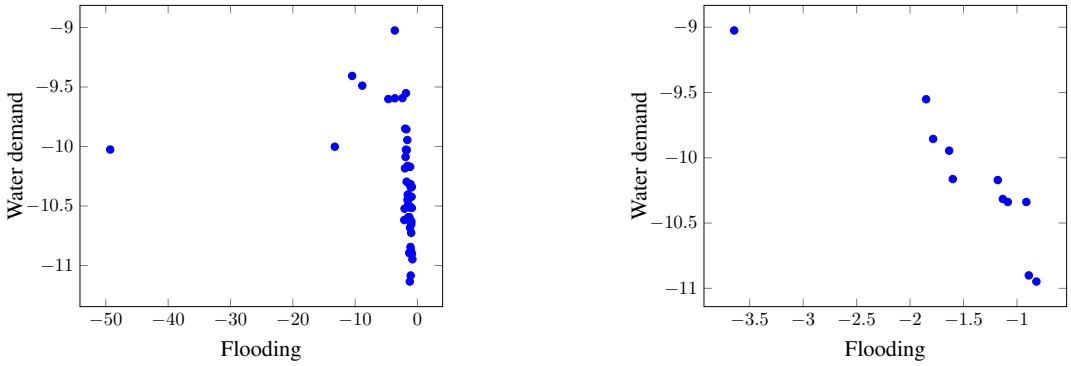
As can be seen in Figure 5, after 30 iterations of training, the policies sampled from the Gaussian distributions achieve diverse combinations of returns. The right part of the figure shows 11 non-dominated solutions, but the vast majority of the policies (48/50) reach returns reasonably close to the frontier, resulting in a set of diverse, high-quality solutions.

9.3 Using single-objective subroutines

Instead of using a dedicated method (MONES) to discover diverse policies, we use an outer loop method. In this particular case, we use Natural Evolution Strategies (NES) as a single objective subroutine. This subroutine is called a number of times, each time with a different utility function, hopefully resulting in different policies that reach different points of the coverage set.

This requires us to know the distribution over user utility functions. We consider a uniform distribution over linear scalarisation functions, i.e. each utility function is a weighted sum, where the weights are uniformly sampled from a 1-simplex (since our problem has two objectives).

Since MONES takes inspiration from NES, both algorithms are quite similar, the main difference being the indicator metric used. While MONES optimises on the combination of ranking and crowding distance, NES optimises on the utility of the return. All other parameters are kept the same as for MONES.



(a) All returns at the last iteration.

(b) Non-dominated returns at the last iteration.

Figure 5: Comparison of *returns* (left) with *non-dominated returns* (right). In order to enhance presentation, the right plot's horizontal axis was clipped to a smaller interval.

We sampled 30 utility functions, resulting in 30 different NES runs. In Figure 6, we compare the coverage sets found by MONES and NES. In order to have the same number of points for each method, we sampled 30 new policies from our trained MONES and plotted the resulting returns.

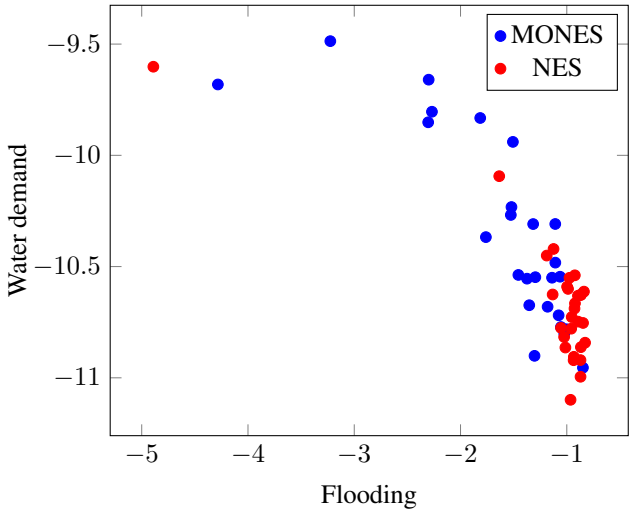


Figure 6: Comparison of MONES and NES policies.

Although the utility functions used by NES were sampled across the whole simplex, it does not result in a spread-out coverage set. The vast majority of returns hover around $(-1.0, -10.7)$, where -1.0 refers to flooding and -10.7 to water demand. In comparison, the policies sampled from MONES result in more spread-out returns. This is because the crowding distance is taken into account in the indicator metric used by MONES, encouraging the returns to be diverse. It is also important to mention that, even though hyperparameters are the same for both methods, NES had to be trained 30 separate times, while MONES only once. Compared to NES, MONES is sample-efficient and results in a more evenly spread-out coverage set. Finally, NES requires us to make assumptions about the distribution over user utility functions, while MONES makes no such assumption.

9.4 Comparing evaluation metrics

In order to evaluate the training progression of MONES after each iteration, we use two of the metrics described in Section 8. First, we use the hypervolume (Equation 16). This metric requires a reference point, which in this case is set to the worst return found across the whole training process. Secondly, we use the Expected Utility Metric (EUM, Equation 23). This metric requires a good prior distribution over user utility functions, as well as a good approximation of the solution set. As a prior distribution, we choose the one used for NES, e.g. a uniform distribution over linear scalarisation functions.

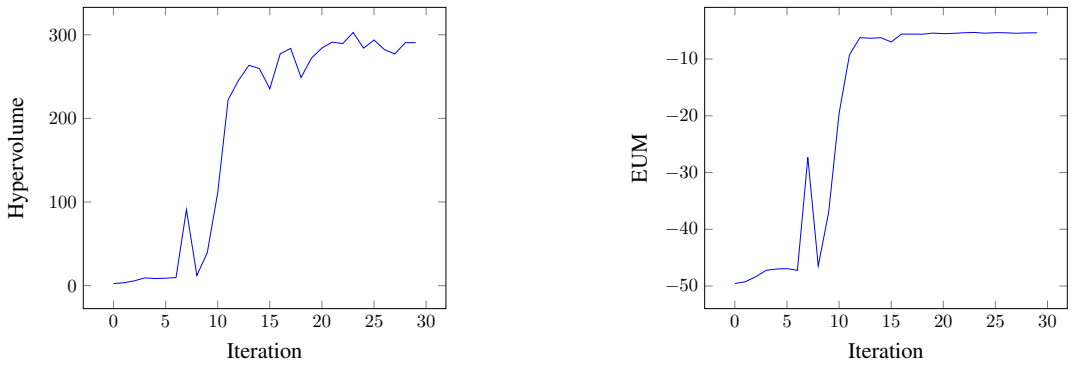


Figure 7: Evaluation of MONES using 2 multi-objective metrics: the Hypervolume (left) and Expected Utility Metric (right).

As we can see in Figure 7, MONES converges towards our approximation of the optimal solution after 15 iterations, and stays stable for the remainder of the training process. We observe similar trends with the other evaluation metric used (the hypervolume). We also note at iteration 7, that the sudden improvement in EUM is reflected in the hypervolume, although not as drastically. The solution set found in iteration 7 almost doubled the expected user utility, but only resulted in a 30% increase in hypervolume. This shows that, although an increase of hypervolume is correlated with an improvement of the coverage set, it does not reflect the utility of the user.

In conclusion, we tackled the water management problem with a dedicated multi-objective algorithm. By changing the indicator metric of NES to cope with multiple criteria, we discovered a solution set that is more diverse than repeatedly applying its single-objective counterpart, and for which the solutions are more evenly spread out. Moreover, MONES only requires minimal assumptions on the utility function (monotonicity). Finally, the number of required interactions with the environment is vastly improved since, in this case, 30 instances of NES needed to be executed, compared to just a single instance of MONES.

10 Conclusion, challenges and open questions

Recent years have seen significant breakthroughs in the capabilities of sequential decision making agents based on planning or reinforcement learning methods. This has lead to the increasing applications of these agents to complex, real-world problems. However, as illustrated by the motivating examples in Section 2, these real-world tasks frequently require trade-offs to be made between multiple conflicting objectives. This contrasts with the inherently single-objective nature of the environments such as board and video games on which the planning and learning algorithms have largely been developed and evaluated. When these single-objective methods are applied to problems which are multi-objective in nature, either some objectives wind up being excluded from consideration, or the objectives are summed together to form a scalar reward. As discussed in Sections 1 through 4, the use of single-objective methods to address multi-objective problems has numerous disadvantages: it forces a priori and uncertain decisions to be made about the desired trade-offs between objectives, it limits the ability to discover multiple policies so as to rapidly adapt to changing preferences, it shifts the responsibility for managing trade-offs from the problem stakeholders to the system developer, and it may result in solutions which fail to maximise user utility.

While the last decade or so has seen significant achievements in the development of planning and RL algorithms for multi-objective problems (as reflected in Section 7), it remains a niche area compared to the amount of research on single-objective agents. In addition a number of challenges arise in the context of multiple objectives which do not exist in the single-objective domain. As such there remain a number of areas where additional research and algorithmic development is required. The remainder of this article will present an overview of the topics which we believe to be the most significant and pressing challenges for multi-objective agent research.

10.1 Lack of multi-objective datasets and benchmarks

Data plays a role in multi-objective decision making (MODM). When solving a given MO problem, data is usually needed to characterise and solve the involved objectives. However, the currently available data may not be sufficient to model some objectives or domains. Whereas this tends not to be an issue for company-oriented research (since companies can usually obtain the required data should it be necessary to achieve its goals), it is often a significant

problem for basic research (where the lack of data may make it impossible to study some problems). Some of the challenges faced here include: heterogeneity, availability, and lack of correlation.

As an example, consider the case of traffic authorities aiming to optimise the control of traffic lights as to minimise competitive objectives like travel time, polluting emissions, and discomfort level. To accommodate all these objectives, one needs to deal with all the aforementioned challenges. Data heterogeneity comes into play because the data comes from different sources: data about specific trips come from drivers and passengers; overall traffic statistics come from traffic authorities; fleet demographics come from censoring authorities; CO_2 emission profiles come from manufacturers, based on existing fleet; etc. Availability refers to the fact that the above information is not openly available, either because of privacy concerns, or due to the lack of data release policies. Finally, the data may not be correlated, as is the case of traffic statistics and fleet demographics, which come from different, possibly incompatible sources.

A challenge related to the availability of data is the availability of good benchmark problems for evaluation of MORL algorithms. So far, a limited number of benchmark problems have been proposed for MORL research, and many of those proposed are quite simple (see Section 8.4). Some advantages of the existing benchmark problems are, e.g., that they are simple to understand, experiments can be run in a short time, and optimal solutions to the problems are often known. However, they lack the complexity of many real-world problems that deal with multiple conflicting objectives. Referring to our motivating examples for multi-objective reinforcement learning and planning (see Section 2), we note that more benchmarks with complex state and action spaces, partial observability, many objectives, multiple agents, and decision making over long time horizons are needed. One approach to quickly increase the number of available MORL benchmarks could be to modify existing single-objective benchmarks, by making their reward functions multi-objective.

In conclusion, these challenges frequently slows down or even hinders research progress on MODM. Building upon this background, it is of utmost importance for companies and researchers to make their data and benchmarks available. Actions towards this direction include: making your MO problems, data, benchmarks, and baseline implementations available on open platforms; supporting other researchers interested in your problem; negotiating data retention procedures with companies; among others. Without the support of the involved parties, the potential benefits that MODM could bring to our society may not be realised.

10.2 Many-objective problems

Within the field of multi-objective evolutionary optimisation, the task of handling problems with many objectives (usually defined as four or more objectives) has emerged as a distinct sub-field, in recognition that algorithms which work well for a small number of objectives may scale poorly to many objectives [Von Lücken et al., 2014, Li et al., 2015]. So far there has been only minimal work in planning or RL for many-objective problems. For instance, Zintgraf et al. [2018] consider a traffic regulation problem with 11 objectives (reflecting the delay duration and queue length for different traffic participants and different directions), and how to elicit and model user utility in such a setting using pairwise comparison queries between solutions and Gaussian processes. Giuliani et al. [2014] demonstrate a dimensionality-reduction approach in which the original objectives are mapped to a lower dimension using Non-negative Principal Component Analysis, while Yahyaa et al. [2014] examined the performance of bandit algorithms on problems with up to five objectives. However, the development of a broader suite of algorithms for many-objective problems remains an important direction for future work.

10.3 Multi-agent problems

Numerous real-world problems involve both multiple actors and objectives that should be considered when making a decision. Multi-objective multi-agent systems represent an ideal setting to study such problems. However, despite its high relevance, it remains an understudied domain, perhaps due to the increasingly complex dimensions involved.

Prior to the recent survey on multi-objective multi-agent decision making [Rădulescu et al., 2020a], the literature in this area was rather fragmented and lacked a uniformly accepted framework or set of assumptions to allow for a proper comparison or placement of contributions and to identify gaps in terms of studied settings. Following the taxonomy set out by [Rădulescu et al., 2020a], and the links that have been made to suitable solution concepts for MOMADM (briefly discussed in Section 7.2.6), we anticipate that the pace of research on multi-objective multi-agent problems will increase in the coming years.

There are countless open challenges brought forward by the MOMADM domain [Rădulescu et al., 2020a], ranging from how to develop negotiation strategies for selecting between multiple potential solutions, how equilibria are affected by the choice of the optimisation criteria (SER vs. ESR, Section 5.3) and utility functions of the agents,

how to learn about the behaviour or objective preferences of other agents, how to deal with sequential or continuous state-action settings.

Finally, all the observations and remarks regarding the scarce availability of datasets and benchmarks we make in Section 10.1 are even more pressing in the case of multi-objective multi-agent settings, rendering the evaluation of MOMADM approaches a challenging task.

10.4 Dynamically identifying and adding objectives

As discussed earlier in Section 5.1 under the "review and adjust" scenario, an analysis of the policy found based on an initial formulation of a problem may reveal the need to modify or extend the objectives considered by the agent in order to find a more acceptable solution. While prior work in the single-objective literature has considered modifying aspects of the problem either during or after learning or planning, such as changes in environmental state dynamics [Nagabandi et al., 2019], dynamic rewards [Devlin and Kudenko, 2012], or introducing new actions [Mandel et al., 2017], obviously the addition of new objectives is unique to multi-objective methods.

Ideally the agent should be able to integrate additional or modified objectives without needing to discard prior learning, and with minimal regret experienced while adjusting its policy. One means by which this might be achieved is to maintain an archive of the agent's experience under its current policy, which can be used to perform offline learning related to updated specifications of objectives without any further interaction with the actual environment. Alternatively, during learning the agent may be able to identify for itself states that could be associated with potential new objectives (for example, states which are highly different in feature space from other states), and create its own rewards associated with these states such that its policy can be rapidly updated should the user define a new objective associated with these states [Karimpanal and Wilhelm, 2017].

10.5 Closing remarks

The aim of this article is to encourage a wider adoption of multi-objective agent technologies in the development of real-world applications. To this end, we have identified a range of factors which need to be considered when designing a multi-objective solution, as well as reviewing how current multi-objective planning and RL algorithms relate to these factors. In addition, we have provided examples demonstrating how existing methods can be applied to discover suitable agent policies for some simple multi-objective tasks. Our hope is that this article will serve to inspire the future growth of applications based on multi-objective agents.

References

S. Abdelfattah, K. Merrick, and J. Hu. Intrinsically motivated hierarchical policy learning in multi-objective markov decision processes. *IEEE Transactions on Cognitive and Developmental Systems*, 2019.

M. Abdullah, A. Yatim, C. Tan, and R. Saidur. A review of maximum power point tracking algorithms for wind energy systems. *Renewable and Sustainable Energy Reviews*, 16(5):3220–3227, 2012.

A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In *International Conference on Machine Learning*, pages 11–20. PMLR, 2019.

N. Aissani, B. Beldjilali, and D. Trentesaux. Efficient and effective reactive scheduling of manufacturing system using sarsa-multi-objective agents. In *MOSIM'08: 7th Conference Internationale de Modelisation et Simulation*, pages 698–707, 2008.

L. M. Antonio and C. A. C. Coello. Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 22(6):851–865, 2017.

K. Aoki, H. Kimura, and S. Kobayashi. Distributed reinforcement learning using bi-directional decision making for multi-criteria control of multi-stage flow systems. In *The 8th Conference on Intelligent Autonomous Systems*, pages 281–290, 2004.

R. J. Aumann. Correlated equilibrium as an expression of bayesian rationality. *Econometrica: Journal of the Econometric Society*, pages 1–18, 1987.

G. Avigad, E. Eisenstadt, and M. W. Cohen. Optimal strategies for multi objective games and their search by evolutionary multi objective optimization. In *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pages 166–173. IEEE, 2011.

A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, pages 41–47, 2008.

G. Beliakov, S. Bowsell, T. Cao, R. Dazeley, V. Mak-Hau, M.-T. Nguyen, T. Wilkin, and J. Yearwood. Aggregation of dependent criteria in multicriteria decision making problems by means of capacities. In *23rd International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, 2019. doi: https://doi.org/10.36334/modsim.2019.B3.beliakov.

D. Borsa, A. Barreto, J. Quan, D. J. Mankowitz, H. van Hasselt, R. Munos, D. Silver, and T. Schaul. Universal successor features approximators. In *International Conference on Learning Representations*, 2019.

D. Bouneffouf, I. Rish, and C. Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

D. Bryce, W. Cushing, and S. Kambhampati. Probabilistic planning is multi-objective. *Arizona State University, Tech. Rep. ASU-CSE-07-006*, 2007.

T. Brys, K. Van Moffaert, K. Van Vaerenbergh, and A. Nowé. On the behaviour of scalarization methods for the engagement of a wet clutch. In *2013 12th International Conference on Machine Learning and Applications*, volume 1, pages 258–263. IEEE, 2013.

A. Castelletti, F. Pianosi, and R. Soncini-Sessa. Water reservoir control under economic, social and environmental constraints. *Automatica*, 44(6):1595–1607, 2008.

A. Castelletti, F. Pianosi, and M. Restelli. Tree-based fitted q-iteration for multi-objective markov decision problems. In *IJCNN*, pages 1–8. IEEE, 2012.

A. Castelletti, F. Pianosi, and M. Restelli. A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research*, 49(6):3476–3486, 2013.

D. Chen, Y. Wang, and W. Gao. Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. *Applied Intelligence*, 2020.

W. Chen and L. Liu. Pareto monte carlo tree search for multi-objective informative planning. In *Robotics: Science and Systems*, 2019.

X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt. Meta-learning for multi-objective reinforcement learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 977–983. IEEE, 2019.

H.-T. Cheng. *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia, 1988.

J. E. Cohen. Cooperation and self-interest: Pareto-inefficiency of nash equilibria in finite random games. *Proceedings of the National Academy of Sciences*, 95(17):9724–9731, 1998. ISSN 0027-8424. doi: 10.1073/pnas.95.17.9724. URL `https://www.pnas.org/content/95/17/9724`.

F. Cruz, R. Dazeley, and P. Vamplew. Memory-based explainable reinforcement learning. In *Australasian Joint Conference on Artificial Intelligence*, pages 66–77. Springer, 2019.

A. da Silva Veith, F. R. de Souza, M. D. de Assunção, L. Lefèvre, and J. C. S. dos Anjos. Multi-objective reinforcement learning for reconfiguring data stream analytics on edge computing. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–10, 2019.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

F. Delle Fave, R. Stranders, A. Rogers, and N. Jennings. Bounded decentralised coordination over multiple objectives. In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 371–378, 2011.

Z. Deng and M. Liu. An integrated generation-compensation optimization strategy for enhanced short-term voltage security of large-scale power systems using multi-objective reinforcement learning method. In *2018 International Conference on Power System Technology (POWERCON)*, pages 4099–4106. IEEE, 2018.

Z. Deng, Z. Lu, Z. Guo, W. Yao, W. Zhao, B. Zhou, and C. Hong. Coordinated optimization of generation and compensation to enhance short-term voltage security of power systems using accelerated multi-objective reinforcement learning. *IEEE Access*, 8:34770–34782, 2020.

S. M. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440. IFAAMAS, 2012.

J. Dornheim and N. Link. Multiobjective reinforcement learning for reconfigurable adaptive optimal control of manufacturing processes. In *2018 International Symposium on Electronics and Telecommunications (ISETC)*, pages 1–5. IEEE, 2018.

M. M. Drugan and A. Nowe. Designing multi-objective multi-armed bandits algorithms: A study. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

R. Duan, R. Prodan, and X. Li. Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. *IEEE Transactions on Cloud Computing*, 2(1):29–42, 2014.

P. Dubey and J. Rogawski. Inefficiency of smooth market mechanisms. *Journal of Mathematical Economics*, 19(3): 285–304, 1990.

I. Dusparic and V. Cahill. Distributed w-learning: Multi-policy optimization in self-organizing systems. In *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 20–29. IEEE, 2009.

A. A. Economides, J. A. Silvester, et al. Multi-objective routing in integrated services networks: A game theory approach. In *Infocom*, volume 91, pages 1220–1227, 1991.

E. Eisenstadt, A. Moshaiov, and G. Avigad. Co-evolution of strategies for multi-objective games under postponed objective preferences. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 461–468. IEEE, 2015.

S. Elfwing and B. Seymour. Parallel reward and punishment control in humans and robots: Safe reinforcement learning using the maxpain algorithm. In *2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 140–147. IEEE, 2017.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

J. G. Falcón-Cardona and C. A. C. Coello. Indicator-based multi-objective evolutionary algorithms: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 53(2):1–35, 2020.

P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilen, R. C. Reinhart, and D. J. Mortensen. Reinforcement learning for satellite communications: from leo to deep space operations. *IEEE Communications Magazine*, 57(5):70–75, 2019.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *ICML*, volume 98, pages 197–205, 1998.

L. Galand and T. Lust. Exact methods for computing all lorenz optimal solutions to biobjective problems. In *International Conference on Algorithmic DecisionTheory*, pages 305–321. Springer, 2015.

J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

P. Geibel. Reinforcement learning for MDPs with constraints. In *European Conference on Machine Learning*, pages 646–653. Springer, 2006.

P. Geibel and F. Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

M. Giuliani, S. Galelli, and R. Soncini-Sessa. A dimensionality reduction approach for many-objective markov decision processes: Application to a water reservoir operation problem. *Environmental Modelling & Software*, 57: 101–114, 2014.

M. Giuliani, A. Castelletti, F. Pianosi, E. Mason, and P. M. Reed. Curses, tradeoffs, and scalable management: Advancing evolutionary multiobjective direct policy search to improve water reservoir operations. *Journal of Water Resources Planning and Management*, 142(2):04015050, 2016.

S. Govindaiah and M. D. Petty. Applying reinforcement learning to plan manufacturing material handling part 1: Background and formal problem specification. In *Proceedings of the 2019 ACM Southeast Conference*, pages 168–171, 2019.

F. Grandoni, P. Krysta, S. Leonardi, and C. Ventre. Utilitarian mechanism design for multi-objective optimization. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 573–584. Society for Industrial and Applied Mathematics, 2010.

Y. Guo, A. Zeman, and R. Li. A reinforcement learning approach to setting multi-objective goals for energy demand management. *International Journal of Agent Technologies and Systems (IJATS)*, 1(2):55–70, 2009.

M. M. Hasan, K. Lwin, M. Imani, A. Shabut, L. F. Bittencourt, and M. A. Hossain. Dynamic multi-objective optimisation using deep reinforcement learning: benchmark, algorithm and an application to identify vulnerable zones based on water quality. *Engineering Applications of Artificial Intelligence*, 86:107–135, 2019.

H. Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

C. F. Hayes, M. Reymond, D. M. Roijers, E. Howley, and P. Mannion. Distributional monte carlo tree search for risk-aware and multi-objective reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, volume 2021, 2021a.

C. F. Hayes, M. Reymond, D. M. Roijers, E. Howley, and P. Mannion. Risk-aware and multi-objective decision making with distributional monte carlo tree search. *arXiv preprint arXiv:2102.00966*, 2021b. URL `https://arxiv.org/abs/2102.00966`.

N. Horie, T. Matsui, K. Moriyama, A. Mutoh, and N. Inuzuka. Multi-objective safe reinforcement learning. *Artificial Life and Robotics*, pages 1–9, 2019.

J. Horwood and E. Noutahi. Molecular design in synthetically accessible chemical space via deep reinforcement learning. *arXiv preprint arXiv:2004.14308*, 2020.

X. Hu, Y. Zhang, X. Liao, Z. Liu, W. Wang, and F. M. Ghannouchi. Dynamic beam hopping method based on multi-objective deep reinforcement learning for next generation satellite broadband systems. *IEEE Transactions on Broadcasting*, 2020.

S. H. Huang, M. Zambelli, J. Kay, M. F. Martins, Y. Tassa, P. M. Pilarski, and R. Hadsell. Learning gentle object manipulation with curiosity-driven deep reinforcement learning. *arXiv preprint arXiv:1903.08542*, 2019.

A. Igarashi and D. M. Roijers. Multi-criteria coalition formation games. In *International Conference on Algorithmic Decision Theory*, pages 197–213. Springer, 2017.

A. Ikenaga and S. Arai. Inverse reinforcement learning approach for elicitation of preferences in multi-objective sequential optimization. In *2018 IEEE International Conference on Agents (ICA)*, pages 117–118. IEEE, 2018.

M. Inja, C. Kooijman, M. de Waard, D. M. Roijers, and S. Whiteson. Queued pareto local search for multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 589–599. Springer, 2014.

R. Issabekov and P. Vamplew. An empirical comparison of two common multiobjective reinforcement learning algorithms. In *Australasian Joint Conference on Artificial Intelligence*, pages 626–636. Springer, 2012.

A. Jalalimanesh, H. S. Haghighi, A. Ahmadi, H. Hejazian, and M. Soltani. Multi-objective optimization of radiotherapy: distributed q-learning and agent-based simulation. *Journal of Experimental & Theoretical artificial intelligence*, 29(5):1071–1086, 2017.

J. Jin and X. Ma. A multi-objective agent-based control approach with application in intelligent traffic signal system. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3900–3912, 2019.

C. M. Jonker, R. Aydoğan, T. Baarslag, K. Fujita, T. Ito, and K. Hindriks. Automated negotiating agents competition (anac). In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on Explainable Artificial Intelligence*, 2019.

T. G. Karimpanal and E. Wilhelm. Identification and off-policy learning of multiple objectives using adaptive clustering. *Neurocomputing*, 263:39–47, 2017.

T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and J. development team. Jupyter notebooks - a publishing format for reproducible computational workflows. In F. Loizides and B. Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, Netherlands, 2016. IOS Press. URL `https://eprints.soton.ac.uk/403913/`.

E. Krasheninnikova, J. García, R. Maestre, and F. Fernández. Reinforcement learning for pricing strategy optimization in the insurance industry. *Engineering Applications of Artificial Intelligence*, 80:8–19, 2019.

E. B. Laber, D. J. Lizotte, and B. Ferguson. Set-valued dynamic treatment regimes for competing outcomes. *Biometrics*, 70(1):53–61, 2014.

A. Lacerda. Multi-objective ranked bandits for recommender systems. *Neurocomputing*, 246:12–24, 2017.

C.-S. Lee. Multi-objective game-theory models for conflict analysis in reservoir watershed management. *Chemosphere*, 87(6):608–613, 2012.

K. Lepenioti, M. Pertselakis, A. Bousdekis, A. Louca, F. Lampathaki, D. Apostolou, G. Mentzas, and S. Anastasiou. Machine learning for predictive and prescriptive analytics of operational data in smart manufacturing. In *International Conference on Advanced Information Systems Engineering*, pages 5–16. Springer, 2020.

B. Li, J. Li, K. Tang, and X. Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 48(1):1–35, 2015.

C. Li and K. Czarnecki. Urban driving with multi-objective deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 359–367. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

K. Li, T. Zhang, and R. Wang. Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, 2020.

X. Li, L. Gao, and W. Li. Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. *Expert Systems with Applications*, 39(1):288–297, 2012.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

D. J. Lizotte, M. H. Bowling, and S. A. Murphy. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 695–702. Citeseer, 2010.

C. Ma, J. Wen, and Y. Bengio. Universal successor representations for transfer reinforcement learning. *arXiv preprint arXiv:1804.03758*, 2018.

T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popovic. Where to add actions in human-in-the-loop reinforcement learning. In *AAAI*, pages 2322–2328, 2017.

L. Mandow and J.-L. Pérez-de-la Cruz. Pruning dominated policies in multiobjective Pareto q-learning. In *Conference of the Spanish Association for Artificial Intelligence*, pages 240–250. Springer, 2018.

P. Mannion, J. Duggan, and E. Howley. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pages 47–66. Springer, Cham, 2016. doi: 10.1007/978-3-319-25808-9_4.

P. Mannion, S. Devlin, K. Mason, J. Duggan, and E. Howley. Policy invariance under reward transformations for multi-objective reinforcement learning. *Neurocomputing*, 263, 2017.

P. Mannion, S. Devlin, J. Duggan, and E. Howley. Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning. *The Knowledge Engineering Review*, 33(e23), 2018. URL `https://doi.org/10.1017/S0269888918000292`.

R. Marinescu. Exploiting problem decomposition in multi-objective constraint optimization. In *International Conference on Principles and Practice of Constraint Programming*, pages 592–607. Springer, 2009.

R. Marinescu. Efficient approximation algorithms for multi-objective constraint optimization. In *ADT 2011: Proceedings of the Second International Conference on Algorithmic Decision Theory*, pages 150–164, October 2011.

F. Mello, D. Apostolopoulou, and E. Alonso. Cost efficient distributed load frequency control in power systems. In *21st IFAC World Congress*, February 2020.

B. M. Méndez-Hernández, E. D. Rodríguez-Bazan, Y. Martinez-Jimenez, P. Libin, and A. Nowé. A multi-objective reinforcement learning algorithm for jssp. In *International Conference on Artificial Neural Networks*, pages 567–584. Springer, 2019.

E. J. N. Menezes, A. M. Araújo, and N. S. B. da Silva. A review on wind turbine control and its associated methods. *Journal of Cleaner Production*, 174:945–953, 2018.

C. Messikh and N. Zarour. Towards a multi-objective reinforcement learning based routing protocol for cognitive radio networks. In *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, pages 84–89. IEEE, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

A. Moghaddam, F. Yalaoui, and L. Amodeo. Lorenz versus pareto dominance in a single machine scheduling problem with rejection. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 520–534. Springer, 2011.

H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson. Multi-objective deep reinforcement learning. In *NIPS 2016 Workshop on Deep Reinforcement Learning*, 2016.

A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *Proceedings of Seventh International Conference on Learning Representations*, 2019.

J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

S. Natarajan and P. Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 601–608, 2005.

T. T. Nguyen, N. D. Nguyen, P. Vamplew, S. Nahavandi, R. Dazeley, and C. P. Lim. A multi-objective deep reinforcement learning framework. *Engineering Applications of Artificial Intelligence*, 96:103915, 2020.

M. Nguyena and T. Caoa. A hybrid decision making model for evaluating land combat vehicle system. In *22nd International Congress on Modelling and Simulation, MODSIM2017, Modelling and Simulation Society of Australia and New Zealand*, pages 1399–1405, 2017.

X. Nian, A. A. Irissappane, and D. Roijers. DCRAC: Deep conditioned recurrent actor-critic for multi-objective partially observable environments. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 931–938, 2020.

R. Noothigattu, D. Bouneffouf, N. Mattei, R. Chandra, P. Madan, K. Varshney, M. Campbell, M. Singh, and F. Rossi. Interpretable multi-objective reinforcement learning through policy orchestration. *arXiv preprint arXiv:1809.08343*, 2018.

J. d. D. Ortúzar and L. G. Willumsen. *Modelling transport*. John Wiley & Sons, Chichester, UK, 4 edition, 2011.

A. Pan, W. Xu, L. Wang, and H. Ren. Additional planning with multiple objectives for reinforcement learning. *Knowledge-Based Systems*, 193:105392, 2020.

S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli. Policy gradient approaches for multi-objective sequential decision making. In *IJCNN*, pages 2323–2330. IEEE, 2014.

S. Parisi, M. Pirotta, and M. Restelli. Multi-objective reinforcement learning through continuous pareto manifold approximation. *J. Artif. Intell. Res.*, 57:187–227, 2016.

S. Parisi, M. Pirotta, and J. Peters. Manifold-based multi-objective policy search with sample reuse. *Neurocomputing*, 263:3–14, 2017.

D. Perez, S. Samothrakis, and S. Lucas. Online and offline learning in multi-objective monte carlo tree search. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8. IEEE, 2013.

J. Perez, C. Germain-Renaud, B. Kégl, and C. Loomis. Responsive elastic computing. In *Proceedings of the 6th international conference industry session on Grids meets autonomic computing*, pages 55–64, 2009.

J. Perez, C. Germain-Renaud, B. Kégl, and C. Loomis. Multi-objective reinforcement learning for responsive grids. *Journal of Grid Computing*, 8(3):473–492, 2010.

P. Perny and P. Weng. On finding compromise solutions in multiobjective markov decision processes. In *ECAI*, volume 215, pages 969–970, 2010.

P. Perny, P. Weng, J. Goldsmith, and J. Hanna. Approximation of lorenz-optimal solutions in multiobjective markov decision processes. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 92–94, 2013.

F. Pianosi, A. Castelletti, and M. Restelli. Tree-based fitted q-iteration for multi-objective markov decision processes in water resource management. *Journal of Hydroinformatics*, 15(2):258–270, 2013.

A. Pla, B. Lopez, and J. Murillo. Multi criteria operators for multi-attribute auctions. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 318–328. Springer, 2012.

Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai. An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning. *The Journal of Supercomputing*, 76(1):455–480, 2020.

S. Qu, Y. Ji, and M. Goh. The robust weighted multi-objective game. *PloS one*, 10(9):e0138970, 2015.

R. N. Raj, A. Nayak, and M. S. Kumar. A survey and performance evaluation of reinforcement learning based spectrum aware routing in cognitive radio ad hoc networks. *International Journal of Wireless Information Networks*, 27(1): 144–163, 2020.

G. de. O. Ramos, R. Rădulescu, A. Nowé, and A. R. Tavares. Toll-based learning for minimising congestion under heterogeneous preferences. In B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, and G. Sukthankar, editors, *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, pages 1098–1106, Auckland, New Zealand, May 2020. IFAAMAS.

N. B. Ravichandran, F. Yang, C. Peters, A. Lansner, and P. Herman. Pedestrian simulation as multi-objective reinforcement learning. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, pages 307–312, 2018.

M. J. Reddy and D. N. Kumar. Optimal reservoir operation using multi-objective evolutionary algorithm. *Water Resources Management*, 20(6):861–878, 2006.

M. Reymond and A. Nowé. Pareto-dqn: Approximating the pareto front in complex multi-objective decision problems. In *Proceedings of the adaptive and learning agents workshop (ALA-19) at AAMAS*, 2019.

D. Roijers, L. Zintgraf, P. Libin, and A. Nowe. Interactive multi-objective reinforcement learning in multi-armed bandits for any utility function. In *Proceedings of the adaptive and learning agents workshop (ALA-18) at AAMAS*, 07 2018a.

D. M. Roijers. *Multi-Objective Decision-Theoretic Planning*. PhD thesis, University of Amsterdam, 2016.

D. M. Roijers and S. Whiteson. Multi-objective decision making. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 11(1):1–129, 2017.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443, 2015a.

D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Point-based planning for multi-objective pomdps. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence (IJCAI)*, pages 1666–1672, 2015b.

D. M. Roijers, L. M. Zintgraf, and A. Nowé. Interactive thompson sampling for multi-objective multi-armed bandits. In *International Conference on Algorithmic Decision Theory*, pages 18–34. Springer, 2017.

D. M. Roijers, D. Steckelmacher, and A. Nowé. Multi-objective reinforcement learning for the expected utility of the return. In *Proceedings of the Adaptive and Learning Agents workshop at FAIM*, volume 2018, 2018b.

D. M. Roijers, E. Walraven, and M. T. J. Spaan. Bootstrapping LPs in value iteration for multi-objective and partially observable MDPs. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS)*, pages 218–226, 2018c.

D. M. Roijers, L. M. Zintgraf, P. Libin, M. Reymond, E. Bargiacchi, and A. Nowé. Interactive multi-objective reinforcement learning in multi-armed bandits with gaussian process utility models. In *ECML-PKDD 2020: Proceedings of the 2020 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, page 16, 2020.

E. Rollón. *Multi-Objective Optimization for Graphical Models*. PhD thesis, Universitat Politècnica de Catalunya, Barcelona, 2008.

E. Rollón and J. Larrosa. Bucket elimination for multiobjective optimization problems. *Journal of Heuristics*, 12: 307–328, 2006.

E. Rollon and J. Larrosa. Multi-objective russian doll search. In *AAAI*, pages 249–254, 2007.

E. Rollon and J. Larrosa. Constraint optimization techniques for multiobjective branch and bound search. In *International conference on logic programming, ICLP*, 2008.

J. Rowe, A. Smith, B. Pokorny, B. Mott, and J. Lester. Toward automated scenario generation with deep reinforcement learning in gift. In *Proceedings of the Sixth Annual GIFT User Symposium*, pages 65–74, 2018.

R. Rădulescu, P. Mannion, D. M. Roijers, and A. Nowé. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(10), 2020a.

R. Rădulescu, P. Mannion, Y. Zhang, D. M. Roijers, and A. Nowé. A utility-based analysis of equilibria in multi-objective normal-form games. *The Knowledge Engineering Review*, 35:e32, 2020b. doi: 10.1017/S0269888920000351.

M. Ruiz-Montiel, L. Mandow, and J.-L. Pérez-de-la Cruz. A temporal difference method for multi-objective reinforcement learning. *Neurocomputing*, 263:15–25, 2017.

S. Saisubramanian, E. Kamar, and S. Zilberstein. A multi-objective approach to mitigate negative side effects. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

N. Shabani. *Incorporating flood control rule curves of the Columbia River hydroelectric system in a multireservoir reinforcement learning optimization model*. PhD thesis, University of British Columbia, 2009.

U. Siddique, P. Weng, and M. Zimmer. Learning fair policies in multiobjective (deep) reinforcement learning with average and discounted rewards. In *International Conference on Machine Learning*, 2020.

H. Soh and Y. Demiris. Evolving policies for multi-reward partially observable markov decision processes (MR-POMDPs). In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 713–720, 2011a.

H. Soh and Y. Demiris. Multi-reward policies for medical applications: Anthrax attacks and smart wheelchairs. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 471–478, 2011b.

Y. Sun, Y. Li, W. Xiong, Z. Yao, K. Moniz, and A. Zahir. Pareto optimal solutions for network defense strategy selection simulator in multi-objective reinforcement learning. *Applied Sciences*, 8(1):136, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

T. Tajmajer. Modular multi-objective deep reinforcement learning with decision values. In *Federated conference on computer science and information systems (FedCSIS)*, pages 85–93. IEEE, 2018.

A. Taylor, I. Dusparic, E. Galván-López, S. Clarke, and V. Cahill. Accelerating learning in multi-objective systems through transfer learning. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 2298–2305. IEEE, 2014.

G. Tesauro, R. Das, H. Chan, J. Kephart, D. Levine, F. Rawson, and C. Lefurgy. Managing power consumption and performance of computing systems using reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1497–1504, 2008.

L. Thomas. *Constrained Markov decision processes as multi-objective problems*. University of Manchester. Department of Decision Theory, 1982.

B. Tozer, T. Mazzuchi, and S. Sarkani. Many-objective stochastic path finding using reinforcement learning. *Expert Systems with Applications*, 72:371–382, 2017.

A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2016.

E. Turgay, D. Oner, and C. Tekin. Multi-objective contextual bandit problem with similarity information. In *International Conference on Artificial Intelligence and Statistics*, pages 1673–1681, 2018.

P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry. On the limitations of scalarisation for multi-objective reinforcement learning of Pareto fronts. In *Australasian Joint Conference on Artificial Intelligence*, pages 372–378. Springer, 2008.

P. Vamplew, R. Dazeley, E. Barker, and A. Kelarev. Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In *Australasian joint conference on artificial intelligence*, pages 340–349. Springer, 2009.

P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, 84(1-2):51–80, 2011.

P. Vamplew, R. Issabekov, R. Dazeley, and C. Foale. Reinforcement learning of Pareto-optimal multiobjective policies using steering. In *Australasian Joint Conference on Artificial Intelligence*, pages 596–608. Springer, 2015.

P. Vamplew, R. Dazeley, and C. Foale. Softmax exploration strategies for multiobjective reinforcement learning. *Neurocomputing*, 263:74–86, 2017a.

P. Vamplew, R. Issabekov, R. Dazeley, C. Foale, A. Berry, T. Moore, and D. Creighton. Steering approaches to Pareto-optimal multiobjective reinforcement learning. *Neurocomputing*, 263:26–38, 2017b.

P. Vamplew, R. Dazeley, C. Foale, S. Firmin, and J. Mummery. Human-aligned artificial intelligence is a multiobjective problem. *Ethics and Information Technology*, 20(1):27–40, 2018.

P. Vamplew, C. Foale, and R. Dazeley. The impact of environmental stochasticity on value-based multiobjective reinforcement learning. *Neural Computing and Applications*, 2021a. doi: 10.1007/s00521-021-05859-1.

P. Vamplew, C. Foale, R. Dazeley, and A. Bignold. Potential-based multiobjective reinforcement learning approaches to low-impact agents for AI safety. *Engineering Applications of Artificial Intelligence*, 100, 2021b. doi: 10.1016/j.engappai.2021.104186.

M. T. van Dijk, J.-W. van Wingerden, T. Ashuri, Y. Li, and M. A. Rotea. Yaw-misalignment and its impact on wind turbine loads and wind farm power output. *Journal of Physics: Conference Series*, 753(6), 2016.

K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

K. Van Moffaert, M. M. Drugan, and A. Nowé. Hypervolume-based multi-objective reinforcement learning. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 352–366. Springer, 2013a.

K. Van Moffaert, M. M. Drugan, and A. Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–199. IEEE, 2013b.

K. Van Moffaert, T. Brys, A. Chandra, L. Esterle, P. R. Lewis, and A. Nowé. A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning. In *2014 International joint conference on neural networks (IJCNN)*, pages 2306–2314. IEEE, 2014.

K. Van Vaerenbergh, A. Rodríguez, M. Gagliolo, P. Vrancx, A. Nowé, J. Stoev, S. Goossens, G. Pinte, and W. Symens. Improving wet clutch engagement with reinforcement learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

T. Verstraeten, A. Nowé, J. Keller, Y. Guo, S. Sheng, and J. Helsen. Fleetwide data-enabled reliability improvement of wind turbines. *Renewable and Sustainable Energy Reviews*, 109:428–437, 2019.

T. Verstraeten, E. Bargiacchi, P. J. K. Libin, J. Helsen, D. M. Roijers, and A. Nowé. Multi-agent thompson sampling for bandit applications with sparse neighbourhood structures. *Scientific Reports*, 10, 2020. doi: 10.1038/s41598-020-62939-3.

C. Von Lücken, B. Barán, and C. Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational optimization and applications*, 58(3):707–756, 2014.

W. Wallach and C. Allen. *Moral machines: Teaching robots right from wrong*. Oxford University Press, 2008.

H. Wang, Z. Lei, X. Zhang, J. Peng, and H. Jiang. Multiobjective reinforcement learning-based intelligent approach for optimization of activation rules in automatic generation control. *IEEE Access*, 7:17480–17492, 2019.

W. Wang and M. Sebag. Multi-objective Monte-Carlo tree search. volume 25 of *Proceedings of Machine Learning Research*, pages 507–522, Singapore, Nov 2012. PMLR.

W. Wang and M. Sebag. Hypervolume indicator and dominance reward based multi-objective monte-carlo tree search. *Machine learning*, 92(2-3):403–429, 2013.

N. Wanigasekara, Y. Liang, S. T. Goh, Y. Liu, J. J. Williams, and D. S. Rosenblum. Learning multi-objective rewards and user utility function in contextual bandits for personalized ranking. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3835–3841. AAAI Press, 2019.

D. Weng, R. Chen, J. Zhang, J. Bao, Y. Zheng, and Y. Wu. Pareto-optimal transit route planning with multi-objective monte-carlo tree search. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

C. C. White and K. W. Kim. Solution procedures for vector criterion Markov decision processes. *Large Scale Systems*, 1:129–140, 1980.

D. White. Multi-objective infinite-horizon discounted markov decision processes. *Journal of mathematical analysis and applications*, 89(2):639–647, 1982.

M. A. Wiering and E. D. De Jong. Computing optimal stationary policies for multi-objective markov decision processes. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 158–165. IEEE, 2007.

M. A. Wiering, M. Withagen, and M. M. Drugan. Model-based multi-objective reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–6. IEEE, 2014.

K. H. Wray and S. Zilberstein. Multi-objective pomdps with lexicographic reward preferences. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

K. H. Wray, S. Zilberstein, and A.-I. Mouaddib. Multi-objective mdps with conditional lexicographic reward preferences. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

S. Q. Yahyaa, M. M. Drugan, and B. Manderick. Knowledge gradient for multi-objective multi-armed bandit algorithms. In *ICAART (1)*, pages 74–83, 2014.

T. Yamaguchi, S. Nagahama, Y. Ichikawa, and K. Takadama. Model-based multi-objective reinforcement learning with unknown weights. In *International Conference on Human-Computer Interaction*, pages 311–321. Springer, 2019.

C. Yang, J. Lu, X. Gao, H. Liu, Q. Chen, G. Liu, and G. Chen. MoTiAC: Multi-objective actor-critics for real-time bidding. *arXiv preprint arXiv:2002.07408*, 2020.

R. Yang, X. Sun, and K. Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, pages 14636–14647, 2019.

L. Yliniemi and K. Tumer. Multi-objective multiagent credit assignment in reinforcement learning and nsga-ii. *Soft Computing*, 20(10):3869–3887, 2016.

H. Yu and H. Liu. Robust multiple objective game theory. *Journal of Optimization Theory and Applications*, 159(1): 272–280, 2013.

H. Zhan and Y. Cao. Relationship explainable multi-objective reinforcement learning with semantic explainability generation. *arXiv preprint arXiv:1909.12268*, 2019.

Y. Zhang, R. Rădulescu, P. Mannion, D. M. Roijers, and A. Nowé. Opponent modelling for reinforcement learning in multi-objective normal form games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2080–2082, 2020.

Z. Zhang, A. Chong, Y. Pan, C. Zhang, and K. P. Lam. Whole building energy model for hvac optimal control: A practical framework based on deep reinforcement learning. *Energy and Buildings*, 199:472–490, 2019.

Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

L. M. Zintgraf, T. V. Kanters, D. M. Roijers, F. Oliehoek, and P. Beau. Quality assessment of MORL algorithms: A utility-based approach. In *Benelearn 2015: Proceedings of the 24th Annual Machine Learning Conference of Belgium and the Netherlands*, 2015.

L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and A. Nowé. Ordered preference elicitation strategies for supporting multi-objective decision making. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1477–1485. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

E. Zitzler, J. Knowles, and L. Thiele. Quality assessment of pareto set approximations. In *Multiobjective Optimization*, pages 373–404. Springer, 2008.