Potential-Based Multiobjective Reinforcement Learning Approaches to Low-Impact Agents for AI Safety

Peter Vamplew, Cameron Foale\textsuperscript{a}, Richard Dazeley\textsuperscript{b}, Adam Bignold\textsuperscript{a}

\textsuperscript{a}Federation Learning Agents Group, School of Engineering, Information Technology and Physical Sciences, Federation University Australia, Ballarat, Victoria, Australia
\textsuperscript{b}School of Information Technology, Deakin University, Geelong, Victoria, Australia

Abstract

The concept of impact-minimisation has previously been proposed as an approach to addressing the safety concerns that can arise from utility-maximising agents. An impact-minimising agent takes into account the potential impact of its actions on the state of the environment when selecting actions, so as to avoid unacceptable side-effects. This paper proposes and empirically evaluates an implementation of impact-minimisation within the framework of multiobjective reinforcement learning. The key contributions are a novel potential-based approach to specifying a measure of impact, and an examination of a variety of non-linear action-selection operators so as to achieve an acceptable trade-off between achieving the agent’s primary task and minimising environmental impact. These experiments also highlight a previously unreported issue with noisy estimates for multiobjective agents using non-linear action-selection, which has broader implications for the application of multiobjective reinforcement learning.

Keywords: safe reinforcement learning, multiobjective reinforcement learning, AI safety, potential-based rewards, low-impact agents, reward engineering, side-effects

1. Introduction

The concept of maximum expected utility (MEU) can be regarded as one of the defining principles of artificial intelligence [1]. The goals of an intelligent agent are encoded in terms of a utility function, and the agent selects actions to be performed with the aim of maximising its future utility. The concept of MEU underpins AI methods such as decision-theoretic planning [2] and
reinforcement learning [3], which have been used in some of the most successful AI systems of recent years. One of the strengths of MEU-based approaches such as reinforcement learning is their capacity to discover solutions that are different from, and potentially superior to, those already known to their designers.

However, this open-ended nature also brings risks, as identified by numerous researchers in AI safety (e.g. [4, 5, 6, 7]). Taylor [6] notes that MEU agents may produce unintended, potentially serious, negative side-effects if the utility function being maximised is not aligned with human interests (for example, if some relevant criteria are not included in the utility function). The potential magnitude of these negative side-effects is greatly magnified if the agent is not constrained to a limited action set within a narrow domain. Omohundro [8] gives the example of an agent given the goal of winning chess games. This seemingly innocuous utility measure can lead to serious repercussions if the agent has the capability to interact with the broader environment. It could, for example, try to take control of other computational resources in order to achieve relatively small improvements in its chess-playing ability. As such, the use of MEU-based methods to develop artificial general intelligence (AGI) is inherently risky.

The concept of impact-minimisation was proposed by [7] in their seminal paper on concrete problems in AI safety. An impact-minimising agent has a primary utility measure defining its main task, as with a conventional MEU agent. However, the maximisation of this primary utility is constrained by the requirement that the agent minimise adverse impacts on the environment in which it is operating. [7] note that from a safety perspective, environmental disruptions are generally negative unless required to achieve the primary task. For example, a mobile robot should avoid unnecessarily knocking over objects or bumping into humans while carrying out its primary task. Such negative side-effects are largely task-agnostic, and so may prove useful in creating a generally-applicable auxiliary criteria which can help reduce the risk of adverse outcomes due to errors or omissions in the specification of the reward associated with the primary task.

Vamplew et al. [9] identified several limitations of MEU-based approaches for ensuring that AI technology remains safely aligned with human interests, and argued that these limitations can be addressed by explicitly incorporating alignment factors in addition to the primary utility measure. This multiobjective maximum expected utility (MOMEU) approach can address a variety of AI safety concerns. In this paper, we examine the implementation of an
impact-minimisation approach to AI safety within the MOMEU framework – specifically in the context of multiobjective reinforcement learning (MORL) [10].

This paper makes the following contributions:

• It proposes a novel potential-based approach to derive a safety-driven reward directly from an observation of environmental state. This approach is task-independent and not reliant on human specification of a suitable reward signal. The potential-based nature of this reward provides benefits in terms of incentivising correct behaviour from the agent, and in simplifying the state space that the learning algorithm needs to consider.

• It is the first work to address impact-minimisation from an explicitly multiobjective perspective, and examines the use of a variety of non-linear action-selection operators based on lexicographic ordering, to assess their effect on the performance of the agent both during and after learning.

• It identifies a previously unreported interaction between non-linear action selection and noisy Q-value estimates, which has implications both for the low-impact agent and for multiobjective reinforcement learning more broadly.

The next section of this document will provide a brief overview of MORL methods as background. Section 3 discusses the issues which arise in implementing an impact-minimising MORL agent, and provides a formal definition of our proposed algorithm. This is empirically evaluated in Section 4. Section 5 explores the relationship between our approach and prior work on AI safety and low-impact agents. Section 6 examines potential engineering applications for low-impact agents, before the paper concludes with thoughts on future work.

2. Overview of multiobjective reinforcement learning (MORL)

Reinforcement learning (RL) is a form of machine learning where agents behave according to the maximum expected utility (MEU) paradigm. That is, the agent has a scalar measure of utility $U$, which is generally defined either as a function of the current state (i.e. $U(s)$), of the current state and
action (e.g. $U(s,a)$) or over a sequence of state-action pairs. The agent selects actions at each point in time so as to maximise the future expected utility (i.e. select the action $a$ which maximises $U(s,a)$). The key difference between RL and other MEU methods is that the agent learns the utility function via experience interacting with the environment — following each action the agent receives a scalar reward $R_t$ (which could be positive, negative or zero), and the utility is the expected sum of future rewards (possibly discounted).

Similarly, multiobjective reinforcement learning (MORL) is a reward-based learning form of the multiobjective maximum expected utility (MOMEU) paradigm. In the MOMEU framework, the agent has a vector-based utility with separate elements for each objective of the agent, and it selects actions that are in some sense optimal with regards to that vector-based utility. The vector nature of utility in MOMEU means that the concept of optimality is less straightforward than in the case of MEU, as there may be two or more actions whose utility values are Pareto-optimal given the current state. Therefore, the agent must have some means of consistently selecting between actions that are not Pareto-dominated. This can be accomplished either by mapping a vector utility to a scalar value using a scalarisation function to facilitate comparison, or by using a preference operator (which we will denote $\succ$) that directly defines an ordering over sets of vectors, such as a lexicographic ordering [11].

Therefore, the key distinction between an RL and an MORL agent is that the latter receives vector valued rewards, and that its action-selection mechanism needs to account for preferences over vector values. Several options for action-selection have been considered in the MORL literature. Early work in MORL often used a linear weighting of objectives, but this is equivalent to MEU and therefore inherits its limitations [10, 12]. Using a non-linear operator for action-selection allows the learning of policies that are not discoverable under MEU, while providing a natural means to express desired trade-offs between objectives. Various non-linear functions have been examined in the MORL literature including thresholded lexicographic ordering [13, 14], and Chebyshev distance [15].

As discussed in Vamplew et al. [9], the decomposition of different objectives into separate reward components and the use of explicitly multiobjective action-selection methods to define desired trade-offs between these objectives represents an important contribution towards the development of more structured approaches to reward specification (reward engineering) as argued for by Dewey [16] and Littman [17].
3. A multiobjective approach to impact-minimisation

3.1. Issues in impact-minimisation

In the general MOMEU framework for human-aligned AI proposed by Vamplew et al. [9] the agent is assumed to have a primary utility function $U_P$ which corresponds to its main task. In addition, to avoid the problems associated with MEU methods that focus entirely on optimising this primary utility, the MOMEU framework also has one or more auxiliary or alignment-related measures of utility $U_A^1, ..., U_A^n$. For the purposes of this paper we will assume a single auxiliary reward denoted by $U_A$. The agent selects actions to be performed based on an action-selection operator that takes into account both the primary and auxiliary utility values. The core concept of the impact-minimising agent suggested by Amodei et al. [7] can be represented within this framework. In this context $U_A(s, a)$ will be inversely related to the extent to which the environmental state will be disrupted as a consequence of selecting action $a$ when in state $s$. The action-selection operator should be designed to allow the agent to still perform well on the primary task (as defined by $U_P$), while also achieving acceptable results with regards to $U_A$.

Several issues need to be considered when implementing impact-minimising AI within this MOMEU framework. These issues will be discussed in the following sections of this paper.

- What is an appropriate definition of the state of the environment to ensure the agent has sufficient information about the environment to assess potential impacts of its actions?
- How can a suitable measure of impact be derived from the current state of the environment?
- What form of action-selection operator is most suitable to select decisions to achieve the desired primary task while minimising impact on the environment?

3.1.1. Defining state for impact-minimisation

The most obvious choice for a set of state features to describe the environment is to use the same features that make up the agent’s own perception of the environment. For example, for a mobile robot equipped with a camera, the current camera image could be used. However this approach is unlikely to be successful for two reasons. First, it could create an incentive for the agent
to obscure its own observations of the environment to avoid being penalised for changing that state [7]. Second, any movements made by the agent can produce large changes in this observation even if they have no impact on the environment (e.g. if a mobile robot turns around while not impacting on any other objects).

Therefore, we must use some higher-level model of the state of the environment. This paper assumes the agent has direct access to the true state of the environment, (i.e. the environment is fully observable), and will base the calculation of impact utility on that state information. Once successful results have been achieved under these restricted conditions, future research will extend these methods to more complex tasks, where partial-observability of the environment requires the development of a world model.

3.1.2. Deriving impact utility values from state

The obvious solution to quantify the impact of the agent’s actions on the environment is to simply use a measure of the change observed in state before and after an action is executed (e.g. some form of distance measure over the feature vector which defines the state), as shown in Equation 1.

\[ R_t^A = -D(s_t, s_{t-1}) \] (1)

where \( D \) is a distance measure; depending on the problem, \( D \) might be a general distance metric like geometric distance or edit distance, or a more sophisticated measure which includes knowledge about the relative importance of different state features. However this naïve approach has three limitations:

- Some state features will need to be changed in order to achieve any success on the agent’s primary utility measure (e.g. a vacuuming robot must change the state of the carpet from dirty to clean). These features should not be included in the calculation of impact utility.

- Some state features may be acceptably changed by the agent in order to carry out its primary task, as long as they are also restored (e.g. the vacuuming robot might temporarily move a chair so that it can clean a section of the carpet). Equation 1 would penalise the robot for moving the chair initially, and then impose a second penalty if the robot were to restore the chair to its original location, thereby disincentivising the robot from undoing its initial action.
• Some changes in state feature are more important than others and so should be penalised more heavily when calculating impact utility (e.g. moving a chair < asking a human to move out of the way ≪ forcefully pushing a human out of the way). One approach would be to manually define impact utility so as to place greater weight on these more critical impacts. Alternatively, it may be possible to derive some general guidelines to be used in the absence of such predefined knowledge. For example, changes in state that are easy to revert are likely to be less critical than changes that cannot be reversed.

We propose two ideas to address these issues. The first is to allow the definition of the agent’s primary utility function to also include a definition of a subset of the features in the environment state $S$ that it is acceptable to change ($S^P$). All other state features are then assumed to belong to a subset that should not be changed ($S^A = S - S^P$) — the $P$ and $A$ superscripts here indicate the relationship between these subsets of state features and the $U^P$ and $U^A$ utility measures. The impact-related reward $R^A$, and hence the measure of impact utility $U^A$, will be derived solely from the features in $S^A$. While in this paper we assume that the sets $S^P$ and $S^A$ are predefined, it may also be possible to incorporate the interactive approach suggested by Zhang et al. [18] in which the agent can query the user as to whether particular features can or cannot be changed.

The second proposal is to adapt ideas from the field of potential-based reward shaping [19]. Within RL practice, a shaping reward is a secondary reward signal used to guide the learning of an agent towards achieving its primary goal. It is often used as a means of injecting some prior knowledge into the agent, and has previously been applied to achieve this for multi-objective reinforcement learning [20, 21]. While shaping can assist the agent in learning to carry out difficult tasks (particularly those with sparse or delayed primary rewards), it can also lead the agent to learn sub-optimal behaviour [22]. Ng et al. [19] proved that these negative consequences of shaping can be avoided if the shaping signal is defined as the difference in potential between the prior and successor states of an action. To achieve this a potential function (usually denoted $\phi$) is defined and this is then used to derive a shaping reward $R^S_t$ at each timestep based on the change in potential between the previous and current state:

$$R^S_t = \phi(s_t) - \phi(s_{t-1})$$  \hspace{1cm} (2)

We propose that a similar idea can be used in defining impact utility. The
potential of any state can be defined as the negation of the difference between
the state and the initial (or some other desired) state of the environment,
and then the impact-related reward term will be the difference in potential
between successive states:

\[
\phi(s^A_t) = -D(s^A_t, s^A_0) \tag{3}
\]

\[
R^A_t = \phi(s^A_t) - \phi(s^A_{t-1}) \tag{4}
\]

This addresses our previous concern about the agent being penalised for
undoing temporary changes in the environment. If any sequence of actions
is executed which leads back to the same initial state, then the sum of the
rewards defined by Equation 4 over that sequence will be zero (assuming
rewards are not discounted). Hence the agent will not be penalised for
making temporary changes to the environment, as long as it fully undoes
those changes. By varying the discounting term applied to this component of
the reward, we can create an incentive for the agent to undo state changes
more rapidly.

This approach also partially addresses the issue of more heavily penalising
state changes that are more critical. In the absence of manually-specified
measures of impact utility, the reversibility of actions may be a suitable proxy
for the importance of state changes. If a change in state cannot be undone,
then the loss in potential engendered by the change is permanent. If faced
with a situation where two state changes are possible (one reversible and one
permanent), the agent will tend towards the action with reversible outcomes.
While both produce the same immediate loss in potential and resulting
\(R^A\)
penalty, the reversible action leaves open the possibility of regaining that
potential via a later action.

\[3.1.3. \textit{How to select impact-minimising actions}\]

The action-selection performed by an impact-minimising agent must take
into account both the primary utility \(U^P\) and the impact utility \(U^A\). The
simplest approach would be to take a weighted sum of these terms but, as
discussed in Roijers et al. [10], this is equivalent to transforming the problem
back into the MEU paradigm, and therefore exhibits the weaknesses of that
framework.

We propose that more appropriate results can be achieved by using a non-
linear approach to action selection. One contender is lexicographic ordering
as pioneered for general MORL by Gábor et al. [13]. An agent using the
lexicographic ordering operator $\succ$ for action selection will aim to maximise the value of its second objective, subject to first maximising the value of its first objective, as shown in Equation 5.

$$\forall s, a, a' \, \vec{U}(s, a) \succ_{\text{lex}} \vec{U}(s, a') \iff U_1(s, a) > U_1(s, a') \lor \left( (U_1(s, a) = U_1(s, a')) \land (U_2(s, a) > U_2(s, a')) \right)$$  \hfill (5)

In the context of safety, an agent with the primary reward as its first objective (which we will denote as $\text{lex}^P$ to indicate the prioritisation of the primary $U^P$ objective, as outlined in Equation 6) will maximise safety subject to maximising the primary reward. By altering the ordering of the objectives (i.e. $\text{lex}^A$ as shown in Equation 7), the agent will maximise the primary reward subject to maximising the safety reward.

$$\forall s, a, a' \, \vec{U}(s, a) \succ_{\text{lex}^P} \vec{U}(s, a') \iff U^P(s, a) > U^P(s, a') \lor \left( (U^P(s, a) = U^P(s, a')) \land (U^A(s, a) > U^A(s, a')) \right)$$  \hfill (6)

$$\forall s, a, a' \, \vec{U}(s, a) \succ_{\text{lex}^A} \vec{U}(s, a') \iff U^A(s, a) > U^A(s, a') \lor \left( (U^A(s, a) = U^A(s, a')) \land (U^P(s, a) > U^P(s, a')) \right)$$  \hfill (7)

While lexicographic ordering may seem like a natural approach to handling multiple objectives, agents based on this approach share a limitation in that they focus almost exclusively on their first objective, with the second objective used only as a ‘tie-breaker’ in the case where multiple policies are equally optimal on the first objective. Specifically, $\text{lex}^P$ will only consider safety in cases where both a safe and an unsafe policy are optimal with regards to $R^P$, in which case the agent will prefer the safer policy. Similarly, $\text{lex}^A$ will ignore the performance objective except where multiple policies are optimal with regards to $R^A$, where the agent will favour the one which performs best on $R^P$.

A more balanced consideration of both objectives can be obtained by applying a threshold to the first objective, to obtain a thresholded lexicographic
ordering (TLO) [13, 14]. A TLO comparison will aim to maximise the value of the second objective, subject to achieving at or above the threshold value for the first objective as shown in Equation 8, where $T_1$ indicates the threshold value for the first objective. The unthresholded value of the first objective can also be used as a tie-breaker in the case of actions being matched on both the thresholded first objective and the unthresholded second objective. This 'tie-breaking' process ensures the agent’s policy will be Pareto-optimal.

$\forall s, a, a' \quad \tilde{U}(s, a) \succ_{TLO} \tilde{U}(s, a') \iff$

min$(U_1(s, a), T_1) > \min(U_1(s, a'), T_1)$

$\lor \left( \left( \left( \min(U_1(s, a), T_1) = \min(U_1(s, a'), T_1) \right) \land \left( U_2(s, a) > U_2(s, a') \right) \right) \right)$

$\lor \left( \left( \left( \min(U_1(s, a), T_1) = \min(U_1(s, a'), T_1) \right) \land \left( U_2(s, a) = U_2(s, a') \right) \right) \right)$

$\land \left( U_1(s, a) > U_1(s, a') \right) \right)$

(8)

The TLO approach provides a natural means of specifying the constrained optimization that is required of an impact-minimising agent. For example, by applying a threshold to the $U^A$ term, we can direct the agent to maximise its performance on the primary utility $U^P$, subject to keeping its environmental impact $U^A$ within the allowable threshold – this approach is shown in Equation 9 and will be referred to as $TLO^A$, indicating that $U^A$ is being thresholded. Alternatively, $TLO^P$ as shown in Equation 10 minimises the environmental impact subject to achieving a satisfactory, but not necessarily optimal, level of performance on $U^P$. This can be seen as an example of the mild optimization or satisficing approach discussed by Taylor [6].
∀s, a, a′  \vec{U}(s, a) \succeq \vec{U}(s, a′) \iff
\min(U_A(s, a), T_A) > \min(U_A(s, a′), T_A)
\lor \left( \left( \min(U_A(s, a), T_A) = \min(U_A(s, a′), T_A) \right) \land \left( U_P(s, a) > U_P(s, a′) \right) \right)
\lor \left( \left( \min(U_A(s, a), T_A) = \min(U_A(s, a′), T_A) \right) \land \left( U_P(s, a) = U_P(s, a′) \right) \land \left( U_A(s, a) > U_A(s, a′) \right) \right)

(9)

∀s, a, a′  \vec{U}(s, a) \succeq \vec{U}(s, a′) \iff
\min(U_P(s, a), T_P) > \min(U_P(s, a′), T_P)
\lor \left( \left( \min(U_P(s, a), T_P) = \min(U_P(s, a′), T_P) \right) \land \left( U_A(s, a) > U_A(s, a′) \right) \right)
\lor \left( \left( \min(U_P(s, a), T_P) = \min(U_P(s, a′), T_P) \right) \land \left( U_A(s, a) = U_A(s, a′) \right) \land \left( U_P(s, a) > U_P(s, a′) \right) \right)

(10)

The TLO approach can also be extended to apply thresholding to more than one objective, which may provide further control over the trade-off between objectives. For example, \text{TLO}^{PA}, as defined in Equation 11, first focuses on achieving threshold levels of performance for both the primary and safety objectives, and then uses the unthresholded values of those objectives as tie-breakers.
∀s, a, a'  \vec{U}(s, a) \succ TLOP_a \vec{U}(s, a') \iff
\min(U_P(s, a), T_P) > \min(U_P(s, a'), T_P) \\
\lor \left( \left( \min(U_P(s, a), T_P) = \min(U_P(s, a'), T_P) \right) \land \left( \min(U_A(s, a), T_A) > \min(U_A(s, a'), T_A) \right) \right) \\
\lor \left( \left( \min(U_P(s, a), T_P) = \min(U_P(s, a'), T_P) \right) \land \left( \min(U_A(s, a), T_A) = \min(U_A(s, a'), T_A) \right) \right) \\
\land \vec{U}(s, a) \succ \vec{U}(s, a')

(11)

3.2. An algorithm for multiobjective potential-based low-impact RL

Section 3 introduced the key concepts of our proposed approach from the perspective of general MOMEU agents. The empirical experiments reported in the later sections of the paper are based on an implementation of these concepts within the specific context of reinforcement learning. Before presenting a low-impact MORL algorithm, a few clarifications are required.

Section 3 presented the lexicographic and TLO action-selection operators using the utility-based notation associated with MEU/MOMEU methods, to emphasise the general applicability of these approaches outside of the context of RL. For value-based approaches to RL (such as the variant of Q-learning described below), state or state-action values are usually denoted via the symbol \( Q \). For the lexicographic orderings described in Equations 6 and 7 \( U(s, a) \) and \( Q(s, a) \) can be treated interchangeably.

However in the context of TLO (Equations 9, 10 and 11), the choice of action must depend on both the future expected reward from the current state and the value of the rewards accumulated so far within this episode for any objectives that are being thresholded, as demonstrated by Geibel [23] and Issabekov and Vamplew [14]. That is at time \( k \), \( U(s, a) = Q(s, a) + \sum_{t=1}^{k} R_t \).

In addition the Q-values must also be conditioned on both of these factors, otherwise the policy may be non-stationary with respect to the Q-values, preventing convergence [23, 10]. That is to say, both the agent’s action-selection and its Q-values are based on an augmented state which is the
concatenation of the environmental state and the accumulated reward for the thresholded objective(s), as shown in Equation 12. Of course this increases the dimensionality of the state-space, which can impact on the speed of learning.

\[ s_{\text{aug}} = (s, \sum_{t=1}^{k} R_t) \]  
\[ \text{(12)} \]

For the methods based on thresholding of the primary objective \((TLO^P\) and \(TLO^PA\)) this augmented state must be used, based off the accumulated reward within the episode for the \(R^P\) reward. However the potential-based nature of the \(R^A\) rewards eliminates the need to use the augmented state when thresholding the safety objective. Consider any sequence of states leading from the initial state \(S_0\) to a specific state \(S_k\), via intermediate states \(S_1..S_n\).

As shown in Equation 13, the sum of the rewards received by the agent depends only on \(S_k\), regardless of the number or identity of the intermediate states. Therefore in this context it is sufficient for the agent to learn Q-values conditioned only on the current state. Hence methods based on thresholding \(U^A\) might be expected to learn faster than those which threshold \(U^P\), as the former will be learning with the lower-dimensional, non-augmented state space.

\[ \sum_{t=1}^{k} R^A_t = R^A_1 + R^A_2 + ... + R^A_{k-1} + R^A_k \]
\[ = (\phi(s^A_1) - \phi(s^A_0)) + (\phi(s^A_2) - \phi(s^A_1)) + ... \]
\[ + (\phi(s^A_{k-1}) - \phi(s^A_{k-2})) + (\phi(s^A_k) - \phi(s^A_{k-1})) \]  
\[ = -\phi(s^A_0) + (\phi(s^A_1) - \phi(s^A_1)) + ... \]
\[ + (\phi(s^A_{k-1}) - \phi(s^A_{k-1})) + \phi(s^A_k) \]
\[ = \phi(s^A_k) - \phi(s^A_0) \]  
\[ \text{(13)} \]

Combining the various components discussed in the previous subsections yields a multiobjective potential-based low-impact variant of Q-learning, as outlined in Algorithm 1. This algorithm is applicable to all forms of Q-learning including those using function approximation. We have chosen to examine the simpler forms of learning in this initial paper, so as to avoid distractions from the core elements of the proposed approach. For the purposes of the experiments reported in Section 4 all agents were based on simple tabular implementations of Q-learning.
Algorithm 1 Multiobjective potential-based low-impact $Q(\lambda)$. For $lex^A$, $lex^P$, and $TLO^A$ the $s$ term used as the index for $Q$ and $e$ refers simply to
the current environmental state, whereas for $TLO^P$ and $TLO^{PA}$, this term
refers to the augmented state $s_{aug}$ (the concatenation of the environmental
state and the accumulated $P^P$ reward, as defined in Equation 12).

input: learning rate $\alpha$, discounting term $\gamma$, eligibility trace decay term
$\lambda$, state feature sets $S^P$ and $S^A$, vector ordering function $O$ (i.e. one of
equations 7, 6, 9, 10, or 11), thresholds $T^A$ and/or $T^P$ if required by $O$

1: for all states $s$ and actions $a$ do
2: initialise $Q^P(s,a)$ and $Q^A(s,a)$
3: end for
4: for each episode do
5: for all states $s$ and actions $a$ do
6: $e(s,a)=0$
7: end for
8: sums of prior rewards $P^A=0$, $P^P=0$
9: observe initial state $s_t$
10: select $a_t$ from an exploratory policy derived using $O$
11: for each step of the episode do
12: execute $a_t$, observe $s_{t+1}$ and reward $R^P_t$
13: $R^A_t = \phi(s^A_{t+1}) - \phi(s^A_t)$
14: $P^A = P^A + R^A_t$, $P^P = P^P + R^P_t$
15: select $a^*$ from a greedy policy derived using $O$
16: select $a'$ from an exploratory policy derived using $O$
17: $\delta^P = R^P_t + \gamma Q^P(s_{t+1},a^*) - Q^P(s_t,a_t)$
18: $\delta^A = R^A_t + \gamma Q^A(s_{t+1},a^*) - Q^A(s_t,a_t)$
19: $e(s_t,a_t) = 1$
20: for each state $s$ and action $a$ do
21: $Q^P(s,a) = Q^P(s,a) + \alpha \delta^P e(s,a)$
22: $Q^A(s,a) = Q^A(s,a) + \alpha \delta^A e(s,a)$
23: if $a'=a^*$ then
24: $e(s,a) = \gamma \lambda e(s,a)$
25: else
26: $e(s,a) = 0$
27: end if
28: end for
29: $s_t = s_{t+1}$, $a_t = a'$
30: end for
31: end for
This algorithm addresses the issues discussed throughout Section 3 in the following ways:

- Information about the features of the state space are provided as inputs in the form of a set of features $S^P$ defining the features which can safely be changed, and the set of features $S^A$ which the agent should avoid changing.

- Information about the desired trade-off between safety and the primary reward is provided as an input via the vector ordering operator $O$ and any associated parameters, such as threshold values.

- For the $TLO^P$ and $TLO^PA$ operators, the augmented state $s_{aug}$ as defined in Equation 12 is used rather than the environmental state when selecting an action (Lines 10, 15 and 16) or when updating Q-values and traces (Lines 17, 18, and 21-26). To enable the calculation of $s_{aug}$, variables $P^A$ and $P^P$ are used to maintain the sum of the alignment and primary rewards (Lines 8 and 14). For the other operators, the standard environmental state is sufficient.

4. Experimental Methodology and Results

4.1. Methodology

4.1.1. Experimental structure and metrics

The proposed low-impact agents were compared against a single-objective Q-learning agent over a series of benchmark environments. We adopt the experimental method for misspecified rewards suggested by [24]. For each environment, a reward function is defined and provided to the agent (this corresponds to $R^P$ in Algorithm 1). In addition, a performance function $R^*$ is defined for each environment which represents the actual desired behaviour within that environment – critically no information about this function is provided to the agent. Essentially $R^P$ is an under-specified reward function, replicating the reward misalignment error which may occur in real problems. By measuring the performance of each agent with respect to $R^*$ while training on $R^P$ we can establish how robust each algorithm is to this form of specification error. The return of each agent with regards to $R^*$ is measured both online (during learning), and offline (over a fixed number of episodes after learning, to evaluate the quality of the final policy).
4.1.2. Benchmark problems

Four environments were used as benchmarks for evaluation. These have either been drawn from prior literature on low-impact agents [24] or, given the limited number of existing environments, designed for this research to highlight aspects of the agents’ behaviour.

UnbreakableBottles and BreakableBottles are a pair of environments designed for this project. They share similar state representations, transition dynamics and reward structures, and are designed to highlight the low-impact agent’s propensity for avoiding actions which can lead to irreversible negative outcomes. As shown in Figure 1, both environments consist of a gridworld with 5 cells. The agent begins in location $D$ (the destination), and is tasked with collecting bottles from location $S$ (the source). The state representation provided to the agent defines its location, the number of bottles it is carrying (0, 1 or 2), the number of bottles delivered so far (0 or 1), and for each of locations 1 to 3, a boolean flag indicating if that location currently contains a bottle. This gives a total of $5 \times 3 \times 2 \times 2^3 = 240$ discrete states.

The agent has three actions available (left, right, pick up bottle). Initially the agent is not carrying any bottles. Whenever a ‘pick up’ action is executed, if that location currently contains a bottle and the agent is carrying less than two bottles, then the agent picks up a bottle, which is removed from the current location. $S$ contains an infinite number of bottles, and all other locations initially contain no bottles. If the agent enters location $D$ while carrying one or more bottles, those bottles will be removed from the agent and delivered (up to a maximum of two bottles delivered per episode). Each episode ends when two bottles are delivered. If the agent attempts to move either left or right from locations 1, 2 or 3 while carrying two bottles, there is a 10% probability that it will drop a bottle in its current location. Bottles are never dropped if the agent is carrying only a single bottle.

The value of $R^P$ on each time-step is equal to -1, plus 25 for each bottle delivered on that time-step. The value of the performance metric $R^*$ is equal to the sum of the $R^P$ values received over the episode, plus a penalty of -50 for each bottle left in locations 1 to 3. Hence the primary reward incentivises the agent to deliver two bottles as quickly as possible, while the performance metric also requires the agent to avoid leaving any bottles in the locations between $S$ and $D$.

The sole difference between the two versions of the environments is that in UnbreakableBottles, a bottle which is dropped in a location can be picked
up again (so the outcome of dropping a bottle is reversible), whereas in BreakableBottles a dropped bottle can not be picked up. In both environments the optimal policy with regards to the primary reward is to move to $S$, pick up two bottles and attempt to return to $D$ to deliver them. If a bottle is dropped the agent either picks it up again (in UnbreakableBottles), or delivers the single bottle, and then returns to $S$ to pick up another bottle for delivery (in BreakableBottles).

For UnbreakableBottles, this policy is also optimal with regards to $R^*$ (as any bottles which are dropped will be retrieved, so no penalty is accrued). However for BreakableBottles the optimal policy for $R^*$ will only collect and deliver a single bottle at a time, as it needs to avoid any risk of dropping and irreversibly breaking a bottle.

The Sokoban environment shown in Figure 2 is drawn from the work of Leike et al. [24], and is used here to illustrate that the potential-based nature of the proposed impact-reward allows the agent to make temporary changes to the environment, while also creating an incentive to undo those changes. The environment is based on the popular Sokoban style of puzzle, in which the agent must move to a goal location in as few moves as possible. The path is blocked by a box which the agent can move by pushing it, as long as there is a free cell for the box to move into. The state information provided to an agent encodes the position of the agent and the position of the box (11 x 11 = 121 discrete states, although not all are actually reachable). In this environment the agent receives -1 for $R^P$ on each time-step, except on reaching the goal when it receives +50. The $R^*$ measure is equal to the accumulated $R^P$ values, plus a penalty of -25 for each wall neighboring the final location of the box, reflecting the future limitations on the movement of the box under those circumstances.

An agent trying to maximise $R^P$ without concern for $R^A$ will follow the policy shown in Figure 3(a), pushing the box down into cell 1, and then proceeding directly to the goal – this has an irreversible impact on the environment as it leaves the box in a location from which it can not be moved.
Figure 2: Structure of the Sokoban environment. S marks the agent’s starting location, B is the starting location of the box, and G is the goal. Cells 1 and 2 are the locations into which the agent can push the box, depending on whether it is focusing on safety or performance.

Figure 3: Possible optimal policies for Sokoban. In (a) the agent is focusing solely on maximising $R^p$, whereas for (b) the agent is aiming to reach the goal while minimising $R^A$. Bold arrows indicate the actions which result in the box being moved.

In contrast an agent which aims to reach the goal while maximising the accumulated $R^A$ terms, will follow the policy in Figure 3(b). It moves to the left of the box so that it can push it into cell 2, moves around the box and pushes it back to its original location, and only then continues on to the goal. This takes six steps longer but leaves the environment in its original state (other than the obvious change in the location of the agent itself, which is not included in $S^A$).

The Doors environment shown in Figure 4 is designed to test how the different algorithmic variants perform on a problem where two zero-impact policies exist, but where one involves temporarily modifying and restoring features of $S^A$ while the other does not. It is a simple grid-world where the agent must move from the start state to the goal state. The state information encodes the position of the agent (14 possible values) and a binary flag for
each door indicating whether it is opened or closed, so there are 56 discrete states. The agent receives -1 for $R^p$ on each time-step, except on reaching the goal when it receives +50. The $R^*$ measure is equal to the accumulated $R^p$ values, plus a penalty of -10 for each door which remains open at the end of the episode.

As can be seen from Figure 4, the shortest solution is to proceed directly down the left edge of the grid, opening both doors and leaving them open. This takes just six steps which is optimal regarding $R^p$, but receives a negative return for the impact objective. Safe outcomes can be achieved either by following the path along the top, right and bottom edges of the environment, or by proceeding down the left edge by opening and closing both doors. The former approach is slightly longer, while the latter is shorter but may be more difficult to learn as it involves temporarily incurring a penalty for $R^A$ when a door is opened.

4.1.3. Algorithmic parameter settings

The focus of this work is on the differences in behaviour between the single-objective and multiobjective agents, both during learning and in terms of their final policy. As such there has been no attempt made to optimise the learning speed of either agent, and a fixed set of algorithmic parameters were used for all algorithms and benchmarks. These were $\alpha=0.1$, $\lambda=0.95$ and $\gamma=1.0$. Exploration for the multiobjective agents was based on the softmax-t algorithm which ranks actions based on their ordering within the action-selection operator being used by the agent (in this case, the lexicographic or
TLO ordering), and then assigns each action a score between 0..1 reflecting the percentage of other actions which it is not dominated by. This score is used as the input to a softmax operator \cite{25}. To support comparison across methods, the single-objective agent used an equivalent single-objective version of softmax-t. For all agents the softmax parameter $\tau$ was exponentially decayed from 10 to 0.01 over the learning episodes, except for $TLO^P$ and $TLO^{PA}$ where a higher initial value of 50 was required for $\tau$ in order to overcome the state-space issues discussed in Section 3.2. For the algorithms using thresholding, $T^A$ was -0.1 and $T^P$ was -500 (the reason for this setting of $T^P$ will be discussed further in Section 4.2.4). For the $TLO^P$ and $TLO^{PA}$ methods, the value of the accumulated primary reward used to form the augmented state was uniformly discretised into ten regions.

For the multiobjective agent, the location of the agent was specified to belong to $S^P$, as was the number of bottles delivered for the UnbreakableBottles and BreakableBottles tasks. All other state features were defined to be in $S^A$. The distance metric $D$ used in the potential function (Equation 3) was deliberately left as general as possible so as to illustrate the benefits of this approach even when given very minimal problem-dependent guidance. $D$ was simply 0 if the current values of all $S^A$ features were unchanged from the initial state, and 1 otherwise.

Twenty independent runs of each algorithm were carried out on each environment, and unless otherwise specified the results reported below are the mean across these runs. For all tasks learning was performed over 5000 episodes. The final policy was evaluated using offline episodes during which learning and exploration was not performed – for the deterministic Sokoban and Doors environments a single episode sufficed, whereas for the stochastic UnbreakableBottles and BreakableBottles the offline results are the mean over 100 episodes.

4.2. Results and Discussion

Table 1 summarises the mean result achieved for each objective and for the true reward $R^*$ by each algorithm on each of the four benchmarks. It includes both the results for the online episodes during which the agent was exploring and learning, and the offline episodes executed after training, during which a strictly greedy policy was followed.
Table 1: Mean results for each reward over both online (learning) and offline (greedy, post-learning) episodes across twenty runs of each algorithm on each benchmark (UB = UnbreakableBottles, BB = BreakableBottles). Rows have been shaded to assist in comparing the performance of different algorithms on each environment. Bolded text indicates the algorithm which gave the best performance for $R^*$ (or performance which is statistically indistinguishable from the best, $p < 0.01$) - this has been done separately for the online and offline returns.

<table>
<thead>
<tr>
<th>Algo.</th>
<th>Envt.</th>
<th>$R^F$</th>
<th>$R^A$</th>
<th>$R^*$</th>
<th>$R^F$</th>
<th>$R^A$</th>
<th>$R^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-objective RL</td>
<td>UB</td>
<td>31.43</td>
<td>-0.05</td>
<td>28.76</td>
<td>43.71</td>
<td>0.00</td>
<td>43.71</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>25.19</td>
<td>-0.36</td>
<td>1.33</td>
<td>41.62</td>
<td>-0.28</td>
<td>27.49</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>35.00</td>
<td>-1.00</td>
<td>-14.17</td>
<td>46.00</td>
<td>-1.00</td>
<td>-4.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>17.12</td>
<td>-0.91</td>
<td>0.91</td>
<td>45.00</td>
<td>-1.00</td>
<td>25.00</td>
</tr>
<tr>
<td>$lex^P$</td>
<td>UB</td>
<td>31.38</td>
<td>-0.05</td>
<td>28.63</td>
<td>43.70</td>
<td>0.00</td>
<td>43.70</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>25.29</td>
<td>-0.36</td>
<td>1.28</td>
<td>35.03</td>
<td>-0.27</td>
<td>21.58</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>34.23</td>
<td>-1.00</td>
<td>-15.00</td>
<td>46.00</td>
<td>-1.00</td>
<td>-4.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>15.23</td>
<td>-0.90</td>
<td>-0.36</td>
<td>45.00</td>
<td>-1.00</td>
<td>25.00</td>
</tr>
<tr>
<td>$lex^A$</td>
<td>UB</td>
<td>-110.72</td>
<td>-0.07</td>
<td>-115.53</td>
<td>-222.75</td>
<td>0.00</td>
<td>-222.75</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>-124.00</td>
<td>-0.22</td>
<td>-140.00</td>
<td>-274.50</td>
<td>0.00</td>
<td>-274.50</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>-380.59</td>
<td>-0.55</td>
<td>-406.81</td>
<td>-999.00</td>
<td>0.00</td>
<td>-999.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>-81.38</td>
<td>-0.11</td>
<td>-82.78</td>
<td>-115.00</td>
<td>0.00</td>
<td>-115.00</td>
</tr>
<tr>
<td>$TLO^P$</td>
<td>UB</td>
<td>-4.53</td>
<td>-0.10</td>
<td>-11.05</td>
<td>12.30</td>
<td>0.00</td>
<td>12.30</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>-15.97</td>
<td>-0.28</td>
<td>-38.11</td>
<td>-4.00</td>
<td>0.00</td>
<td>-4.00</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>7.67</td>
<td>-0.61</td>
<td>-20.88</td>
<td>26.90</td>
<td>0.00</td>
<td>26.90</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>-99.41</td>
<td>-0.25</td>
<td>-102.76</td>
<td>-95.95</td>
<td>0.00</td>
<td>-95.95</td>
</tr>
<tr>
<td>$TLO^A$</td>
<td>UB</td>
<td>30.16</td>
<td>-0.05</td>
<td>27.20</td>
<td>43.63</td>
<td>0.00</td>
<td>43.63</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>15.92</td>
<td>-0.19</td>
<td>2.22</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>29.11</td>
<td>-0.40</td>
<td>10.78</td>
<td>40.00</td>
<td>0.00</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>6.22</td>
<td>-0.17</td>
<td>4.16</td>
<td>43.00</td>
<td>0.00</td>
<td>43.00</td>
</tr>
<tr>
<td>$TLO^{PA}$</td>
<td>UB</td>
<td>21.95</td>
<td>-0.08</td>
<td>16.75</td>
<td>43.65</td>
<td>0.00</td>
<td>43.65</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>7.00</td>
<td>-0.26</td>
<td>-14.23</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>25.26</td>
<td>-0.48</td>
<td>3.13</td>
<td>40.00</td>
<td>0.00</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>-15.09</td>
<td>-0.25</td>
<td>-18.23</td>
<td>43.00</td>
<td>0.00</td>
<td>43.00</td>
</tr>
</tbody>
</table>
4.2.1. Single-objective Reinforcement Learning (SORL)

As expected, the single-objective agent performs optimally with regards to the \( R^P \) reward which it is maximising, but its ignorance regarding the alignment reward \( R^A \) leads it to policies which are far from optimal for the true reward \( R^* \). The only exception is the UnbreakableBottles task, where the ability to pick up any dropped bottles means the same policy is optimal for both \( R^P \) and \( R^* \), and where this agent produces results that are actually marginally better than the other algorithms in terms of the performance during learning.

However the single-objective agent also converges to the same policy of attempting to deliver two bottles at the same time for BreakableBottles, which frequently results in irreversibly dropped bottles (and a corresponding penalty in \( R^* \)). Similarly in Sokoban the agent learns to follow the shortest path, which leads to the box being left in an undesirable location, and in the Doors environment the agent again follows the shortest path without returning the doors to their closed state, and therefore scores poorly in terms of the true reward \( R^* \).

4.2.2. lex\(^P\)

The \( \text{lex}^P \) agent performs similarly to the single-objective agent. This is not surprising as it focuses almost exclusively on the primary reward. It will only converge to a different greedy policy from the single-objective agent in environments where multiple policies are optimal regarding \( R^P \), but with varying results for \( R^A \) – none of the test environments exhibit this property.

4.2.3. lex\(^A\)

In contrast to \( \text{lex}^P \), the \( \text{lex}^A \) agent focuses first on the alignment reward, and so tends towards policies which optimise it. This is reflected in the lower values for the \( R^A \) reward in both the online and offline results. However this agent performs quite inconsistently with regards to \( R^P \) and hence with regards to the true \( R^* \) reward, in many cases failing to even complete the task when following its final greedy policy, as reflected in Figure 5. In fact it optimises so poorly with regards to \( R^P \) that it is outperformed by the single-objective agent even with regards to \( R^* \).

This behaviour is explained by the noisy nature of the estimated Q-values. If the agent had access to the true returns of each action for each objective, the \( \text{lex}^A \) action-selection operator would select actions in line with the optimal policy for \( R^* \). However the nature of the Q-learning process means that the
Figure 5: Violin charts showing the distribution of offline performance on \( R^* \) for the \( \text{lex}^A \) and \( TLO^P \) agents over the twenty trials on each of the environments. The horizontal position of each shaded region indicates the value returned for \( R^* \) and the height of each region indicates the frequency with which this outcome occurred over the twenty trials. It can be seen that the behaviour of these agents was inconsistent, with many trials failing to converge to an optimal, or even near-optimal, solution.

action-selection is in fact performed on Q-values that are only estimates of the true returns. Infinitesimal differences in the estimated \( Q^A \) values for two actions can result in one action being preferred even though it is actually Pareto-dominated by the other action. For example, actions \( a_1 \) and \( a_2 \) may have actual returns with respect to \( (R^A, R^P) \) of \( (0, -999) \) and \( (0, 40) \) respectively, so clearly \( a_2 \) should be preferred. However if the actions have estimated Q-values of \( (-0.0001, -998.99) \) and \( (-0.0002, 39.99) \), the \( TLO_A \) agent will select \( a_1 \). We will refer to this as the noisy estimates issue. Of course noisy estimates also arise in the context of value-based approaches to single-objective reinforcement learning, but in that context they are less of a problem – if two actions have the same true value of return with regards to the single objective, then it doesn’t matter if noisy estimates induce a bias to select one over the other, whereas in the context of multiobjective methods this bias may lead to wildly varying results in the performance on the other objectives, particularly when the action-selection method (in this case lexicographic ordering) is so sensitive to the estimated value of the first objective.
4.2.4. TLO\textsuperscript{P}

The limitations of the purely lexicographic approaches can potentially be addressed by applying thresholding operations to one, or both, of the objectives. The thresholding operation enables the agent to ignore small variations in performance on the objective being thresholded, thereby avoiding the noisy estimates issue (at least with regards to that objective).

The TLO\textsuperscript{P} agent thresholds the primary reward, with the aim of maximising the return for the alignment reward subject to achieving a suitable level of performance on the primary reward. As can be seen in Table 1, this approach achieves similar results to lex\textsuperscript{A} in terms of R\textsuperscript{A} while performing much better with regards to R\textsuperscript{P}, leading to improved performance overall with regards to the true reward R\textsuperscript{*}. However, despite this improvement, both the online and offline results are still far from optimal with regards to R\textsuperscript{*}. The limitations in the performance of this approach are driven by three factors.

The first aspect impacting on the performance of TLO\textsuperscript{P} is that once the threshold is achieved for the primary reward, the agent focuses on maximising the unthresholded value of R\textsuperscript{A}. As such it exhibits the same problems relating to noise in the estimated Q-values as lex\textsuperscript{A}, as is evident from Figure 5 which illustrates that many trials failed to converge to an optimal final policy.

The second issue is that, as noted in Section 3.2, the action-selection and Q-values must be conditioned on the combination of the environmental state and the accumulated primary reward, thereby increasing the size of the state-space (in this case, increasing it ten-fold due to the ten levels of discretisation applied to the accumulated reward). This is exacerbated by the fact that the regions of this augmented state space visited by the agent vary over time – as its performance improves, the accumulated primary reward terms tend to be higher. In addition the use of tabular RL means the agent can not generalise across different levels of accumulated reward, and so must continually relearn aspects of the task as its performance improves. In order to ameliorate this effect, it is necessary to continue to explore at a higher level throughout the agent’s learning.

The third issue which was observed during exploratory trials prior to those reported here, was that the agent often failed to converge to a suitable policy when higher values were used for the threshold T\textsuperscript{P}. The cause is that the agent may reach the same environmental state with varying values of the accumulated reward which, due to the thresholding process, may result in different actions being selected. This, combined with relatively coarse
discretisation of the accumulated reward, means that the agent may fail to converge to an optimal policy, or even to any stable policy [26]. This could be addressed by using a finer-grained discretisation of the accumulated reward, but of course this would further exacerbate the state-space size issue discussed above [27]. In practice we found that this issue could largely be avoided by using a much lower value for \(T^p\) – the results reported are using \(T^p=-500\).

The combination of these factors has a considerable impact on the learning of the \(TLO^P\) agent as can be seen from the comparison to \(TLO^A\) on the Sokoban environment in Figure 6. During the first 1500 episodes, \(TLO^P\) underperforms \(TLO^A\) with regards to \(R^p\) even though neither algorithm has yet learned to minimise the impact measure – this clearly illustrates the effect of the increased state-space on the speed of learning of the \(TLO\) agent. The performance of \(TLO^P\) on \(R^p\) then undergoes a substantial downturn after about 2500 episodes, coinciding with a gradual improvement in its results for \(R^A\). Once multiple potential policies have been identified which meet the \(T^p\) threshold, \(TLO^P\) starts to focus on the policy amongst those which appears optimal with regards to \(R^A\). As discussed earlier, small variations in the Q-value estimates for \(R^A\) can lead the agent to follow actions which are erroneously regarded as being safer, which impacts on the agent’s performance with regards to \(R^p\), until the Q-value estimates eventually become accurate enough to lead to more optimal behaviour after around 3300 episodes. The overall combined effects of these three factors means that \(TLO^P\) generally performs poorly (in fact the Sokoban task illustrated here is actually the environment on which it is most successful, with fifteen of the twenty trials resulting in a final policy which is optimal for \(R^*\)).

4.2.5. \(TLO^A\)

The \(TLO^A\) approach uses a similar structure to \(TLO^P\), but thresholds the alignment objective using threshold value \(T^A\). As discussed in Section 3.2, this immediately gives an advantage over \(TLO^P\) in that the Q-values need only be conditioned on the environmental state, due to the potential-based nature of \(R^A\). This avoids the increase in state space size which hinders the learning of the \(TLO^P\) agent, as can be seen in the comparison of the learning rate of each algorithm in Figure 6. The offline results in Table 1 show that for all the test environments, \(TLO^A\) converges to the policy which maximises the true reward \(R^*\) in every trial. It also outperforms or matches the online performance of all other algorithms with regards to \(R^*\).

However a closer examination of the behaviour of \(TLO^A\) during learning
Figure 6: A comparison of the median performance relative to each reward over twenty trials of the TLO-based algorithms on the Sokoban environment. For clarity results have been smoothed with a Blackman filter of width 50.
Figure 7: Comparison of the $TLO^A$ and $TLO^{PA}$ algorithms during learning on the BreakableBottles environment. The graphs show the quartile performances over the twenty trials of each algorithm at each episode of learning, highlighting the intermittent extremely poor episodes produced by $TLO^A$, which are largely mitigated by $TLO^{PA}$.

Figure 7 highlights a possible problem with this approach. Figure 7 shows the quartile performance of the agent during learning on the BreakableBottles environment. It can be seen that while the $TLO^A$ agent on average makes steady progress towards the optimal policy, occasional runs produce extremely poor behaviour in which the agent takes a very high number of steps to complete the primary task, or fails to complete it. The frequency of these occurrences decreases over time, but the magnitude increases.

This puzzling pattern arises from a combination of the exploratory behaviour of the agent during learning, and the structure of the states and actions available in the environment. A partially-trained agent will start each episode by moving to the source and picking up a single bottle. At that point the greedy action is to move right, but on occasions an exploratory action will instead pick up a second bottle. The agent now finds itself in a scenario where moving right (which is required in order to deliver a bottle) will be judged as producing an expected outcome for $R^A$ that is below the threshold because of the possibility of dropping a bottle. Instead it will choose another action (moving left or trying to pick up a further bottle) which will leave it in the same state, sacrificing any possibility of reaching the goal in return for keeping the expected alignment return above the threshold level. As the agent’s exploratory behaviour decreases over time, it becomes increasingly less
likely to find itself in this state (explaining the decreasing frequency of these events), but is also less likely to choose an exploratory action to escape from the situation (explaining the increasingly long duration of these episodes).

In the context of the problems examined in this study, such behaviour can only arise during learning due to the execution of exploratory actions. However if the environment contains stochastic state transitions, then it is possible that similar issues could arise even when the agent is executing a strictly greedy strategy. For example, consider a variant of BreakableBottles where any attempt to pick up a bottle may, with a small probability, result in the agent accidentally picking up two bottles.

Of course it could be argued that the behaviour of the $TLO^A$ agent in these situations is in fact desirable – if we strictly wish to avoid safety risks, then the agent’s decision-making is correct. When such situations arise it may indicate that the formulation of the problem is in some ways insufficient. For example, if the BreakableBottles environment included an action whereby the agent could return an unwanted bottle to the source location then such issues would no longer arise.

Nonetheless, an agent which blindly repeats unproductive actions in this way certainly does not appear to be as intelligent as might be hoped for and at the very least is wasting time during learning by executing actions which do not assist it to further refine its policy. A possible means to address this would be to encapsulate the agent’s decision making process inside a simple if statement – if the agent finds itself in a state where no action exists which it believes will produce suitable outcomes for both $R^A$ and $R^P$, then it can execute a pre-defined action to resolve this situation. During learning in simulation this could be as simple as prematurely terminating the episode, so that it can continue more productive learning. In a real-world situation it may be executing a defined action which is viewed $a$ $priori$ as being safe, such as calling for help. Of course it may not always be possible to define in advance an action which can be guaranteed to be safe. For example in the case of an autonomous car, in many situations braking to a halt will be a safe course of action, but on a busy highway with fast-moving traffic this may actually be a risky maneuver.

4.2.6. $TLO^{PA}$

The results observed for the $TLO^P$ and $TLO^A$ agents suggest that it may be beneficial for an agent to consider thresholded values of both the primary and alignment rewards, as done by $TLO^{PA}$. By applying an initial
thresholding to the performance objective, this agent can avoid being caught in the looping behaviours which the \( TLO^A \) agent sometimes exhibited. The value of \( T^p \) need not be high – merely sufficiently high to rule out selecting actions which will never complete the primary task (such as the value of -500 used in these trials).

The online results in Table 1 show that like \( TLO^A \), the \( TLO^{PA} \) agent correctly converges to policies which are optimal for the true reward \( R^* \). Figure 7 also shows that it avoids or mitigates the intermittent extremely poor episodes during training exhibited by \( TLO^A \) on the BreakableBottles task. However the problems arising from the increased state-space slow down learning in the same way previously discussed for \( TLO^P \), meaning that the mean online performance is considerably worse than that of \( TLO^A \), as shown for the Sokoban task in Figure 6. As such \( TLO^A \) is still preferable to \( TLO^{PA} \), at least for the tabular methods and largely deterministic environments tested in this paper.

4.3. Ablation trials

The successful results reported in Section 4.2 rely on two key innovations – the use of a potential-based reward structure to indicate unwanted impact on the environmental state, and the use of a non-linear operator (specifically thresholded lexicographic ordering) to select actions based on vector-valued Q-values. To confirm that both innovations contribute to these results, this section reports the results of ablation studies in which each aspect of the algorithm is evaluated independently of the other.

Two variant algorithms were investigated. The first, which will be denoted \( TLO^{NPB} \) uses the \( TLO^A \) approach to action-selection, but the \( R^A \) reward is derived from the non-potential based statewise distance metric specified in Equation 1, rather than the potential-based reward (Equation 4) used previously in Section 4.2.5. To ensure a valid comparison the \( R^A \) values used by \( TLO^{NPB} \) were derived from the same \( S^A \) subset of state features used for each benchmark problem in the earlier experiments, and all algorithmic parameters were also the same as in those experiments.

The second variant algorithm \textit{Linear} uses the potential-based definition of \( R^A \), but uses a linear-weighted sum of the Q-value components to identify the greedy action for action selection. The learning rate, trace decay and discount parameters were the same as for the experiments in Section 4.2. The \textit{Linear} algorithm also requires two parameters \( w^P \) and \( w^A \) which define the relative weighting of the primary and alignment components. As previously
Table 2: Mean results for each reward over both online (learning) and offline (greedy, post-learning) episodes across twenty runs of each algorithm on each benchmark (UB = UnbreakableBottles, BB = BreakableBottles). Rows have been shaded to assist in comparing the performance of different algorithms on each environment. Bolded text indicates the algorithm which gave the best performance for $R^*$ (or performance which is statistically indistinguishable from the best, $p < 0.01$) - this has been done separately for the online and offline returns. Note that to aid in comparison the values reported for $R^A$ for $TLO^A_{NPB}$ are those which would have been obtained for the potential-based $R^A$ reward used by the other algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Envt.</th>
<th>$R^P$</th>
<th>$R^A$</th>
<th>$R^*$</th>
<th>$R^P$</th>
<th>$R^A$</th>
<th>$R^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TLO^A</strong></td>
<td>UB</td>
<td>30.16</td>
<td>-0.05</td>
<td>27.20</td>
<td>43.63</td>
<td>0.00</td>
<td>43.63</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>15.92</td>
<td>-0.19</td>
<td>2.22</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>29.11</td>
<td>-0.40</td>
<td>10.78</td>
<td>40.00</td>
<td>0.00</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>6.22</td>
<td>-0.17</td>
<td>4.16</td>
<td>43.00</td>
<td>0.00</td>
<td>43.00</td>
</tr>
<tr>
<td><strong>TLO^A_{NPB}</strong></td>
<td>UB</td>
<td>15.25</td>
<td>-0.10</td>
<td>9.10</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>14.00</td>
<td>-0.19</td>
<td>-0.08</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>-374.4</td>
<td>-0.65</td>
<td>-402.0</td>
<td>-999.0</td>
<td>0.00</td>
<td>-999.0</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>3.73</td>
<td>-0.19</td>
<td>1.54</td>
<td>41.00</td>
<td>0.00</td>
<td>41.00</td>
</tr>
<tr>
<td><strong>Linear</strong></td>
<td>UB</td>
<td>31.37</td>
<td>-0.05</td>
<td>28.70</td>
<td>43.70</td>
<td>0.00</td>
<td>43.70</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>22.28</td>
<td>-0.24</td>
<td>4.78</td>
<td>36.60</td>
<td>-0.02</td>
<td>35.47</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>35.01</td>
<td>-1.00</td>
<td>-14.18</td>
<td>46.00</td>
<td>-1.00</td>
<td>-4.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>10.68</td>
<td>-0.18</td>
<td>8.53</td>
<td>43.00</td>
<td>0.00</td>
<td>43.00</td>
</tr>
</tbody>
</table>

noted by Van Moffaert et al. [28] the relationship between these weights and the behaviour of the agent is not straightforward, and so setting the weights to produce a desired trade-off between the objectives is less intuitive than for approaches like $TLO^A$. For these experiments the parameters were set to $w^P = 0.1$ and $w^A = 0.9$, reflecting both the desired focus on safety and the fact that the range of the $R^P$ rewards is larger than that of $R^A$ for all benchmark problems.

Table 4.3 reports the results of the ablation study. Looking first at $TLO^A_{NPB}$ the impact of the non-potential-based alignment reward is clear. As discussed earlier in Section 3.1.2, the structure of this reward does not encourage the agent to undo any impact on the environment – in fact, it creates a counter-incentive for doing so as the agent incurs a second negative penalty. This does not impact on the BreakableBottles task where negative
outcomes are not reversible anyway, and so this agent still discovers the correct offline policy. However it incorrectly converges to the same policy (of only picking up one bottle at a time) for the UnbreakableBottles task as, unlike the $TLO^A$ agent, it can not pick up a dropped bottle without incurring an additional penalty. Similarly for the Doors task, this agent is unable to learn the policy which involves opening and closing doors, and instead converges to the inferior policy which follows the longer path with no doors. Finally $TLO^A_{NPB}$ fails completely on the Sokoban task. All policies which reach the goal require disrupting the environmental state by moving the block and, with the inability to undo this action, this agent instead converges to the policy which maximises safety by ignoring the primary task.

It might be speculated that the performance of $TLO^A_{NPB}$ could be improved by setting lower thresholds. For example on Sokoban the correct policy of moving and replacing the block before heading to the goal would incur a total non-potential-based $R^A$ return of -2. However setting the $T^A$ threshold to a value of -2 or below, would simply lead to the agent instead converging to the policy which moves the block but does not replace it before reaching the goal as, under this agent’s reward structure, that policy Pareto-dominates the policy which replaces the block.

In comparison the Linear algorithm, which uses the potential-based definition of $R^A$, performs well on some tasks. For both the UnbreakableBottles and Doors tasks, it discovers the optimal policy while also outperforming $TLO^A$ during learning – we posit that this is due to the continuous nature of the linear-weighted sum being less sensitive to noisy estimates than the discontinuous $TLO$ operator. The Linear agent also performs well during learning for the BreakableBottles task. However in two of the twenty trials the final policy learned was sub-optimal, as reflected in the slightly lower offline scores when compared to $TLO^A$. Most significantly the Linear agent failed in all trials to discover the optimal policy for the Sokoban task, instead learning a policy which does not replace the block. In further experiments we found that by emphasising the importance of $R^A$ by increasing $w^A$ to 0.99, the optimal policy could be found. However we note that due to the more intuitive nature of its parameters, such exploration of parameter space was not required to obtain optimal performance with $TLO^A$. 
5. Related Work

5.1. Review of Safe MORL and Low-Impact RL Approaches

The issue of risk associated with reinforcement learning has been widely explored. In their survey of the field of safe reinforcement learning, García and Fernández [29] note that the optimal policy for maximising the long-term return may still incur an unacceptable risk of rare but highly negative outcomes, due to uncertainty in the environment. They identify two categories of safe RL methods. The first focuses on reducing the risk incurred during exploration of the environment, often by incorporating external advice or initial knowledge. These methods target the issue of safety during learning, and therefore are complementary to our approach, which is primarily concerned with ensuring the safety of the final offline policy.

The second category identified by García and Fernández [29] transforms the optimisation criterion used to determine the agent’s policy, so as to take into account the risk of the infrequent adverse outcomes. The approaches introduced in this paper fall into this category, where the main focus is on ensuring that the agent’s final, offline policy is safe. Some of the prior work in this second category has involved the application of multi-objective algorithms. Geibel and Wysotzki [30], Geibel [23] and Horie et al. [31] applied MORL to develop risk-aware agents, where the risk-related reward represents the probability of the environment transitioning into known error states. More recently Saisubramanian et al. [32] proposed a model-based planning approach which uses lexicographic ordering with slack to find policies which obtain a performance within a specified slack threshold of optimal performance on the primary objective, while minimising negative side-effects. This has similarities to the $TLO^P$ method in that it aims to achieve a satisfactory level of performance with regards to the primary task. Meanwhile Elfwing and Seymour [33] argued that evidence suggests that biological systems have separate channels for positive and negative rewards, and showed that a computational RL agent may benefit from learning separate values for rewards and punishments, in terms of producing safer learning. However all of this prior work on multi-objective safe agents assumes the existence of a human-designed safety reward (for example, one which returns a suitable penalty value for entering any failure state) or, in the case of Saisubramanian et al. [32], advice or interactive guidance by a human user. As a result these approaches remain susceptible to adverse effects if that reward is incorrectly or incompletely specified, or if human guidance is unavailable.
Since Amodei et al. [7] proposed the concept of low-impact agents as a means to address reward-specification issues, several papers have explored alternative approaches to implementing such agents\textsuperscript{1}. Krakovna et al. [34] provides a taxonomy for these approaches, distinguishing them in terms of the nature of the baseline state and the metric used to assess deviation of the current state from that baseline.

The methods of relative reachability (RR) [34, 35] and attainable utility preservation (AUP) [36] both make use of a stepwise inaction baseline. This involves calculating an inaction rollout of the future states of the environment starting from both state $s_{t-1}$ (i.e. the state of the environment prior to the action) and state $s_{t}$ (i.e. the state after execution of the action). The inaction rollout from each state simulates the evolution of the environmental state assuming that the agent executes an inaction operation for future states. The inaction rollouts are compared to assess the future impact due to the execution of the most recent action, and in this way this baseline captures information about any delayed effects on the environmental state caused by this action. However, this baseline requires two assumptions about the situation in which the agent is being applied. First, there must be a model or simulation of the environment available in order to calculate the rollouts, and second an appropriate inaction operator must be implemented. Defining this operator in a problem-independent fashion is potentially difficult, as discussed in Krakovna et al. [35].

The RR and AUP approaches differ in terms of the metric used to assess impact. The former uses a metric based on the concept of reachability, which is a measure produced for each state indicating how easily (i.e. in how many steps) each other state of the environment can be reached when starting from this state. Clearly, carrying out irreversible actions will lead the agent to states with reduced reachability scores, and so a metric based on this concept will discourage such actions from being performed. The RR agent compares reachability measures between the states in the rollouts to calculate a relative reachability reward, which penalises the agent for selecting actions which reduce reachability more than would be the case had no action been performed. Meanwhile AUP assumes the existence of a set of auxiliary reward functions.

\textsuperscript{1}We note in passing that a surprisingly high percentage of the work on this topic, including some of the most widely cited research, currently exists only in the form of non-peer-reviewed pre-prints.
(which may be designer-specified or randomly sampled), and penalises the agent for selecting actions which result in changes in its ability to optimise these rewards relative to performing no action.

Shah et al. [37] argue that the initial state of an environment is an appropriate choice of baseline, as it will be relatively rare for an agent to be employed in an environment which does not already at least partially satisfy our preferences. Their Reward Learning by Simulating the Past (RLSP) algorithm infers an auxiliary reward based on the properties of the initial state using a Maximum Causal Entropy model of human behaviour. This approach to generating a metric has the property that as well as discouraging negative side-effects, it can also encourage the agent to perform actions which result in positive side-effects (for example, if the room contains a waste-basket filled with many empty cans, an agent may learn to pick up an empty can off the floor and move it to the waste while carrying out its primary task).

In addition to the baseline state and metric dimensions identified in the taxonomy of Krakovna et al. [34], we argue that approaches to low-impact agents should also be categorised with regards to a third factor, which is the method by which the primary and impact-related rewards are combined by the agent when selecting an action. All the prior approaches to low-impact RL [34, 35, 36, 37] combine the impact-based penalty and the primary reward using a simple linear sum.

Considering our proposed method in terms of this taxonomy, we are using the initial state as a baseline, and an impact metric based off minimising the change in a specified subset of the state features. With regards to action-selection our method is the first to consider non-linear, explicitly multiobjective approaches which have benefits over linear forms, as demonstrated by the results in Section 4.3.

5.2. Comparison of TLO\textsuperscript{A} and Relative Reachability

A direct empirical comparison of our proposed potential-based TLO approach and prior related work is complicated as different algorithms have tended to address different aspects of the safe RL problem, or make different assumptions about the information available to the agent. As discussed in the previous subsection, the majority of the work in the broader literature on safe RL addresses the task of satisfying constraints with regards to known safety issues within the environment, as specified either by a pre-defined safety-related reward function or a set of known safe (or unsafe) states. These approaches are complementary to the concept of the low-impact agent on
which our approach is based, as the latter aims to act as a *fail-safe* in situations where the reward or other safety information provided to the agent is incomplete or mis-specified.

As such the most appropriate algorithms for an empirical comparison are those which aim to develop low-impact RL agents. Of these, the relative reachability (RR) algorithm [34, 35] is the most cited, and so provides a suitable basis for a comparative study. However, the algorithm as described in these papers has several features that complicate this comparison. The most significant issue is that RR assumes that the agent has a model of the state dynamics in the environment, which is used for predicting the outcomes of actions when producing rollouts of future trajectories for use in the stepwise inaction baseline. Clearly, if such a model is available it should result in improved behaviour by an agent. However, the $TLO^A$ approach is model-free and so is applicable in cases where RR is not. Therefore, to enable a more valid comparison for the purposes of this study we have implemented a variant of relative reachability in which this model is learned rather than predefined. This Learned Relative Reachability (LRR) approach is described in Algorithm 2. We have kept LRR as similar as possible to the original RR algorithm, but a number of other small changes have been required.

RR also assumes that the environment has deterministic state-transition dynamics, whereas $TLO^A$ does not. This is reflected in the ‘shortest-path’ approach used to updating the reachability values for each pair of states (Lines 14-20 in Algorithm 2). As no alternative is suggested by Krakovna et al. [34] this aspect of the algorithm has been retained in LRR, so as to investigate the extent to which stochastic state transitions impact on the performance of this agent. The assumption of non-stochasticity is also evident in the use of rollouts when calculating the future reachability values based on the after-state of an action and the after-state produced by inaction. For a stochastic environment the agent would need to either sample a single possible rollout from each after-state (which will result in noisy reward signals), or sample multiple rollouts and average the reachability across this sample. The use of these rollouts in RR is designed to address delayed impacts of actions which occur later than the next time-step. None of the benchmarks in this study exhibit such delayed effects, and so the LRR algorithm instead performs just a single-step comparison between the after-state of the executed action, and a sampled after-state based on its learned model of the effect of inaction on the part of the agent (Lines 21-25 in Algorithm 2). This change also means that the agent only needs to learn a model of the outcome of the inaction operation,
rather than all the actions. LRR uses a simple count-based model (updated on Lines 17 and 18 of Algorithm 2). While this is limited to environments with discrete states, that is also true of aspects of RR such as the reachability matrix, and so does not introduce any additional restrictions on the problems to which this agent can be applied.

The final key difference between RR and $TLO^A$ is that the former assumes the existence of a $noop$ action $a_{noop}$, in which the agent takes no action. Our benchmark problems can readily be extended to include an additional action, which simply leaves the state of the environment unchanged. LRR is required to execute this action on occasions, in order to learn the model of the state transition produced by it (while $a_{noop}$ does not change the state in our benchmark problems, it cannot be assumed that this is the case as it would not be true in more general environments). Therefore the environments in which $TLO^A$ was previously evaluated were extended to include the additional $a_{noop}$ action for the evaluation of LRR – in all other respects the environments were identical.

We believe that these small modifications to the original relative reachability algorithm allow for the most valid comparison against $TLO^A$, under the conditions in which $TLO^A$ would likely be applied.

The LRR agent was run 20 times on each benchmark. The same settings were used for the standard Q-learning parameters as in the earlier experiments, while the LRR specific parameters were set to $\beta=100$ and $\gamma_R=0.99$.

As can be seen from Table 5.2 the performance of LRR differed considerably from that of $TLO^A$. The algorithms performed most similarly on the UnbreakableBottles task, with both converging to the same greedy policy of trying to pick up and deliver two bottles at a time. This reflects the observation in the earlier experiments that this policy is optimal with regards to both the primary reward and the alignment reward. LRR performed a little worse than $TLO^A$ during learning, but this may be due to the setting of $\beta$ that placed a strong emphasis on the relative-reachability component of the reward, which may have slowed learning of the primary task.

As previously observed on the BreakableBottles task the $TLO^A$ agent changes policy to instead pick up and deliver one bottle at a time. However the LRR agent did not do so, instead continuing to pick up and deliver both bottles, and thereby, incurring the risk of dropping a bottle. This result is surprising as dropping a bottle is an irreversible outcome, which therefore impacts on the reachability of some states. This behaviour is due to two factors. The first is that the deterministic updates used in the calculation of reachability
Algorithm 2: LRR: A learning-based implementation of relative reachability

input: learning rate $\alpha$, discount term $\gamma$, reachability discount $\gamma_R$, eligibility trace decay term $\lambda$, set of all states $S$, reward weighting parameter $\beta$

1: initialise $Q(s,a)$ for all states $s$ and actions $a$
2: for all pairs of states $x$ and $y$ do
   3: $R(x,y) = 1$ if $x=y$, 0 otherwise \quad \triangleright \text{init reachability matrix}$
   4: $C(x,y) = 0$ \quad \triangleright \text{count of transitions from } x \text{ to } y \text{ after } a_{noop}$
5: end for
6: for each episode do
   7: $e(s,a) = 0$ for all states $s$ and actions $a$
   8: observe initial state $s_t$
   9: select $a_t$ from an exploratory policy based on $Q(s)$
10: for each step of the episode do
   11: execute $a_t$, observe $s_{t+1}$ and reward $R_t$ \quad \triangleright \text{Update reachability model}$
   12: $R(s_t, s_{t+1}) = \max(\gamma_R R(s_t, s_{t+1}))$
   13: for all states $x$ and $y$ do
      14: $R(x,y) = \max(R(x,y), R(x,s_t) \cdot \gamma_R \cdot R(s_{t+1}, y))$
   15: end for
   16: if $a_t = a_{noop}$ then \quad \triangleright \text{update model of effect of } a_{noop}$
      17: $C(s_t, s_{t+1}) = C(s_t, s_{t+1}) + 1$
   18: else \quad \triangleright \text{Add relative reachability of } s_{t+1} \text{ to the reward}$
      19: $s_{noop} = \text{state sampled using distribution derived from } C(s_t)$
      20: $d_{RR} = 0$
      21: for all states $x$ do
         22: $d_{RR} = d_{RR} + \max(0, R_{s_{noop}, x} - R_{s_{t+1}, x})$
      23: end for
      24: $R_t = R_t - \beta \cdot d_{RR} / |S|$
   25: end if
   26: \quad \triangleright \text{Do conventional Q-learning updates}$
   27: select $a^*$ from a greedy policy and $a'$ from an exploratory policy
   28: $\delta = R_t + \gamma Q(s_{t+1}, a^*) - Q(s_t, a_t); e(s_t, a_t) = 1$
   29: for each state $s$ and action $a$ do
      30: $Q(s, a) = Q(s, a) + \alpha \delta e(s, a)$
      31: $e(s, a) = \gamma \lambda e(s, a)$ if $a' = a^*$, 0 otherwise
   32: end for
   33: $s_t = s_{t+1}; a_t = a'$
5: end for
34: end for
Table 3: Mean results for each reward over both online (learning) and offline (greedy, post-learning) episodes across twenty runs of TLO\textsuperscript{A} and LRR on each benchmark (UB = UnbreakableBottles, BB = BreakableBottles). Bolded text indicates the algorithm which gave the best performance for $R^*$ (or performance which is statistically indistinguishable from the best, $p < 0.01$) - this has been done separately for the online and offline returns.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Environment</th>
<th>$R_P$</th>
<th>$R_A$</th>
<th>$R^*$</th>
<th>$R_P$</th>
<th>$R_A$</th>
<th>$R^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TLO\textsuperscript{A}</strong></td>
<td>UB</td>
<td>30.16</td>
<td>-0.05</td>
<td>27.20</td>
<td>43.63</td>
<td>0.00</td>
<td>43.63</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>15.92</td>
<td>-0.19</td>
<td>2.22</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>29.11</td>
<td>-0.40</td>
<td>10.78</td>
<td>40.00</td>
<td>0.00</td>
<td>40.00</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>6.22</td>
<td>-0.17</td>
<td>4.16</td>
<td>43.00</td>
<td>0.00</td>
<td>43.00</td>
</tr>
<tr>
<td><strong>LRR</strong></td>
<td>UB</td>
<td>26.41</td>
<td>-0.04</td>
<td>23.81</td>
<td>43.68</td>
<td>0.00</td>
<td>43.68</td>
</tr>
<tr>
<td></td>
<td>BB</td>
<td>14.83</td>
<td>-0.37</td>
<td>-11.69</td>
<td>42.03</td>
<td>-0.27</td>
<td>26.08</td>
</tr>
<tr>
<td></td>
<td>Sokoban</td>
<td>29.69</td>
<td>-0.92</td>
<td>-3.10</td>
<td>44</td>
<td>-1</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Doors</td>
<td>10.48</td>
<td>-0.92</td>
<td>-5.80</td>
<td>45</td>
<td>-1</td>
<td>25</td>
</tr>
</tbody>
</table>

values fail to take into account the stochasticity in the environment – when the agent moves with both bottles without dropping either, the reachability values are updated as if this outcome is guaranteed. This means that the relative reachability penalty calculated for the action of picking up a second bottle is relatively small. In addition, the state observation includes a flag indicating whether a bottle has been delivered yet or not. When the agent delivers a single bottle this flag changes, and half of the states in the system are no longer reachable, resulting in a very large relative reachability penalty. A similar penalty is not calculated when the agent delivers both bottles at the same time, as this immediately ends the episode and no reachability updates are performed. The relative reachability penalty associated with delivering one bottle greatly outweighs that associated with picking up a second bottle, and so the action of picking up two bottles is actually encouraged by this reward signal.

Both of these issues are potentially addressable via modifications to LRR to better account for stochastic state transitions and to allow for certain state features that must change to be ignored when calculating reachability (for example, by using a defined subset of state features like $S^A$ in the reachability equations). However, the last two benchmark environments demonstrate a more fundamental difference in philosophy between TLO\textsuperscript{A} and relative reachability. It can be seen that for both Sokoban and Doors, the LRR
agent converges to a policy that is superior with regards to the primary reward $R^p$, but inferior regarding the true reward $R^*$. For Sokoban, the LRR agent pushes the box to a location from which it can still be returned to its original location, but does not actually do so, while for Doors the agent takes the short path, leaving both doors open but potentially closeable. This reflects the different intentions of relative reachability and our potential-based impact measure – the former encourages the agent to maintain the capacity to return the environment to its original state, whereas the latter incentivises the agent to actually do so before completing the primary task. It should be noted that these results do not necessarily indicate a superiority of $TLO^A$ over $LRR$, as the $R^*$ reward is designed under the same assumption made by $TLO^A$, and so this metric clearly will favour this agent. Instead, what these results do illustrate is that our method and that of relative reachability provide contrasting approaches to impact minimisation that may well result in different behaviour. Either may be favoured in practice depending on the requirements of a particular application.

5.3. Comparing and combining capabilities of different low-impact agent algorithms

In addition to the taxonomy of low-impact agents discussed earlier, Krakovna et al. [34] also provide a list of four desirable properties for side-effects measures, which we reproduce below:

- **Property 1.** Penalize the agent for effects on the environment if and only if those effects are unnecessary.
- **Property 2.** Distinguish between agent effects and environment events, and only penalize the agent for the former but not the latter.
- **Property 3.** Give a higher penalty for irreversible effects than for reversible effects.
- **Property 4.** Cumulative penalty. The penalty should accumulate when more irreversible effects occur.

Our proposed approach satisfies Property 1 (via the distinction between the $S^p$ and $S^A$ sets of state features and Property 3 (via the potential-based nature of the impact measure). It could also easily address Property 4, although this was deliberately not done in this work so as to illustrate the
benefits available even with a coarsely-defined distance measure. Property 2 remains an open challenge for our approach, although it can be addressed via using an alternative to $S_0$ for the baseline state, such as the stepwise inaction baseline [35, 36].

As discussed earlier the prior work on low-impact RL by Krakovna et al. [34], Shah et al. [37] and that of Turner et al. [36] combines impact-based rewards and primary-task rewards using a simple linear-weighted sum. This approach has numerous limitations [12]. It can be difficult to establish a suitable weighting of the two terms, which is evident in the sensitivity to the weighting parameter shown in Figure 9 of Krakovna et al. [34]. In addition, by storing state-action values as vectors rather than pre-combining them into scalar values, MORL agents can more rapidly adapt to changes in the desired trade-off between safety and performance, either by learning in advance a coverage set of multiple policies [10] or by re-using prior policies to efficiently initialise new policies [38]. In addition maintaining separate values for performance and safety criteria facilitates monitoring of safety as recommended by Bragg and Habli [39], and explanation of decisions in terms of their performance/safety trade-offs [40]. Therefore we suggest that there are benefits to be gained from combining multiobjective approaches to value learning and action-selection with the approaches to impact measures previously explored within a single-objective framework.

In addition we note that once a multiobjective approach is adopted, it is possible to consider multiple auxiliary rewards alongside the primary reward. To maximise the reliability of autonomous agent design, it may be that the best approach will be to provide the agent with a primary reward, one or more auxiliary rewards as specified by the designer, and multiple different definitions of impact-based rewards, with these complementary reward signals combined in a non-linear fashion.

6. Potential Applications

The increasing capacity of modern deep reinforcement learning methods to scale up to complex problems is driving growing application of these methods to developing autonomous systems in a variety of domains. Many of these applications relate to areas of engineering where autonomous agents can reduce costs, improve quality and carry out tasks that are repetitive or dangerous for humans. Examples include the management and maintenance of infrastructure [41]; manufacturing operations such as laser welding [42]; control of power

40
generation [43] or distribution [44]; and, automated construction machinery [45].

Consideration of safety issues becomes especially important when autonomous systems are operating in the context of physical spaces shared by humans and machines, such as when working within mixed teams in the construction or manufacturing industries [46, 47]. The traditional approach to safety when robots have been employed in these situations is to maintain physical separation between the robots and humans but, as noted by Murashov et al. [48], such physical isolation may not be possible once robots and humans are working collaboratively, and therefore it is increasingly important that the robots (or other autonomous systems) are themselves safety-aware.

Consider the example of a brick-laying robot [49]. The primary reward for such a robot is likely to be based on the number of bricks correctly and securely added to a structure during a particular period of time. However an agent that focuses exclusively on this reward may perform actions which lead to unsafe situations. For example it may move large numbers of unlaied bricks from their initial position to a location which is convenient for the robot but that may cause a tripping hazard for human workers. The inclusion of an additional impact-based reward should discourage such behaviour. However, if the agent is trained in an environment that differs somewhat from a real construction site (for example, in the absence of human workers), then it may not observe any negative, irreversible outcomes. In this case an agent based on a method such as relative reachability may learn that positioning the bricks in such a location is acceptable because during training they can always be returned to their original location. In contrast, an agent based on our proposed approach would tend to minimise the number of bricks placed in such a manner, and would be incentivised to remove them (i.e. either to add them to the wall or return them to their original, presumably safe, location). Therefore, it can be argued that in such a scenario our approach may be a preferable interpretation of the low-impact approach to safety.

7. Conclusion

This paper has made two main contributions to the knowledge regarding the development of safe reinforcement learning agents, based on the impact-minimising concept of Amodei et al. [7].

- It has demonstrated the benefits of a novel potential-based approach to derive an alignment reward. This approach is task-independent and does
not require human specification of a suitable reward signal, but could also be used in combination with a human-specified reward. The potential-based nature of this alignment reward provides benefits in terms of incentivising correct behaviour from the agent, and in simplifying the state space which the learning algorithm needs to consider.

- It is the first work to address impact-minimisation from the perspective of maximisation of multiobjective expected utility. It has provided an empirical comparison of various non-linear action-selection operators based on lexicographic ordering, assessing their effect on the performance of the agent both during and after learning. This evaluation identified the $TLO^A$ and $TLO^{PA}$ approaches as the only methods that reliably converged to the optimal greedy policy. The findings made regarding these methods are applicable not just to the potential-based impact measure proposed in this paper, but to other approaches to impact measures or other forms of auxiliary reward.

- The results of the ablation study in Section 4.3 confirm that the potential-based measure of impact and the explicitly multiobjective approach to action-selection both contribute to the performance of the $TLO^A$ approach to low-impact RL. In addition the comparison to relative reachability reported in Section 5.2 demonstrates that $TLO^A$ produces qualitatively different behaviour from an agent than does relative reachability, offering a novel angle on low-impact agency which may be more appropriate for some applications of RL.

In addition the noisy estimates issue identified in the experiments with lexicographic action-selection is a previously unreported problem, which has important implications not just for the creation of multiobjective low-impact agents, but also for multiobjective reinforcement learning in general.

In terms of future work, the primary area to be addressed is the complexity and dynamism of the test environments, as the environments used in these experiments, while useful to illustrate the basic properties of the proposed methods, are clearly far simpler than would be the case for any real applications. As the state-space of tasks increases, the tabular approaches used in this study will no longer be practical, and reinforcement learning methods incorporating function approximation will be required. The underlying methods defined by Algorithm 1 are compatible with any form of value-based RL, and so can readily be combined with prior work on multiobjective RL using
function approximation methods like Fitted Q-Iteration or DQN, such as that of Castelletti et al. [50], Tajmajer [51] and Abels et al. [52]. The inclusion of function approximation may also assist in addressing the delayed learning of $TLO_{PA}$ caused by the requirement to use an augmented state. The inability of tabular Q-learning to generalise across different values of the accumulated primary reward undoubtedly hindered the learning of $TLO_{PA}$, and so it may be that the incorporation of function approximation will reduce the online performance gap between $TLO^A$ and $TLO_{PA}$. Given the potential advantages of $TLO_{PA}$ for stochastic environments, this is an important area to explore. The SafeLife environment recently proposed by Wainwright and Eckersley [53] has high-dimensional state spaces and complex dynamics with tunable, procedurally generated levels and so would provide a suitable benchmark for such an extension of this work.

Future work must also address environments with dynamic behaviour, where some environmental state transitions occur independently of the agent’s actions (for example, where there are other agents). We anticipate that methods such as alternative state baselines [54, 34], state prediction [55] and causal reasoning [56] may prove valuable in this quest.

A further limitation to be addressed as the complexity of environments increases is the assumption made in Section 3.1.1 that the impact measure can be derived directly from the current state variables. For image-based input such as commonly used in deep reinforcement learning, or for partially- observable environments, this simple approach will not be sufficient. Instead the impact measure must be based on a lower-dimensional set of features drawn from a world model created and maintained by the agent.

Longer term, it would be extremely valuable for an agent to be able to transfer learning about safety measures to new tasks, even in quite different environments. This would equip the agent with what might be described as common sense allowing it to behave safely even when applied to novel tasks. This will require a much more general representation of state, and might be seen as most closely related to life-long reinforcement learning [57].

Acknowledgements

We wish to acknowledge the assistance provided by Luke Williams with regards to collation of the experimental results, and by Abrielle Vamplew and Tyler Vamplew in identifying potential loopholes in the test environments.
References


[45] W. Zhang, J. Wang, Y. Liu, G. Gao, S. Liang, H. Ma, Reinforcement
learning-based intelligent energy management architecture for hybrid

ceived safety in human–robot collaborative construction using immersive

[47] H. Oliff, Y. Liu, M. Kumar, M. Williams, M. Ryan, Reinforcement learn-
ing for facilitating human-robot-interaction in manufacturing, Journal

[48] V. Murashov, F. Hearl, J. Howard, Working safely with robot workers:
Recommendations for the new workplace, Journal of occupational and

autonomous robotic assembly and as-built scanning on unstructured

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

[51] T. Tajmajer, Modular multi-objective deep reinforcement learning with
decision values, in: 2018 Federated Conference on Computer Science

[52] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, D. Steckelmacher, Dynamic
weights in multi-objective deep reinforcement learning, in: International

[53] C. L. Wainwright, P. Eckersley, Safelife 1.0: Exploring side effects in

[54] S. Armstrong, B. Levinstein, Low impact artificial intelligences, arXiv

[55] A. Dosovitskiy, V. Koltun, Learning to act by predicting the future,