Mach Learn (2011) 84:51–80
DOI 10.1007/s10994-010-5232-5

Empirical evaluation methods for multiobjective reinforcement learning algorithms

Peter Vamplew · Richard Dazeley · Adam Berry · Rustam Issabekov · Evan Dekker

Received: 26 February 2010 / Revised: 24 November 2010 / Accepted: 3 December 2010 /
Published online: 22 December 2010

Abstract While a number of algorithms for multiobjective reinforcement learning have been proposed, and a small number of applications developed, there has been very little rigorous empirical evaluation of the performance and limitations of these algorithms. This paper proposes standard methods for such empirical evaluation, to act as a foundation for future comparative studies. Two classes of multiobjective reinforcement learning algorithms are identified, and appropriate evaluation metrics and methodologies are proposed for each class. A suite of benchmark problems with known Pareto fronts is described, and future extensions and implementations of this benchmark suite are discussed. The utility of the proposed evaluation methods are demonstrated via an empirical comparison of two example learning algorithms.

Keywords Multiobjective reinforcement learning · Multiple objectives · Empirical methods · Pareto fronts · Pareto optimal policies

1 Introduction

The observation that most real-world applications require the simultaneous satisfaction of multiple objectives drove the growth in multiobjective optimisation research during the 1990s (Coello et al. 2002). Recent years have seen the beginnings of corresponding research into extending reinforcement learning (RL) techniques to multiobjective problems. Algorithms for handling multiobjective tasks have been proposed for both dynamic programming (Wiering and de Jong 2007) and RL (for example, Gabor et al.

Editors: S. Whiteson and M. Littman.

P. Vamplew (✉) · R. Dazeley · R. Issabekov · E. Dekker
Graduate School of Information Technology and Mathematical Sciences, University of Ballarat,
P.O. Box 663, Ballarat, Victoria, 3353 Australia
e-mail: p.vamplew@ballarat.edu.au
url: http://www.ballarat.edu.au/ard/itms/staff/pvamplew.shtml

A. Berry
CSIRO Energy Centre, 10 Murray Dwyer Circuit, Steel River Estate, Mayfield West, New South Wales, 2304, Australia

1998; Mannor and Shimkin 2001, 2004; Shelton 2001; Natarajan and Tadepalli 2005; Barrett and Narayanan 2008). Multiobjective reinforcement learning (MORL) methods have been applied to tasks such as lake water level control (Castelletti et al. 2002) and balancing power consumption and performance in Web servers (Tesauro et al. 2007).

However the field of MORL is still in its infancy and as such the literature remains fragmented. In particular little work has been published comparing the performance of different algorithms. Generally the performance of each algorithm has been examined in isolation, with the algorithm tested on only one or two small problems. These experimental results exhibit two limitations. First, the small number of test problems may not fully test the algorithm's ability to cope with a range of different problem features. Second, there is no overlap between the test problems and methodologies used by different authors, making direct comparison of results impossible. In addition no consensus exists on appropriate performance metrics, further complicating such comparisons between different studies.

This paper aims to address the methodological limitations of current MORL research by proposing an approach for empirical evaluation, based on standardized metrics and benchmarks.

2 Overview of MORL

Before outlining our proposed approach, it is useful to define the characteristics of MORL problems and algorithms, to present some key concepts from multiobjective optimisation that form a foundation for our experimental methods, and to review previous approaches to empirical evaluation of MORL algorithms.

2.1 Definition of multiobjective reinforcement learning

MORL problems differ from conventional RL problems in having two or more objectives to be achieved by the agent, each with its own associated reward signal—so the reward is a vector rather than a scalar value. Specifically we consider problems where the objectives are in conflict—if all objectives are either directly related or completely independent, they can easily be combined into a single objective and a policy found that can maximize all of them. In contrast if the objectives are in conflict then any policy must either maximize only one objective, or represent a trade-off between the conflicting objectives.

It is important to note the distinction between the multiobjective tasks described in this paper, and the multigoal task addressed by other authors such as Crabbe (2001). In the latter there are specific goal states with a reward received only when the agent reaches the goal which is its specific target. In this case at most one reward will be non-zero at any point in time, whereas in the more general MORL task that we consider there are no such restrictions on the reward vector.

2.2 Pareto dominance and the Pareto front

The aim of any MORL algorithm is to identify policies that produce suitable compromises between the multiple objectives of the task. A 'good' compromise can be defined in terms of Pareto dominance (Pareto 1896), which allows comparison of a pair of solutions to a multiobjective problem as shown in Fig. 1.[1]

[1] In multiobjective optimisation, the task is generally to minimise each objective, so a lower objective value is superior to a higher value. In contrast in RL the task is to maximise the reward, and so the notion of superiority is reversed. Given the expected audience, this paper will be framed in terms of maximisation.

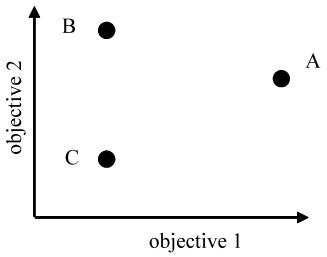


Fig. 1 Illustrating Pareto dominance in the context of maximizing objectives—Solution A strongly dominates solution C; Solution B weakly dominates solution C; A and B are incomparable

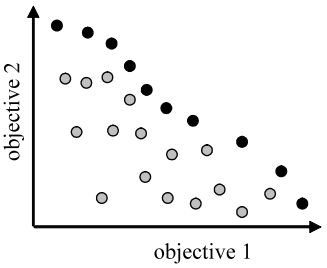


Fig. 2 The *black points* indicate solutions which form the Pareto front; all *grey* solutions are dominated by at least one member of the Pareto front

One solution (A) strongly Pareto dominates another (C) if it is superior on all objectives. A solution weakly dominates another if it is superior on at least one objective, and at least equal on all other objectives (B weakly dominates C). Finally two solutions (A and B) are incomparable if each is superior to the other on at least one objective. Any solution that is dominated by another is of little value, as clearly the dominating solution is preferable. Therefore the best solutions can be extracted from a set of solutions by retaining only those solutions that either dominate or are incomparable with every other member of the set. If this process is applied to the set of all possible solutions, the resulting set of non-dominated solutions is referred to as the Pareto front, and represents the optimal set of compromise solutions. Figure 2 illustrates this concept; in the context of MORL, each point in this figure corresponds to the performance achieved on each objective by a particular policy and hence there are multiple Pareto optimal policies.[2] Of course establishing the true front for any problem of significant size is generally impractical, and so the goal is to produce a set of solutions that approximates the Pareto front. A good approximate front should contain solutions that are accurate (close to the actual front) and evenly distributed along the front, with an extent similar to that of the actual front (Zitzler et al. 2003).

2.3 Empirical evaluation of multiobjective reinforcement learning

Empirical evaluation is a vital component of machine learning research, particularly in supporting the comparison of algorithms. Such comparisons can be carried out most reliably when standard empirical methodologies are followed, as this eliminates variation in results due to differing procedures. For example the use of standard datasets such as the UCI repository (Frank and Asuncion 2010) has greatly aided in the comparison of supervised learning approaches.

[2]For simplicity the majority of this paper will consider only the discrete set of solutions produced using deterministic policies. The use of stochastic policies will be addressed in Sect. 6.4.

Over the last five years there has been significant progress made in establishing standard empirical methods for the evaluation of single-objective RL algorithms. As noted by White (2006), the dynamic nature of the test environments used for RL makes the task of sharing benchmark problems more complex than for supervised learning; fully defining environmental dynamics within the space of a conference or journal paper is often not possible, and in any case can lead to errors in implementation of the environment. Rather than sharing data files as the UCI repository does, it is instead necessary to share implementations of test environments which in turn leads to issues of compatibility between programming languages and code frameworks. The development and public release of standard frameworks and implementations such as RL-Glue (Tanner and White 2009) and the UMass RL repository (UMass 2010) have greatly supported the adoption of standard problems. This process has been aided by the RL bake-offs which have been held in conjunction with major conferences in recent years (see for example Dutech et al. 2005). These competitions have promoted both the use of these standard frameworks and benchmarks, and also the adoption of consistent approaches to experimental structure and evaluation metrics.

In contrast there has been relatively little focus in the MORL literature on empirical testing, and very little consideration of the need for standard methods for evaluating, reporting and comparing the performance of algorithms. An insight into the state of empirical evaluation of MORL algorithms can be gained by examining the empirical approaches adopted in a collection of papers that propose new or modified multiobjective algorithms (Gabor et al. 1998; Shelton 2001; Mannor and Shimkin 2004; Natarajan and Tadepalli 2005; Geibel 2006; Barrett and Narayanan 2008; Handa 2009).

Most of these papers report results on only one or two test problems, which may fail to adequately represent the true general performance of the algorithm. For example a central aspect of the multi-criteria approach of Gabor et al. (1998) is the application of thresholds to the objectives, yet the tic-tac-toe problem reported in the paper does not utilize this feature. These concerns can be addressed by testing on a wider range of benchmark problems or by using generalized problems with parameters that can be varied to produce a class of related learning tasks, as suggested for single-objective RL by Whiteson et al. (2009). The only MORL work to have used generalized problems so far is that of Geibel (2006) and Handa (2009), although the latter reports results for only two variations.

In addition to the small number of problems considered per paper, another issue is the lack of overlap between the test problems used by these authors, making it impossible to compare results between different papers. This becomes even more important when we note that most papers either provide results for their proposed algorithm with no comparison to any other algorithm, or compare results only against an earlier variant of their algorithm. In addition for the majority of test problems, there is no knowledge of the actual Pareto optimal policies, so the absolute quality of the solutions reported in the papers cannot be judged.

A further issue complicating the comparison of results is the inconsistency in the manner in which algorithmic performance is reported. Results have been reported as graphical representations of fronts (Shelton 2001; Geibel 2006; Barrett and Narayanan 2008; Handa 2009), as trajectories through reward space (Mannor and Shimkin 2004), as per-objective reward over time (Gabor et al. 1998), and as average weighted reward over time (Natarajan and Tadepalli 2005). The one consistent feature has been that almost all authors have reported results in a graphical format, rather than providing numeric measurements. This tendency to graphical results complicates the task of comparing the performance of algorithms between papers, even if common test problems were to be used.

Some of the lack of consistency of empirical methods can be attributed to the varying assumptions and potential areas of application that have underpinned the design of MORL

algorithms, as will be discussed in Sect. 3. However over recent years research in single-objective RL has undoubtedly benefitted from the establishment of standard empirical methods and tools. In order for the field of MORL to move beyond its current status as a series of isolated studies, there is a clear need for the establishment of conventions and standards for empirical evaluation. In particular we would argue for the following:

- MORL algorithms should be tested on a wider range of test problems, and the properties of these problems should be understood to aid in interpreting the variations in the performance of algorithms.
- Standard benchmark problems and standard implementations of these problems should be established so as to facilitate comparison of the performance of different algorithms.
- Standard approaches to experimental methodology and reporting of results should be adopted, again to aid in the comparison of algorithms between papers. In particular numeric measures of performance will prove more useful for this purpose than the graphical reporting of results.

The remaining sections of this paper address these recommendations. Section 3 categorises MORL algorithms into two classes (multiple-policy and single-policy) and presents two example algorithms which will be referred to throughout the remainder of the paper. Sections 4 and 5 propose empirical evaluation metrics and methods for both classes of algorithms, Sect. 6 presents a suite of benchmark problems, and Sect. 7 demonstrates the utility of these proposed metrics and benchmarks via an empirical comparison of the two example algorithms.

3 MORL algorithms

3.1 Classes of multiobjective reinforcement learning algorithms

The extension of RL from a single objective to multiple objectives introduces new possibilities for variations in the aims of RL algorithms. While single-objective algorithms may vary in terms of their internal mechanisms, they all share the same aim of maximizing the reward received, which will be achieved by identifying a single optimal policy. As Sect. 2.2 indicated, for multiobjective tasks there is no longer a single optimal policy, as many policies may in fact be Pareto optimal. Therefore variations can exist between MORL algorithms in terms of the number and nature of the policies that they aim to discover.

Current MORL algorithms can be divided into two categories based on the number of policies that they learn. One class aims to learn the single policy that best satisfies a set of preferences between objectives as specified by a user or derived from the problem domain. We will refer to these as single-policy algorithms. The second class seeks to find a set of policies which approximate the Pareto front. We will refer to these as multiple-policy approaches.[3]

[3]It is possible to imagine MORL algorithms which blur the boundaries between these classes. For example, an algorithm may use on-policy methods to learn a single policy while simultaneously applying off-policy learning to learn the value of nearby policies along the Pareto front. However to our knowledge no concrete examples of such algorithms have previously been proposed or tested in the literature.

3.1.1 Single-policy algorithms

The majority of MORL algorithms proposed so far are of the single-policy nature; that is they aim to learn a single policy that is in some way 'optimal'. In this sense these algorithms are similar to single-objective algorithms. However in the single objective case, there is a unique optimal policy (or several optimal policies with equivalent performance) that is the target for the learning process. In the multiobjective case, many Pareto optimal policies may exist, and so the algorithm must be given guidance as to which of these policies is to be preferred. A fundamental difference between the single-policy algorithms proposed so far is the manner in which these preferences are expressed, which derives in large part from the nature of the problem for which each algorithm has been designed.

Gabor et al. (1998) provide the earliest example of a single-policy algorithm. Their approach is designed for problems where constraints apply to some of the objectives—for example a robot carrying out a navigation task while maintaining a battery level above zero. The nature of the desired policy is defined by specifying threshold values which specify the constraints on objectives, and also an ordering of the objectives. The algorithms of Mannor and Shimkin (2001, 2004) also use preferences defined in objective space to specify the desired characteristics of the policy being learnt. In this case a target region in objective space is defined in which the policy's long-term average reward should fall.

Preferences defined in objective space clearly require some pre-existing knowledge of the problem domain, in order to identify the range of values which are achievable for each objective. An alternative approach to specifying preferences is linear scalarisation, which has been used by several authors (for example Natarajan and Tadepalli 2005; Castelletti et al. 2002). The user specifies a weight per objective, and a single objective reward is formed by calculating the weighted sum of the individual objective rewards. Varying the weights allows the user to express the relative importance of the objectives—increasing an objective's weight will bias the learning towards that objective. This approach requires no pre-existing knowledge about the likely values of rewards for each objective. However the relationship between the weights and the policy found may be unpredictable. Depending on the nature of the Pareto front small changes in weights may produce large changes in the policy which is learnt, or vice versa. Linear scalarisation will be discussed further in Sect. 3.2.2.

3.1.2 Multiple-policy algorithms

Multiple-policy algorithms aim to learn multiple policies that form an approximation to the Pareto front. Examples of this class include the approaches of Shelton (2001) and Barrett and Narayanan (2008). Shelton applies policy gradient methods to the multi-objective domain. Gradients in the parameter-space of the policy are calculated individually for each objective and then combined to form a weighted gradient. By varying the weighting of the objective gradients a range of policies can be discovered. Barrett and Narayanan's Convex Hull Value Iteration algorithm learns in parallel all deterministic policies which define the convex hull of the Pareto front, and uses these to form mixture policies (stochastic combinations of deterministic policies) which lie along the boundaries of this hull.

3.1.3 Comparing single-policy and multiple-policy approaches

There are several advantages to the multiple-policy approach of searching for a set of compromise solutions rather than attempting to find a single 'optimal' solution. Methods that aim

for a single solution require a priori decisions from the user about the desired nature of that solution (such as defining the partial-ordering and thresholds for objectives, or specifying the objective weights). This requires domain knowledge on the part of the user, and minor variations in these preferences may result in significant variations in the solution achieved. This can easily lead to the acceptance of a sub-optimal solution. For example, allowing a slightly lower value for one objective may afford significant improvement on all other objectives. Systems which produce sets of solutions allow a posteriori decisions about the solution to be accepted, which are easier and better informed as they are based on knowledge of the trade-offs available as encapsulated by the front. Additionally, the presentation of the front to the user may provide better insight into the interaction between the competing objectives.

The primary disadvantage of generating multiple policies rather than a single policy is the increased computational cost and the increased time spent interacting with the environment. The latter is particularly important in on-line learning tasks within a real environment, as the additional costs incurred in searching for multiple policies may be impractical. By focusing on learning a policy matching the user's preferences, single-policy methods can reduce the costs incurred relative to those preferences during learning, which is extremely important in on-line learning. For this reason we expect that most use of single-policy algorithms will occur in the context of on-line learning. There may also be some situations (such as the web-server task in Tesauro et al. 2007) in which the desired characteristics of the policy are well-known in advance, and hence a single-solution algorithm is suitable even if off-line learning is performed.

Given the widely differing aims of the single-policy and multiple-policy approaches to MORL, different approaches to experimental structure and metrics will be required for each class of algorithm.

3.2 Example multiobjective reinforcement learning algorithms

To aid in presenting the proposed empirical methodology, this paper will use two example algorithms as a basis for discussion and experimentation. These algorithms have been chosen as they are relatively simple, and widely used and cited in the MORL literature. In addition they can serve as examples of both the single-policy and multiple-policy approaches. Both algorithms can be integrated into any value-based RL system, but the implementations used in Sect. 7 are based on Q-learning, so we will first present a general discussion of multi-objective Q-learning before proceeding to the details of the individual algorithms.

3.2.1 Multiobjective Q-learning

Extending Q-learning (or other temporal-difference algorithms) to multiobjective tasks requires two changes to the single-objective version of the algorithm:

- The values learnt by the algorithm must be altered from a scalar to a vector, with an element for each objective. We denote the expected value relative to objective j of performing action a when in state s as $Q_{s,a,j}$.
- Greedy action selection is determined by applying a multiobjective selection mechanism to the vector values for the actions. The two example algorithms used in this paper vary in terms of the operation of the action selection mechanism used to derive a policy from Q.

3.2.2 Linear scalarised reinforcement learning

The first example algorithm is the scalarisation approach that has been discussed and applied by a number of authors, including Aissani et al. (2008), and Perez et al. (2009). This

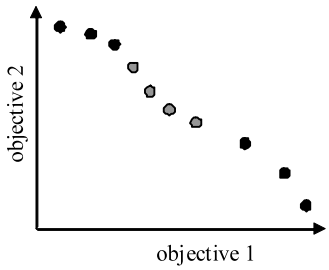


Fig. 3 A Pareto front containing a concave region, as indicated by the grey points

approach reduces a multiobjective task to a single objective by applying a function to the reward vector to produce a single, scalar reward. The scalarisation function may be a non-linear function tuned to the problem domain (Tesauro et al. 2007), but most commonly it is a linear weighted sum of the objective rewards. The choice of weights allows the user some control over the nature of the policy found by the system, by placing greater or lesser emphasis on each objective. The main advantage of scalarisation is its simplicity—it can be integrated into single-objective RL algorithms with very little modification. However linear scalarisation has a fundamental limitation, in that it cannot find policies which lie in non-convex regions of the Pareto front—an example of such policies is shown in Fig. 3.

3.2.3 Thresholded lexicographic reinforcement learning

The second example algorithm is a naïve implementation of the approach of Gabor et al. (1998), which we will call thresholded lexicographic Q-learning (TLQ-learning). This algorithm is designed for problems where one objective must be maximised, subject to satisfying constraints on the other objectives (such as maximizing factory production while maintaining a required safety level). Action selection is performed by applying a combination of thresholding and lexicographic ordering to the objective values of the available actions, as follows:

- Let n denote the number of objectives, labeled from $1..n$
- Let A denote the set of available actions
- Let C_j be the threshold value (minimum acceptable value) for objective j, as defined by the constraint for that objective (note: objective n will be unconstrained, hence $C_n = +\infty$)

$$CQ_{s,a,j} \leftarrow \min(Q_{s,a,j}, C_j)$$

In state s, the greedy action a' is selected such that $\text{superior}(CQ_{s,a'}, CQ_{s,a}, 1)$ is true $\forall a \in A$ where $\text{superior}(CQ_{s,a'}, CQ_{s,a}, i)$ is recursively defined as:

```
if CQ_{s,a',i} > CQ_{s,a,i}
   return true
else if CQ_{s,a',i} = CQ_{s,a,i}
   if i = n
      return true
   else
      return superior(CQ_{s,a'}, CQ_{s,a}, i + 1)
else
   return false
```

3.2.4 Single-policy and multiple-policy applications

Both the example algorithms can be seen as examples of single-policy MORL—given a set of parameters they aim to learn a single policy that is optimal with respect to those parameters. However, as demonstrated by Castelletti et al. (2002), scalarised Q-learning can be applied in a multiple-policy context by performing repeated runs of the algorithm using different parameter values. Similarly a multiple-policy approach based on TLQ-learning can be implemented by performing multiple runs with varying objective threshold values and orderings.

4 Empirical methods and metrics for multiple-policy MORL algorithms

4.1 Overview of multiple-policy evaluation

Evaluating a multiple-policy algorithm involves assessing how well the policies found by the algorithm approximate the true Pareto front for the task, and how rapidly the algorithm discovers these policies. These are essentially the criteria by which multiobjective optimisation algorithms are assessed, and so performance metrics from the well-established multiobjective optimisation literature can be adapted for use in MORL research. Performance analysis in a multiobjective context is complex because the algorithm is endeavouring to satisfy multiple aims—the derived front should be accurate (near to the Pareto front), diverse and well-spread. But how does an accurate, though poorly spread, front compare with a diverse, inaccurate result set? If two fronts are equally accurate, but one has wide reaching extent while the other is evenly distributed, which is preferable?

Early studies in multiobjective optimisation (such as Srinivas and Deb 1994, and Horn et al. 1994) used graphical representations of fronts and basic scalar metrics designed to measure the extent, cardinality, diversity or accuracy of the prevailing front. This approach has two problems: interpretation of visual output is potentially biased, lacks statistical rigour and is difficult to assess in all but clear-cut cases of superiority; and simple metrics which evaluate only one aspect of a front are potentially misleading and prone to inconclusive results as unless all measures point to a single algorithm, it is difficult to indicate the preferable system. More importantly these metrics can prove misleading, as they may indicate a front to be superior to another front when the latter actually weakly dominates the former (Knowles et al. 2006). Therefore performance metrics should be based on comparative measures which are compatible with the Pareto dominance relation. A metric is Pareto compliant if, and only if, it indicates preference for front a over front b, if b does not dominate a. As the majority of diversity, cardinality and spread measures are not Pareto compliant, the implication is that composite metrics, which seek to produce a single output based on the multi-faceted Pareto front, are the superior option (Berry 2008). Several Pareto-compliant composite metrics have been proposed and utilised in the multiobjective optimisation literature—here we will discuss the widely used and accepted hypervolume metric (Zitzler and Thiele 1999).

4.2 The hypervolume indicator

Given a point r which is dominated by all members of a frontal set S, the hypervolume of space that is dominated by members of S, and dominates r can be calculated (Fig. 4). With respect to performance, if S is the prevailing front of an optimiser, the larger the resultant hypervolume, the better the algorithm has performed. An important feature of this metric is

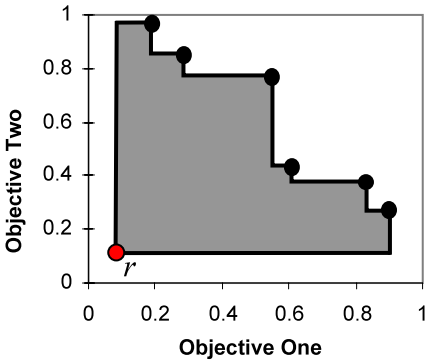


Fig. 4 An example hypervolume: The *shaded area*, bounded by the prevailing front and the reference point *r*, represents the region from which the hypervolume is derived

that improvements in any frontal characteristics (accuracy, extent, diversity) will be reflected in increased hypervolume. Therefore this metric provides a single value by which the relative performance of two multiple-policy learning algorithms can be compared. The value of the hypervolume is dependent upon the choice of reference point, so r must be consistent between experiments to allow meaningful comparison of results—this is best achieved by defining r in the specification of any benchmark problems. If the true front (or a good approximation thereof) is known then the hypervolume of this front can also be calculated and used as in establishing the absolute quality of performance of a learning algorithm on that task.

4.3 Multiple-policy evaluation methodology

As well as choosing a suitable metric, it is vital that this is applied in an appropriate manner. As noted in Sect. 3.1.3, multiple-policy methods are most likely to be used in off-line learning, and as such the most important factor for measurement is the quality of the final policies learnt by the system, rather than the rewards received during learning. Hence the appropriate use of the hypervolume metric is based on the accumulated reward while following a fixed policy for a given period of time, with no exploration or learning during this period. Kaelbling et al. (1996, p. 242) refer to this as a "train/test perspective" on evaluation, as it is similar to the evaluation structure commonly used in supervised learning. We will refer to this as the offline hypervolume, to emphasise that it is not measured during the learning process.

A second issue to be considered is the point in learning at which this "test period" evaluation should be carried out. Many researchers in multiobjective optimisation report the values of metrics at only a single point in the optimization process (Berry 2008). This can be misleading—simple cases such as the one in Fig. 5 show that results can change depending on when the metric is evaluated (B is better early; A is preferable later). This can be avoided by evaluating the hypervolume metric at periodic intervals during learning process. The frequency of these testing periods, the length of each testing period and the total length of training will be dependent on the difficulty and level of stochasticity associated with the learning problem, and as such should be defined as a component of any benchmark problem (for example, problems with noisy reward structures will require a longer testing period to give an accurate estimate of the true value of a policy).

As with empirical testing of conventional single-objective RL, it is necessary to carry out such evaluations across multiple executions of each learning algorithm, to account for stochasticity in the environment and in the algorithms themselves.

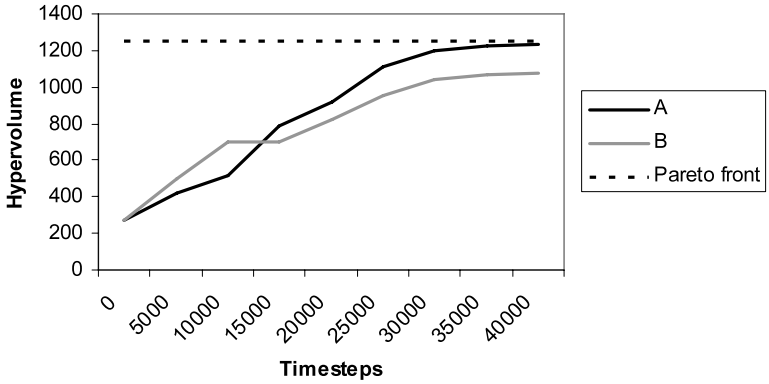


Fig. 5 A simulated example of the performance of multiple-policy algorithms: The relative performance of Algorithms *A* and *B* varies depending on the point in learning at which the results are compared—as discussed above the aim is to maximise the hypervolume metric, which is measured in test periods occurring at fixed intervals during learning. The hypervolume of the Pareto front provides a reference point for the absolute performance of the algorithms

5 Empirical methods and metrics for single-policy MORL algorithms

5.1 Overview of single-policy evaluation

As described in Sect. 3.1.1, the single-policy algorithms proposed in the literature vary widely in terms of how preferences over policies are specified, and in terms of the assumptions made about the underlying problem being solved. As such constructing an evaluation metric which can be applied across all single-policy algorithms is difficult. However all single-policy algorithms share a common aim—to efficiently learn a single policy which best matches the user's specified preferences. In this way single-policy MORL algorithms have much in common with traditional single-objective RL algorithms (which also learn only a single policy, generally in an online-context), and hence evaluation methods for this class of algorithms can draw on the single-objective RL literature. When evaluating single-objective RL algorithms we are concerned with two aspects of their behaviour—how closely the policy to which they converge matches the optimal policy; and how quickly they converge to this final policy.

That is to say, the rewards received during the learning process are important, which differs from the off-line learning evaluation we have proposed for multiple-policy algorithms. On-line performance can be measured in two main ways—as the accumulated reward during the learning process (either total or average-per-time-step), or via the regret metric (Berry and Fristedt 1985, as recommended by Kaelbling et al. 1996), which is the accumulated loss of reward during the learning process compared to following the optimal policy from the beginning of this process. Each metric has its own strengths and weaknesses—regret provides an indication of the performance of an algorithm relative to the best possible performance, but requires knowledge of the optimal policy, whereas accumulated reward measures can be calculated without any knowledge of the optimal policy, but do not provide a direct measure of the absolute quality of performance. In the following subsections we will consider how accumulated reward and regret metrics may be applied in a multi-objective context.

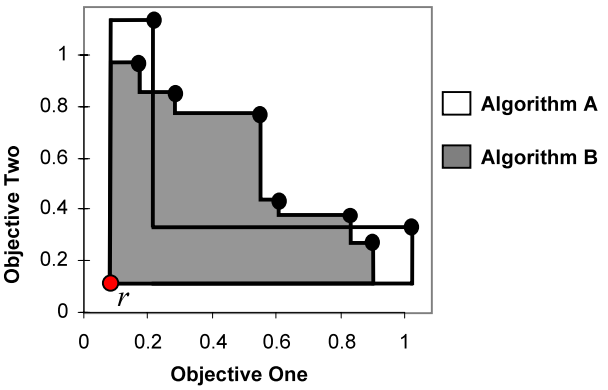


Fig. 6 Example accumulated-reward hypervolumes for two hypothetical MORL algorithms. Points represent the accumulated reward received by an algorithm for a particular set of preferences over the objectives; enclosed regions represent the hypervolume for each algorithm

5.2 Multiobjective accumulated reward metrics

Regret-based performance metrics require knowledge of the performance of the optimal policy or policies—in multiobjective learning this means knowledge of the true Pareto front. While this is possible for simple problems such as the benchmarks in Sect. 6, it may not be possible for more complex problems, and so in those cases metrics based on accumulated reward must be used instead.

The basic implementation of such metrics is straightforward—an MORL algorithm is executed for a given period of time with a particular set of preference values, and the reward vectors received by the agent are accumulated. However how are the accumulated-reward vectors for different algorithms to be compared? If one algorithm's vector dominates the other, then clearly the first algorithm is preferable. However if the accumulated reward vectors are incomparable in a Pareto sense, then we must consider a metric which judges how well each algorithm has satisfied the preferences given to it. For some forms of preference specification this may be relatively easy—for example if both algorithms are based on linear scalarisation, then the weighted sum of their reward vectors can be directly compared. However more generally we may wish to compare algorithms which express their preferences in different, possibly incompatible, ways.

In this case we recommend that the learning algorithms be applied over a range of preference settings, and a hypervolume metric be calculated at regular intervals. However, unlike the offline hypervolume metric used for multiple-policy methods, in this case the metric is based on the accumulated reward during learning so as to provide a measure of the algorithm's online performance. Consider how this online hypervolume metric would apply to the MORL algorithms illustrated in Fig. 6. Algorithm A learns quickly, but always converges to a policy which strictly favours one objective or the other. In contrast algorithm B learns more slowly, but produces a larger number and more even distribution of policies. Algorithm B will be favoured by the online hypervolume metric.

5.3 Multiobjective regret

While the online hypervolume is the best measure to use in the absence of knowledge of the true front, where the front is available regret-based metrics can be more informative. Mathematically, the extension of regret to multiple objectives is straightforward—it is treated as a vector value, rather than a scalar value as shown in (1), where T is the number of time-steps

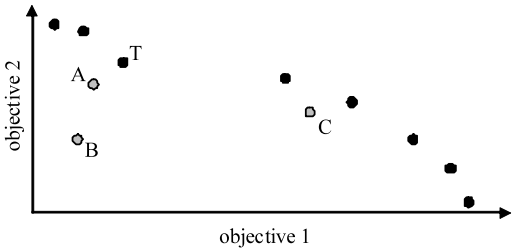


Fig. 7 An illustration of multidimensional regret—*black points* show the performance of Pareto optimal policies, with T being the target policy for a given set of preferences. A, B and C represent the online rewards achieved by different learning algorithms—B is dominated by both A and C

over which the regret is measured, $\boldsymbol{r}_t$ is the vector reward received at time t, and $\boldsymbol{\rho}^*$ is the average vector reward when following an optimal policy:

$$\boldsymbol{R}_T = T\boldsymbol{\rho}^* - \sum_{t=0}^{T-1} \boldsymbol{r}_t \tag{1}$$

The regret vectors for different algorithms can be compared in a number of ways. The simplest case is when the regret vector for one algorithm dominates that of the second algorithm—clearly in this case the first algorithm is preferable. More generally the regret vectors may be incomparable, and so other methods of comparison must be used to determine which has best satisfied the user's preferences. A possible candidate would be to use the length of the regret vector. However as shown in Fig. 7 this metric may favour policies which are dominated—the length metric would correctly prefer policy A as being closest to the target policy T, but would favour policy B over policy C even though C dominates B.

This occurs because it is possible for a policy being evaluated to have outperformed the target policy relative to some (but not all) objectives. To ensure compatibility with the strong Pareto dominance relation, we instead recommend using the length of the non-negative components of the regret vector as the scalar measure of regret as shown in (2).

$$\boldsymbol{R}_S = \sqrt{\sum_{j=0}^{N} (\max(0, \boldsymbol{R}_{T,j}))^2} \tag{2}$$

The second key step in adapting the concept of regret to the multiobjective case is the issue of identifying the appropriate member of the Pareto front to be used as the target policy. In an on-line learning context, the user has essentially only one chance at specifying their preferences prior to the commencement of learning,[4] and so the ease with which they can specify parameters which actually guide the algorithm towards their desired region in objective space may in fact have as much impact on the performance of the system as the actual learning abilities of the algorithm itself. Therefore we would argue that to fully assess the capabilities of single-policy MORL algorithms, they must be evaluated in the context of their use by human subjects—the following Sect. 5.4 will discuss an experimental methodology for this user-based testing. However we acknowledge that widespread user-based testing may prove impractical, and so Sect. 5.5 proposes an experimental methodology based on simulating the role of the user.

[4] Although we can envisage interactive learning algorithms in which the user observes the performance of the agent as it learns, and dynamically alters their preference settings to 'steer' the agent towards a desirable policy—to our knowledge no such algorithm has yet been applied to MORL problems.

5.4 User-based testing of single-policy algorithms

The setting of preferences by the user plays an essential role in the online performance of a single-policy algorithm—a poor choice of parameters may guide the policy to a region of objective space well away from that desired by the user. An algorithm which learns slowly but with intuitive parameters may outperform a faster learning algorithm with less intuitive parameters. Single-policy methods can use a diverse range of approaches to specify these preferences such as scalarising weights (Natarajan and Tadepalli 2005), objective thresholds (Gabor et al. 1998) and ranges in objective space (Mannor and Shimkin 2004), and it is important to establish which of these approaches can be most effectively utilized by users. The experimental methodology discussed in this section is designed to address this issue by evaluating algorithms within the context of usage by human subjects.

A series of benchmark scenarios will be presented to the user. For each scenario the user will specify a set of parameters expressing their preferred solution. If more than one algorithm is being evaluated the user must specify preferences in the format required for each algorithm—for example, for scalarised Q-learning they would specify objective weights, while for TLQ-learning they would specify the objective ordering and threshold values. The user will then be shown the actual Pareto front, and will select their desired solution from this set.[5] This policy is then used as the target for calculating the regret metric as the learning algorithms are applied with the user's preference parameters. If the parameters guide the learning algorithm towards a different policy, then the regret values will be higher, reflecting the fact that the user's preferred policy was not discovered.

User-based testing introduces additional concerns which do not usually need to be considered in RL research. To avoid bias introduced by a single user, a diverse set of users must be tested. To provide a realistic situation in which the user carries out the setting of preferences, each scenario must provide some guidance as to the nature of an acceptable solution (but without directly guiding them as to the actual values of the preference parameters). Finally in order to test an algorithm's ability to handle a range of different preferences the scenarios given to different users must be altered so as to guide them towards different regions of the Pareto front. These additional considerations mean that this user-based testing will be time-consuming. Therefore while we suggest it would be an interesting study into the usability of single-policy MORL systems, we do not anticipate that this approach will be the standard method for evaluating single-policy algorithms. Therefore in the next section we propose an alternative, less resource-intensive evaluation methodology based on simulation of the setting of preferences by a user.

5.5 Simulated user testing of single-policy algorithms

To overcome the practical limitations of user-based testing, there is a need for an automated approach which simulates the user's actions, namely the specification of parameters and selection of a target policy from the Pareto front. A set of parameters can readily be calculated algorithmically (for example, randomly), but an approach is then required to identify the appropriate target policy corresponding to those parameters. A simple approach would be to apply the learning algorithm using the specified parameters, and then to set the target policy

[5]The user is not shown the Pareto front until after they have specified their preferences as this better reflects the real situation in which the algorithms would be applied, where the user would be required to specify preferences without any existing knowledge of the nature of the front.

to be the Pareto optimal policy which is closest in objective-space to the learnt policy. However this could be quite misleading. As an extreme example consider a hypothetical learning algorithm which simply maximised its performance on the first objective regardless of the preference settings. Clearly this is a poor algorithm, yet it would score quite well as the target policy selected would always be the Pareto optimal policy which maximises the first objective. While this example is pathological, the potential exists for similar problems on any learning algorithm which is biased towards particular regions of the Pareto front. For example Vamplew et al. (2008) argued that linear scalarisation algorithms will fail to find Pareto-optimal policies which lie in concave regions of the Pareto front—the evaluation approach outlined here would not penalise these algorithms for this failing, as these policies would also be unlikely to be selected as the target policy.

To address this issue a means is required to determine the appropriate target policy directly from the parameter settings, rather than based on the policy found by the algorithm. Our proposed method to achieve this is based on the assumption that while a user may not have a priori knowledge about the shape and extent of the Pareto front, they will have expectations about where the policy found using a particular set of preference values will lie with respect to those extents. The exact nature of these expectations will vary depending on the nature of the preferences used by the learning algorithm. Therefore we will first propose a general method for simulated user testing of any single-policy algorithm, before illustrating how this method would be adapted for each of the example learning algorithms.

5.5.1 General algorithm for simulated user testing

1. Iterate through a range of preference parameter settings. For each set of parameter values:
   a. Map the parameter settings to a point P_T which is a member of the Pareto front
   b. Use the policy corresponding to P_T as the target policy.
   c. Calculate the regret vector R_T with respect to the target policy using (1).
   d. Convert the regret vector R_T into scalar regret value R_S via (2).
2. Summarise the R_S values across the range of parameter settings.

The exact details of the mapping in Step 1a will vary based on the nature of the preference parameters used by the algorithm being evaluated. It is vital that this mapping is performed in an unbiased manner so as to provide a valid comparison between the different algorithms being evaluated.

5.5.2 Simulated user testing of scalarised Q-learning

In the linear scalarisation approach to MORL, the user specifies a weight for each objective such that these weights sum to 1. If the user only cares about the value of one objective, then clearly they set its weight to 1 and all other weights to 0. If they regarded all objectives as equally important they would specify equal weights and, in the absence of any knowledge about the nature and shape of the Pareto front, would expect a solution lying near the middle of the range of possible outcomes for both objectives. If the first objective's weight was slightly higher than that for the second objective then they would expect the solution to be closer to the Pareto front extrema that favours the first objective—exactly how far the result moves towards this extreme will depend on the shape of the Pareto front. The simulated user mimics these expectations via the following process:

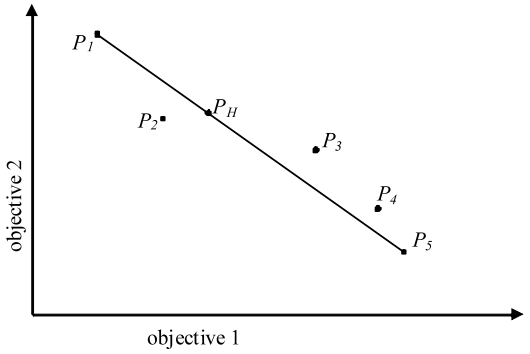


Fig. 8 An illustration of the simulated user testing approach to linear scalarisation for a problem with two objectives. Points P_1 and P_5 represent the extremes of the Pareto front, while P_2, P_3 and P_4 are other points on the front. Point P_H is the mapping of the user's preference weights (0.4, 0.6) onto the hyperplane passing through P_1 and P_5 calculated as $P_H = 0.4P_5 + 0.6P_1$

1. Establish the maximum and minimum bounds achievable for each objective by Pareto-optimal strategies, either from the true front, or from an approximate front estimated from the results of previous experiments.
2. Construct a hyperplane passing through the extremal points of the front (it seems reasonable that in the absence of any prior knowledge of the front, the user may assume that it is flat).
3. For a given set of preference weights W, calculate a point P_H on the hyperplane by treating W as a set of weights for a barycentric coordinate system with basis points being the extrema of the hyperplane.
4. Identify the point P_T on the Pareto front which minimises $|P_H - P_T|$.

Figure 8 illustrates this process for a problem with two objectives. The weights of (0.4, 0.6) are mapped onto the hyperplane, and the policy corresponding to the closest Pareto front point (in this case P_2) is selected as the target policy.

5.5.3 Simulated user testing of TLQ-learning

For TLQ-learning, the user specifies an ordering of the objectives, and a threshold value C_j representing the minimum acceptable level for each objective (except the last). As these threshold values have a direct interpretation within objective space, mapping them to select a target policy on the Pareto front is straightforward—the thresholded lexicographic ordering is simply applied to all Pareto front policies, as illustrated for a two-objective problem in Fig. 9.

The main issue to be considered here is the nature of the values used to specify the objective-space position of the Pareto front policies. These must match the values being learnt by the MORL agent—for episodic tasks the values used should be the undiscounted per-episode return for each Pareto policy, whereas for continuing tasks they should be either the discounted return or the average reward for the policy, depending on which of these values the agent is learning.

5.5.4 Summarising regret metric results

Once regret vectors have been calculated and scalarised for a range of preference values, these can be summarized in a number of ways to gain insight into the behaviour of the algorithm. The average R_s value across all preference settings provides a single measure of the performance of a single-policy algorithm, and therefore is a good basis for comparison between algorithms. Examining the variation in R_s between different preference settings will

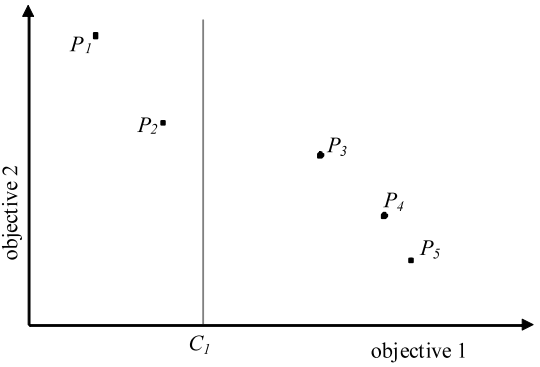


Fig. 9 An illustration of simulated user testing for TLQ-learning for a problem with two objectives. Points P_1 and P_5 represent the extremes of the Pareto front, while P_2, P_3 and P_4 are other points on the front. Line $C1$ shows the user-specified criteria for objective 1. $P3$ will be selected as the target policy as it scores higher on objective 2 than any of the other policies which satisfy criteria $C1$

provide insight into any potential weaknesses—in particular identifying the characteristics of the target policies corresponding to the maximum values of R_s achieved by a particular algorithm should aid in understanding the limitations of that algorithm.

6 Multiobjective RL benchmarks

6.1 The need for multiobjective benchmarks

As discussed in Sect. 2.3 it is difficult to assess the relative merits of current MORL algorithms as there have been no standard benchmarks—authors have reported results on a small number of problems, and there has been no overlap between the test problems used in different papers. In addition the Pareto fronts have not been known for most test problems, so systems that actually work quite poorly may have previously been accepted, as there was no baseline on which to judge their competence (Vamplew et al. 2008). Therefore there is a clear need for a suite of benchmark problems which provide a range of characteristics in order to fully evaluate the performance of MORL algorithms. Ideally the Pareto front should be known for these problems to provide a measure of the absolute quality of performance of MORL algorithms on these tasks. The suite should contain problems which exhibit a mixture of the following characteristics:

- two or more objectives—as noted by Berry (2008) biobjective problems have special properties such that algorithms which perform well on these tasks may perform poorly on problems with three or more objectives;
- stochasticity in transition dynamics and/or rewards;
- continuous state or action spaces;
- state dimensionality high enough to require the use of function approximation;
- partially-observable state;[6]

[6]As an aside, Chaterjee et al. (2006) and Vamplew et al. (2009) have shown that in the context of multi-objective tasks it is necessary to consider policies involving stochastic action selection. The front derived from deterministic policies provides only a discrete set of tradeoffs between objectives, which may be widely spaced in objective-space, whereas stochastic policies offer a continuous range of trade-offs between the objectives, thereby making it more likely that a solution acceptable to the user can be found. In addition some policies which lie on the Pareto front of deterministic policies may in fact be dominated by stochastic policies. As stochastic policies have previously been considered in the context of problems with partially observable state, extension of these methods to multiobjective tasks may be fruitful.

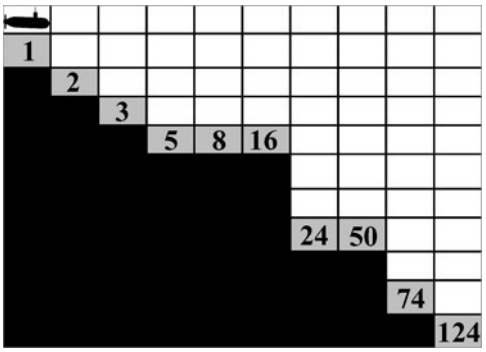


Fig. 10 The Deep Sea Treasure environment. *Black cells* indicate the sea-floor; *grey cells* indicate a treasure location. The submarine marks the position in which the agent commences each episode

- a mixture of episodic and continuing tasks;
- different Pareto front features such as concavities and discontinuities.

6.2 Current benchmarks

We have established a website at http://hdl.handle.net/102.100.100/4461 to act as a repository for benchmark problems for multiobjective reinforcement learning. Currently this site provides details for four benchmarks—our intention is to extend this benchmark suite by adding additional problems over time. For each benchmark the environment's dynamics and reward structure is given, and the Pareto front points are provided in a CSV file. To our knowledge these are the only MORL tasks with known Pareto fronts, and as such they provide key support for our experimental methods. The remainder of this sub-section will describe these benchmarks, while Sects. 6.3 and 6.4 will address future extensions of this benchmark suite. The first three benchmarks described are from Vamplew et al. (2008). The fourth is drawn from Barrett and Narayanan (2008).

6.2.1 Deep Sea Treasure

This episodic problem was created specifically to highlight the limitations of scalarisation. The environment is a grid of 10 rows and 11 columns, as shown in Fig. 10. The agent controls a submarine searching for undersea treasure. There are multiple treasure locations with varying values. There are two objectives—to minimise the time taken to reach the treasure, and to maximise the value of the treasure. Each episode commences with the vessel in the top left state, and ends when a treasure location is reached or after 1000 actions. Four actions are available to the agent—moving one square to the left, right, up or down. Any action which would cause the agent to leave the grid will leave its position unchanged. The reward received by the agent is a 2-element vector. The first element is a time penalty, which is -1 on all turns. The second element is the treasure value which is 0 except when the agent moves into a treasure location, when it is the value indicated in Fig. 10.

The Pareto front formed by the ten non-dominated policies is illustrated in Fig. 11. The front is globally concave, and also has local concavities at the second, fourth and sixth points from the left.

6.2.2 MO-Puddleworld

Puddleworld (Boyan and Moore 1995) is a two-dimensional environment, which has previously been used as a single-objective RL bench-mark. The agent starts each episode at

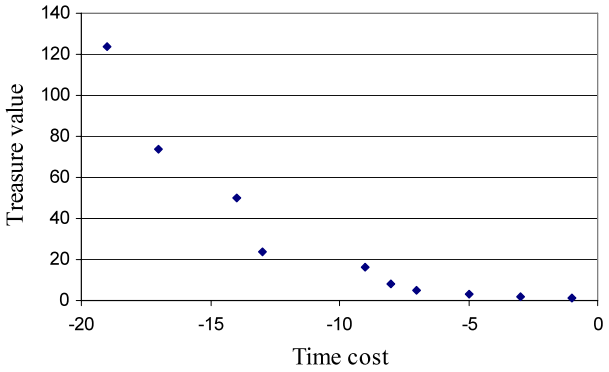


Fig. 11 The Pareto front for the Deep Sea Treasure problem

a random, non-goal state and has to move to the goal in the top-right corner of the world, while avoiding the puddles. The agent receives its current coordinates as input, and at each step selects between four actions (left, right, up or down) which move it by 0.05 in the desired direction. At each step a small amount of gaussian noise (standard deviation 0.01) is also added. The agent's position is bounded by the limits of the world $(0, \ldots, 1)$. The reward structure for Puddleworld is interesting, as it is effectively a form of scalarisation with fixed weights for the two objectives of reaching the goal quickly and avoiding the puddles. On each step on which the goal is not reached, the agent receives a penalty of -1. An additional penalty is applied when the agent is within a puddle, equal to 400 multiplied by the distance to the nearest edge of the puddle. To convert this problem to a multiobjective task, we simply present the two penalties as separate elements of a reward vector (omitting the multiplication by 400, as it is no longer relevant).

In order to facilitate the evaluation of the Pareto front, it was necessary to make several alterations to the original problem specification in order to limit the number of policies which needed to be considered. The noise added to the movement of the agent was omitted. The goal was enlarged from its original triangular shape to fill the entire 0.05 unit square in the top-right corner of the world. The environment is shown in Fig. 12. With these alterations in place, and through the application of several manually identified constraints, it was possible to identify all non-dominated policies to construct the Pareto front shown in Fig. 13. The overall shape of the front is clearly convex. However a closer inspection of the front reveals a number of subtle local concavities and linearities, as shown in Fig. 14. It should be noted that to limit the search required to produce this front, the policies considered were based on a 20×20 discretisation of the state space, and therefore policies based on a finer-grained discretisation or on continuous state values may improve slightly on this front.

6.2.3 MO-Mountain-Car

The Mountain-Car task (Sutton 1996) requires a car to escape from a valley. The car's engine is less powerful than gravity, and so it must reverse up the left side of the valley to build enough potential energy to escape from the right side. The inputs are the current position and velocity, and there are three actions—full throttle forward, full throttle backward, and zero throttle. In the single-objective case a penalty of -1 is received on all steps on which the goal is not reached.

To test the generality of MORL systems, it is important that some benchmarks involve more than two objectives. Therefore this task was converted to a multiobjective case by

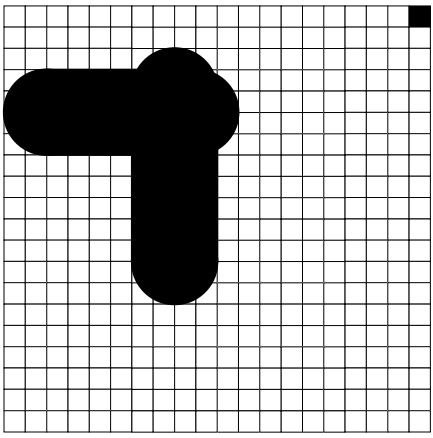


Fig. 12 MO-Puddleworld. The goal is the *black square* in the *top-right corner*, and the *puddles* are capsules with radius 0.1, defined by the line segments (0.1, 0.75) to (0.45, 0.75), and (0.45, 0.4) to (0.45, 0.8). *Grid lines* show the boundaries of the cells used in the discretisation of the space

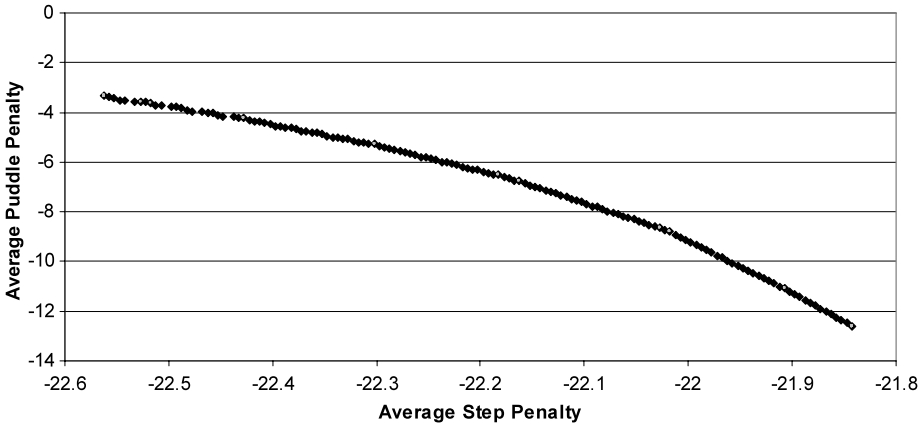


Fig. 13 The Pareto front for the MO-Puddleworld problem

adding two further objectives—minimising the number of reversing and acceleration actions. −1 is received in the corresponding element of the reward vector whenever one of these actions is executed. Interestingly in the single-objective case the zero throttle action is largely redundant as it is rarely beneficial, whereas in the multiobjective formulation, the choice of when to choose zero throttle is one of the key differences between policies.

As with Puddleworld, it was necessary to restrict the policies to a discretised state space in order to evaluate the front—in this case a 6×6 discretisation was used. The space of all policies was explored with a depth-first search with pruning to identify the front.[7] It should be noted that this front only considers policies which actually escape from the valley—this would need to be handled as a constraint by any MORL system, as it does not directly arise from the reward structure.

[7]No illustration of this front has been included as its 3-dimensional nature and the large number of points involved make interpretation of an image extremely difficult.

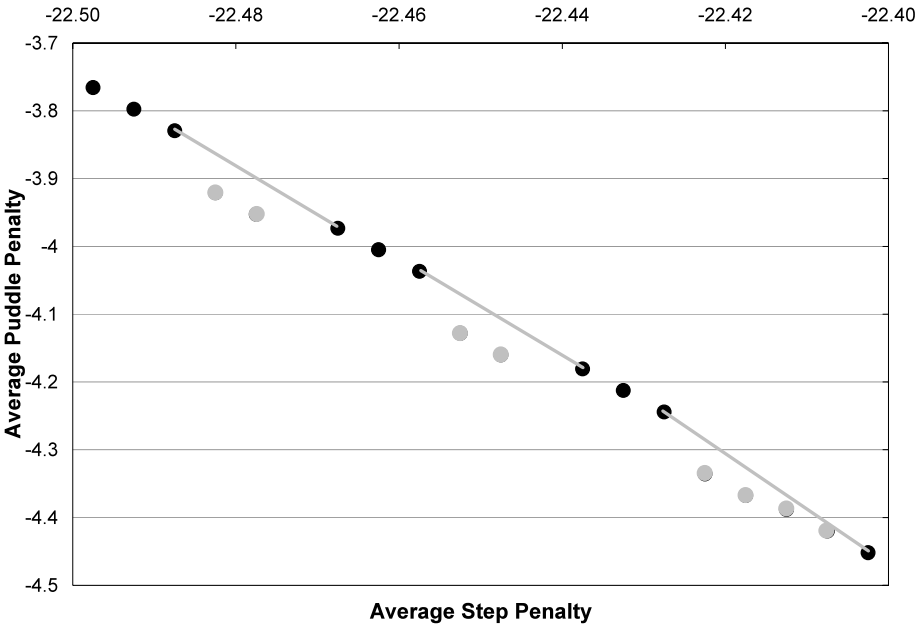


Fig. 14 A close-up view of one region of the MO-Puddleworld Pareto front, showing local concavities (solutions contained in concave regions are shown in *grey*—line segments have been added to highlight the concavity)

6.2.4 Resource Gathering

This task is drawn from Barrett and Narayanan (2008) and is inspired by the resource gathering tasks required in many real-time strategy games. An agent begins at the home location in a 2D grid as shown in Fig. 15, and can move one square at a time in each of the four cardinal directions. The agent's task is to collect either or both of two resources (gold and gems) which are available at fixed locations, and return home with these resources. The environment contains two locations (indicated by swords) at which an enemy attack may occur, with a 10% probability. If an attack happens, the agent loses any resources currently being carried and is returned to the home location. The reward vector is ordered as [enemy, gold, gems] and there are four possible rewards which may be received on entering the home location (there is zero reward on all other time-steps):

- $[-1, 0, 0]$ in case of an enemy attack;
- $[0, 1, 0]$ for returning home with gold but no gems;
- $[0, 0, 1]$ for returning home with gems but no gold;
- $[0, 1, 1]$ for returning home with both gold and gems.

This environment has a discrete state space of 100 states corresponding to the 25 grid cells in which the agent may currently be positioned, multiplied by the four possible states of resources currently held (none, gold only, gems only, both gold and gems). Unlike the previous benchmarks, this task is continuous rather than episodic. Barrett and Narayanan (2008) identified six non-dominated deterministic policies for this problem using their Convex Hull Value Iteration (CHVI) algorithm with a discounting term of 0.9. These policies and their objective-space positions are shown in Fig. 16. CHVI finds policies lying on the

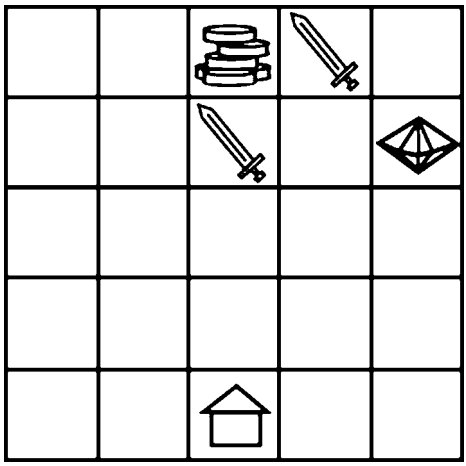
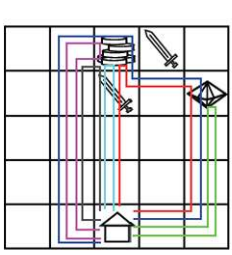


Fig. 15 The environment for the Resource Gathering task (Barrett and Narayanan 2008)



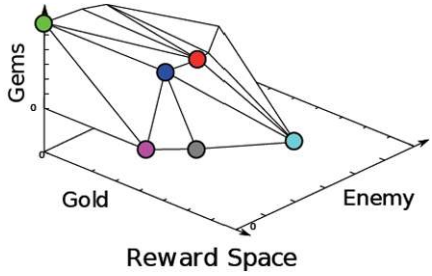


Fig. 16 The policies for Resource Gathering found by CHVI with a discounting factor of 0.9 (*left*), and the hull formed in objective-space by these policies (*right*) (Barrett and Narayanan 2008)

convex hull of the Pareto front—for a discount factor of 0.9 there are only six such policies (a seventh policy which retrieves both gold and gems while avoiding both enemies lies in a concave region of the front). However if a discounting parameter of 0.96 or higher is used, then the values of the policies shift such that this seventh policy is also on the convex hull. To avoid this sensitivity to the discounting parameter, some researchers may wish to address this task using average-reward rather than discounting approaches—the average-reward-per-time-step values of the seven non-dominated policies are also available from our repository.

6.3 Creating standardized benchmark implementations

As discussed in Sect. 2.3 over recent years the RL research community has moved towards standardised implementations of benchmarks and algorithms, to ensure consistency in results. To maximize the utility of the benchmarks and metrics reported in this paper we are developing for public release a standard version of these in a framework based on RL-Glue (Tanner and White 2009). This has been used as the basis for the empirical study reported in Sect. 7.

The RL-Glue framework extends to multiobjective cases fairly simply—the main change required is that the *env_step* function implemented by RL-Glue environments needs to return

a vector reward, rather than a scalar value. The other main change required is that each environment must provide access to those features required to calculate the evaluation metrics (the reference point r and, if known, details of the environment's Pareto front). In addition there may be a need for learning algorithms to also maintain the non-dominated set of the policies which they have discovered. Therefore functions for storage and maintenance of non-dominated sets are provided for use by both agents and environments. Similarly utility functions for the calculation of the hypervolume and regret metrics described in Sects. 4 and 5 of this paper have been implemented.

A final issue to be considered concerns the ongoing maintenance of approximate fronts for each benchmark. As the benchmark suite is expanded to include more complex problems, it is unlikely that it will be possible to calculate the true front for each new problem. This is not a substantial problem as the applications of the hypervolume metrics proposed are independent of knowledge of the true front. However it will still be useful to maintain an approximation to the true front consisting of the set of non-dominated policies that have so far been discovered, as this provides a basis for judging the absolute level of performance achieved by a learning algorithm. When each benchmark is first released, an initial approximate front can be established by applying standard algorithms to the benchmark. However as more sophisticated MORL algorithms are developed and applied to each benchmark, any superior policies that may be discovered should be incorporated into the front stored for that benchmark. The mechanism by which these fronts are updated needs to ensure the integrity of the front is maintained, by validating potential new policies before admitting them into the front. The server capability provided by RL-Glue provides a possible solution to this approach—agents can connect remotely to the server and execute the new policy within the environment. The server can then automatically update the front should the policy prove to be non-dominated with respect to the current front.

6.4 Critique of proposed benchmarks

The benchmark problems described in Sect. 6.2 address the lack of standard benchmarks which has limited previous empirical studies of MORL, by providing well-documented test environments which will be supported by publically available implementations. These problems are particularly valuable as their Pareto fronts are known. However the restrictions imposed on the problems in order to facilitate the calculation of the fronts mean that this nascent benchmark suite currently does not exhibit all of the desirable features outlined in Sect. 6.1.

All the current benchmarks are deterministic in their state transition dynamics and rewards (other than the stochastic nature of the enemy attacks in the Resource Gathering Task) which simplified the calculation of the fronts. However clearly the assumption of non-stochasticity may not be valid for many of the tasks to which we might like to apply MORL. Therefore there is a need to augment the current problems with benchmarks that do incorporate stochasticity. For these more complex tasks it may not be possible to calculate the true front and so only relative measures of the performance of algorithms on these tasks will be possible.

Similarly the first three problems are all undiscounted episodic tasks, and while Barrett and Narayanan (2008) present the Resource Gathering task in a discounted, non-episodic context, it could quite naturally be framed as an episodic task given the regular returns to the home state. Therefore there is a need to add further non-episodic tasks to the suite. The network routing simulation of Natarajan and Tadepalli (2005) is both stochastic and non-episodic, but establishing the true front for problems of this nature may not be feasible—

instead an approximation of the front based on the best policies found while testing algorithms will be maintained in the repository. A similar approach is likely to be needed for benchmarks exhibiting additional complexities such as continuous or high-dimensional state and actions spaces, or partially-observable states.

It should also be noted that the Pareto fronts reported only consider deterministic policies, whereas MORL algorithms need not be restricted in this way. This limitation is not significant with regards to the current benchmarks, as Vamplew et al. (2009) demonstrated that for episodic tasks the Pareto front of stochastic policies can be constructed from the convex hull of the Pareto front of deterministic policies. However as non-episodic tasks are introduced to the benchmark suite it will be essential to consider stochastic policies when establishing the Pareto fronts for these tasks.

For all of these reasons the benchmarks established in this paper should be seen only as a starting point, rather than a complete benchmark suite. As MORL algorithms grow more sophisticated and as our understanding of the factors affecting the performance of these algorithms increases, this initial suite must be augmented by more complex problems spanning the entire range of characteristics described in Sect. 6.1. However the initial suite described here provides a solid foundation for future work in empirical evaluation of MORL algorithms.

7 Demonstrating the effectiveness of the proposed metrics and methods

This section presents a small empirical study as a demonstration of the utility of the empirical evaluation methodologies proposed in the preceding sections. This is not intended to be a comprehensive comparative study, but instead to demonstrate that the proposed metrics are suitably powerful to provide meaningful insight into the performance of MORL algorithms. As such, rather than evaluating a wide range of algorithms across the complete benchmark suite, we will present and discuss results for the two example MORL algorithms on the Deep Sea Treasure task. This will allow for a more detailed discussion of the methodologies than would be possible were multiple benchmarks used. The simple, well understood structure of the selected benchmark task will facilitate this discussion. A more comprehensive study of a broad set of MORL algorithms on the complete benchmark suite is planned as future work.

7.1 Experimental design

Deep Sea Treasure consists of a small number of discrete states, and so there is no need to use function approximation. Thus tabular implementations of the scalarised Q-learning and TLQ-learning algorithms from Sects. 3.2.2 and 3.2.3 were used. The same learning parameters were used for both algorithms, and no attempt was made to fine-tune these values to optimize performance of either algorithm:

- ε-greedy exploration was used with a fixed ε value of 0.1
- the learning rate α was set to 0.1
- the table of state-action values were optimistically initialized (to 0 for the time reward, and to 125 for the treasure reward)
- no eligibility traces were used
- as this is an episodic task, no discounting was used ($\gamma = 1$)

Each algorithm was tested using different sets of preference values. For scalarised Q-learning these values consisted of different objective weights evenly spaced in the range $(0, 1)$ to $(1, 0)$, giving eleven unique sets of weights. For TLQ-learning, the preference values consisted of an ordering of the objectives, and eleven different objective threshold values, which were in the range -19 to -2 for the time objective, and 0 to 125 for the treasure objective.

For each set of preferences, ten learning trials were performed. The results reported in the following sections are based on the mean of the rewards achieved over the ten trials. Following the methodology from Sects. 4 and 5, each trial consisted of alternating training and testing phases. Each training phase consisted of 500 time-steps. During testing the agent's performance was measured as it executed its current greedy policy with no learning or exploration. As the Deep Sea Treasure task is deterministic with a fixed starting point for each episode, this testing phase consisted of only a single episode. This process was repeated 50 times, so that a total of 25,000 time-steps of learning occurred within each trial.

7.2 Multiple-policy results and discussion

As noted in Sect. 3.2.4, while the example algorithms are single-policy in nature, they can be applied in a multiple-policy context by combining the policies achieved by multiple executions of the algorithm using different preference settings. For this experiment this was achieved by performing the eleven executions in sequence, and then collating the results.[8] Figure 17 shows the offline hypervolume achieved by each algorithm at each testing phase, based on a reference point of $(-100, 0)$. The hypervolume of the true front is also shown for purposes of comparison.

From Fig. 17 it can be seen that TLQ-learning converges rapidly to the true Pareto front when thresholding is applied to the treasure objective, achieving the maximum possible hypervolume value of 10455. However the performance of TLQ-learning is dependent on the ordering of the objectives, as it performs extremely poorly when the time objective is thresholded. This variation is explained by the nature of the rewards associated with these objectives—non-zero treasure rewards are received only on the final step of an episode, whereas the time reward is non-zero on all steps. The naïve TLQ-learning algorithm fails in the presence of the latter type of reward as its action selection mechanism considers only the expected future reward for each action from the current state, ignoring any rewards received earlier in the current episode. Consider an agent using this algorithm with a threshold for the time reward of -6. At the start of an episode the agent will begin following the policy leading to the treasure location with a value of 3, as it has the highest treasure reward of the policies that exceed the time threshold. However as the agent nears this location, the expected time penalty for moving to a deeper treasure location will drop below the threshold and so the agent will instead move towards that location. This switching between treasures may occur several times depending on the value of the threshold. The overall effect will be that TLQ-learning will be unable to find some of the Pareto-optimal policies regardless of the value of the threshold. To our knowledge this failing of TLQ-learning has not previously been identified in the literature, although Geibel (2006) reported experimental evidence of instability in learning using this approach.

The hypervolume of scalarised Q-learning never reaches that of the true front, peaking at a maximum of 10062. Examination of the policies learnt by scalarised Q-learning indicates

[8] No attempt was made to improve performance by re-using learning between these executions, although this has previously been shown to improve learning speed (Natarajan and Tadepalli 2005).

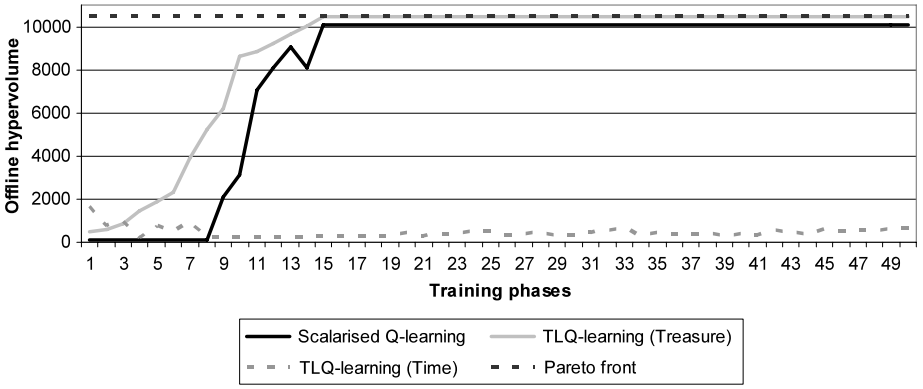


Fig. 17 The offline hypervolume performance of the example MORL algorithms on the Deep Sea Treasure task

that this impaired performance is due to the algorithm failing to learn any of the policies which lie in the concave region of the Pareto front—this empirical observation supports the previous theoretical analysis of the algorithm by Vamplew et al. (2008).

7.3 Single-policy results and discussion

While the offline hypervolume results provide information about each algorithm's ability to learn the Pareto front policies, they do not illustrate the online behavior of each algorithm. To achieve this, the single-policy performance of each algorithm was measured as outlined in Sects. 5.2 and 5.5. The reward for each objective was accumulated over each training phase. At the end of each phase this accumulated reward was converted into an average per episode. To show how single-policy performance would be evaluated if the true front were not known the online hypervolume of the average reward vector was calculated—results based on this metric are reported in Sect. 7.3.1. As the true front for this task was known, the average reward vector was also used to calculate the multidimensional regret relative to a target policy, as reported in Sect. 7.3.2. In practice only one of these measures would normally be used (regret if the true front is known; online hypervolume if not)—both are used here to validate the online hypervolume approach, and to demonstrate the benefits of the regret metric.

7.3.1 Single-policy hypervolume metric results

Figure 18 shows the online hypervolume of each algorithm after each training phase, averaged across all trials. The results show that while the performance of TLQ-learning thresholding the treasure objective improves slightly faster than that of scalarised Q-learning, their overall performance is similar (TLQ-learning achieves a maximum mean online hypervolume of 10001, compared to a maximum of 9867 for scalarised Q-learning). This is due to their shared reliance on Q-learning and ε-greedy exploration. More variation between the online and offline results would be evident when comparing algorithms based on different underlying RL mechanisms (such as SARSA or policy-gradient methods), or using varying forms of exploration. As with the offline results, TLQ-learning thresholding the time objective performs poorly with a maximum online hypervolume of 969.

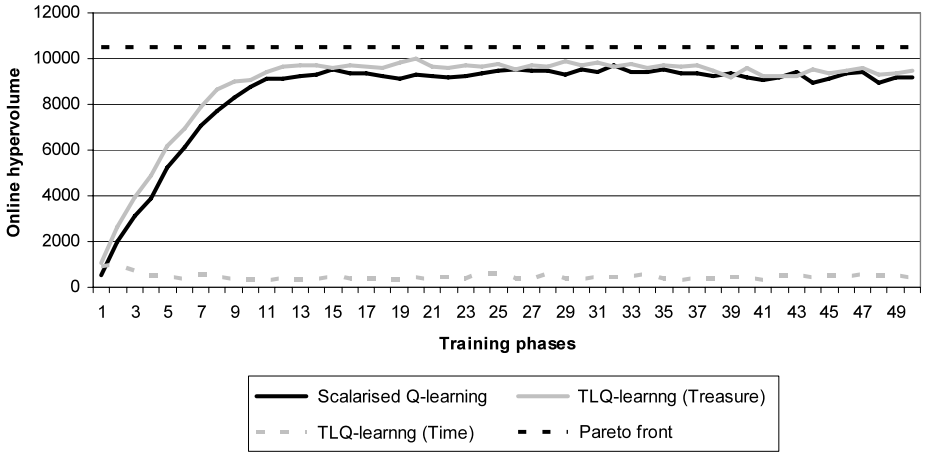


Fig. 18 The online hypervolume performance of the example MORL algorithms on the Deep Sea Treasure task

7.3.2 Single-policy regret metric results

Figure 19 shows the regret metric performance of each algorithm after each training phase, averaged across all trials. TLQ-learning based on the time objective performs poorly with a minimum regret value of 28.4. TLQ-learning based on the treasure objective gives much better performance achieving a minimum regret value of 2.6, while scalarised Q-learning's best regret value is 11.0. It can be seen that by utilizing information about the actual front, the regret metric more clearly highlights the difference in the online performance of the example algorithms than does the online hypervolume metric. The regret metric highlights that scalarised Q-learning's performance actually peaks after 10 training phases, and subsequently degrades as it converges to the two non-concave policies—this aspect of scalarised Q-learning's behavior is not evident from the online hypervolume results.

It can be seen that the regret metric provides more information than the online hypervolume, and therefore should be used where possible. However the online hypervolume does still identify the same overall trends as the regret metric, and therefore offers a valid metric where the Pareto front is not known.

7.4 Summary of demonstration

This small empirical study has demonstrated the ability of our proposed evaluation methodology and metrics to support meaningful comparisons between different MORL algorithms. The offline hypervolume metric has been shown to clearly indicate differences between the quality of the final Pareto fronts found by MORL algorithms. For online performance, it has been demonstrated that the regret measure provides a clearer comparison of algorithmic performance than does the online hypervolume metric. However where regret cannot be calculated as the true front is not known, the online hypervolume still provides useful information. For the particular algorithms used in this study, the online and offline metrics indicate similar trends between the algorithms. This is because both algorithms share the same underlying Q-learning structure, and in particular the use of ε-greedy exploration.

This study has also demonstrated the value of a well-understood benchmark problem in explaining the differing performance of algorithms. The vast differences in the performance

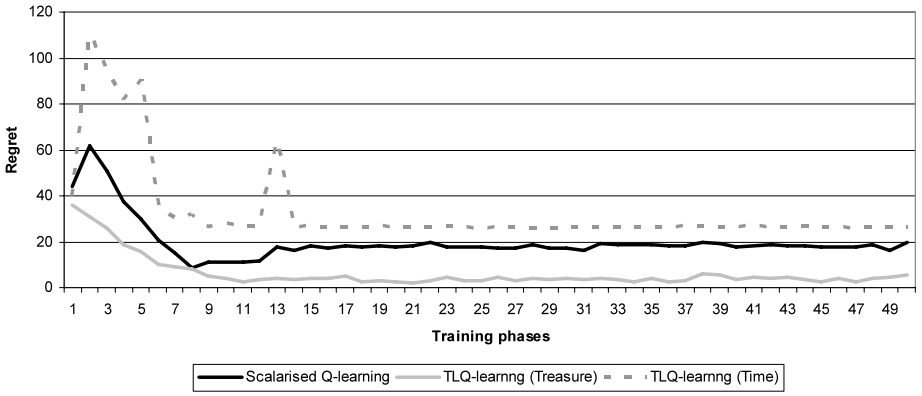


Fig. 19 The online regret performance of the example MORL algorithms on the Deep Sea Treasure task. A lower regret value indicates superior performance

of TLQ-learning depending on the objective ordering can be understood based on knowledge of the structure of the rewards of the Deep Sea Treasure task. Similarly knowing the true Pareto front for this task is vital to understanding the failings of scalarised Q-learning on this task.

8 Conclusion

Research into algorithms for MORL is a relatively new field of study, and as such the development of standard methods for empirical evaluation of associated algorithms has so far not been undertaken. Most studies have tested algorithms on a limited number of learning tasks that have been unique to that paper and lacking some of the features that would be desirable for a rigorous testing of the capabilities of the algorithm. Comparison of algorithms between studies has been difficult due to the lack of uniformity in test problems and methodologies, and the lack of known Pareto fronts for the test problems has prevented the establishment of absolute measures of performance for each algorithm.

We have addressed this problem by proposing standard metrics, experimental methodologies and benchmark tasks that in combination will support rigorous empirical evaluation of MORL systems. It has been shown that the differing application contexts of the two classes of MORL algorithms (online learning of single policies and offline learning of multiple policies) require alternative experimental methods and evaluation metrics, and suitable approaches have been described for each class of algorithm.

It has also been shown that there are significant advantages to using benchmark tasks with known Pareto fronts, and a benchmark suite of four such tasks has been provided. The limitations of this suite have been identified, and further extensions of this suite (both in terms of the number and nature of the problems, and in the provision of standardised implementations and interfaces) have been outlined.

The utility of our proposed metrics, experimental methodologies and benchmark tasks has been demonstrated by a small empirical study based on two simple MORL algorithms. This study indicated that the metrics were sufficiently powerful to identify limitations in these algorithms, and that knowing the properties of the benchmark problem (in particular the fact that the front was known) provided significant insight into the reasons for the failures of these algorithms.

Acknowledgements We wish to thank Peter Andraea for his useful comments on single-policy methods and online learning, Leon Barrett for permission to use Figs. 15 and 16, Brian Tanner for advice on RL-Glue, and the reviewers of the original submission of this paper for their insightful feedback.

References

Aissani, N., Beldjilali, B., & Trentesaux, D. (2008). Efficient and effective reactive scheduling of manufacturing system using sarsa-multi-objective agents. In *MOSIM'08: 7th conference internationale de modelisation and simulation*, Paris, April 2008.

Barrett, L., & Narayanan, S. (2008). Learning all optimal policies with multiple criteria. In *Proceedings of the international conference on machine learning*.

Berry, A. (2008). Escaping the bounds of generality—unbounded bi-objective optimisation. Ph.D. thesis, School of Computing, University of Tasmania.

Berry, D. A., & Fristedt, B. (1985). *Bandit problems: sequential allocation of experiments*. London: Chapman and Hall.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: safely approximating the value function, NIPS-7.

Castelletti, A., Corani, G., Rizzolli, A., Soncinie-Sessa, R., & Weber, E. (2002). Reinforcement learning in the operational management of a water system. In *IFAC workshop on modeling and control in environmental issues*, Keio University, Yokohama, Japan (pp. 325–330).

Chaterjee, K., Majumdar, R., & Henzinger, T. (2006). Markov decision processes with multiple objectives. In *Lecture notes in computer science: Vol. 3884. Proceedings of the 23rd international conference on theoretical aspects of computer science (STACS)* (pp. 325–336). Berlin: Springer.

Coello, C. A. C., Veldhuizen, D. A. V., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*. Dordrecht: Kluwer Academic.

Crabbe, F. L. (2001). Multiple goal Q-learning: Issues and functions. In *Proceedings of the international conference on computational intelligence for modelling control and automation (CIMCA)*. San Mateo: Morgan Kaufmann.

Dutech, A., Edmunds, T., Kok, J., Lagoudakis, M., Littman, M., Riedmiller, M., Russell, B., Scherrer, B., Sutton, R., Timmer, S., Vlassis, N., White, A., & Whiteson, S. (2005). Reinforcement learning benchmarks and bake-offs ii. In *Workshop at advances in neural information processing systems conference*.

Frank, A., & Asuncion, A. (2010). UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Gabor, Z., Kalmar, Z., & Szepesvari, C. (1998). Multi-criteria reinforcement learning. In *The fifteenth international conference on machine learning* (pp. 197–205).

Geibel, P. (2006). Reinforcement learning for MDPs with constraints. In *ECML 2006: European conference on machine learning* (pp. 646–653).

Handa, H. (2009). Solving multi-objective reinforcement learning problems by EDA-RL—acquisition of various strategies. In *Proceedings of the 2009 ninth international conference on intelligent systems design and applications* (pp. 426–431).

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multiobjective optimisation. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Knowles, J. D., Thiele, L., & Zitzler, E. (2006). *A tutorial on the performance assessment of stochastive multiobjective optimizers* (TIK-Report No. 214). Computer engineering and networks laboratory, ETH Zurich, February 2006.

Mannor, S., & Shimkin, N. (2001). The steering approach for multi-criteria reinforcement learning. In *Neural information processing systems*, Vancouver, Canada (pp. 1563–1570).

Mannor, S., & Shimkin, N. (2004). A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, *5*, 325–360.

Natarajan, S., & Tadepalli, P. (2005). Dynamic preferences in multi-criteria reinforcement learning. In *International conference on machine learning*, Bonn, Germany (pp. 601–608).

Pareto, V. (1896). *Manuel d'economie politique*. Paris: Giard.

Perez, J., Germain-Renaud, C., Kegl, B., & Loomis, C. (2009). Responsive elastic computing. In *International conference on autonomic computing*, Barcelona (pp. 55–64).

Shelton, C. R. (2001). *Importance sampling for reinforcement learning with multiple objectives* (Tech. Report No. 2001-003). Massachusetts Institute of Technology, AI Laboratory.

Srinivas, N., & Deb, K. (1994). Multiobjective optimisation using nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

Sutton, R. S. (1996). Generalisation in reinforcement learning: successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems: proceedings of the 1995 conference* (pp. 1038–1044). Cambridge: MIT Press.

Tanner, B., & White, A. (2009). RL-glue: language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, *10*, 2133–2136.

Tesauro, G., Das, R., Chan, H., Kephart, J. O., Lefurgy, C., Levine, D. W., & Rawson, F. (2007). Managing power consumption and performance of computing systems using reinforcement learning. *Neural information processing systems*.

UMass (2010). University of Massachusetts reinforcement learning repository. http://www-all.cs.umass.edu/rlr/.

Vamplew, P., Dazeley, R., Barker, E., & Kelarev, A. (2009). Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In *Lecture notes in artificial intelligence*. *Proceedings of AI09: the 22nd Australasian conference on artificial intelligence*, Melbourne, Australia, December 2009. Berlin: Springer.

Vamplew, P., Yearwood, J., Dazeley, R., & Berry, A. (2008). On the limitations of scalarisation for multiobjective learning of Pareto fronts. In W. Wobcke & M. Zhang (Eds.), *Lecture notes in artificial intelligence: Vol. 5360*. *Proceedings of AI08: the 21st Australasian conference on artificial intelligence* Auckland, New Zealand, December 2008 (pp. 372–378). Berlin: Springer.

White, A. (2006). *A standard system for benchmarking in reinforcement learningi*. Master's thesis, University of Alberta, Alberta, Canada.

Whiteson, S., Tanner, B., Taylor, M. E., & Stone, P. (2009). Generalized domains for empirical evaluations in reinforcement learning. In *Proceedings of the 4th workshop on evaluation methods for machine learning at ICML-09*, Montreal, Canada.

Wiering, M. A., & de Jong, E. D. (2007). Computing optimal stationary policies for multi-objective Markov decision processes. In *Proceedings of the IEEE international symposium on approximate dynamic programming and reinforcement learning (ADPRL)* (pp. 158–165).

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Grunert da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, *7*(2), 117–132.