On the Limitations of Scalarisation for Multi-objective Reinforcement Learning of Pareto Fronts

Peter Vamplew, John Yearwood, Richard Dazeley, and Adam Berry

School of ITMS, University of Ballarat,
University Drive, Mt Helen, Ballarat, Victoria, Australia
{p.vamplew,j.yearwood,r.dazeley}@ballarat.edu

Abstract. Multiobjective reinforcement learning (MORL) extends RL to problems with multiple conflicting objectives. This paper argues for designing MORL systems to produce a set of solutions approximating the Pareto front, and shows that the common MORL technique of scalarisation has fundamental limitations when used to find Pareto-optimal policies. The work is supported by the presentation of three new MORL benchmarks with known Pareto fronts.

Keywords: multiobjective, reinforcement learning, scalarisation, Pareto fronts.

1 Introduction

Most reinforcement learning (RL) algorithms aim to maximise performance on a single objective. Many problems naturally fit this model, but there has been growing recognition in the optimisation community that many real-world problems exhibit multiple objectives [1], and that specialised multiobjective optimisation (MOO) techniques are required for these problems. Recently there has been interest in developing RL methods to handle multiobjective tasks. This paper argues that so far most multiobjective reinforcement learning (MORL) research has failed to capitalise on the knowledge in the MOO literature. Specifically this paper discusses the role of Pareto dominance within MORL, and examines the limitations of scalarised MORL.

2 Reviewing Existing Approaches to MORL

The easiest way to apply RL algorithms to multiobjective problems is to convert the problems themselves into single-objective tasks. In single-objective RL the reward is scalar, whereas in MORL it is a vector with an element for each objective. So a multiobjective task can be reduced to a single objective via scalarisation, which applies a function to the reward vector to produce a scalar reward. Commonly this is a linear weighted sum of the individual rewards [3, 4]. The weights allow the user some control over the nature of the solution, by placing greater or lesser emphasis on each objective. Less frequently a non-linear function tuned to the problem domain may be used [2]. In the simplest implementation, rewards are scalarised prior to reaching the agent, allowing the learning algorithm to remain unaltered. Alternatively the algorithm may be modified to learn the expected values for each objective, which may facilitate re-use of earlier learning when multiple policies are being found [3].

W. Wobcke and M. Zhang (Eds.): AI 2008, LNAI 5360, pp. 372–378, 2008.

A small number of alternatives to scalarisation have also been proposed. [5] assumes a known partial ordering of the objectives, and threshold values which must be achieved for objectives (e.g. a robot maintaining a non-zero energy level whilst accomplishing some task). This approach has been applied in the specific context of risk-sensitive learning [6, 7]. [8, 9] describe MORL algorithms to achieve long-term average rewards lying within an externally defined 'target' region in objective space. These produce non-stationary policies where actions are influenced by the current state and by the position of the current average reward relative to the target region.

These existing MORL systems find single solutions, whereas most MOO systems aim to produce a set of solutions which form a range of good compromises between the objectives. A 'good' compromise is often defined in terms of Pareto dominance. A solution dominates another if it is superior on at least one objective, and at least equal on all others. They are incomparable if each is superior on at least one objective (see Fig 1, which assumes the aim is to maximise each objective's value). A dominated solution is of little value, as the dominating solution is preferable. If all dominated solutions are eliminated from the set of all possible solutions, the resulting set is the globally optimal set of compromise solutions, known as the Pareto optimal front (see Fig 2). Of course finding the true front for any substantial problem is impractical, and so the goal is to produce a set of solutions which approximates the true front.

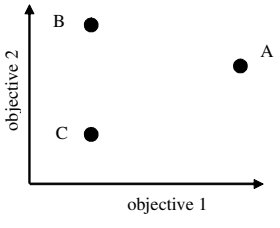


Fig. 1. Solutions A and B dominate C; solutions A and B are incomparable

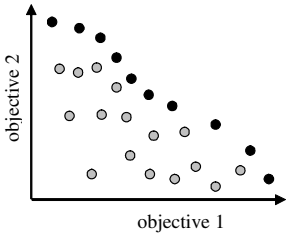


Fig. 2. Black points form the Pareto front; all grey points are dominated by at least one black point

There are several advantages to producing a set of solutions rather than a single solution. Methods producing a single solution require *a priori* input from the user about the desired nature of that solution (e.g. defining the partial-ordering and thresholds, or specifying objective weights). This requires domain knowledge on the part of the user, and minor variations may result in significant changes in the solution achieved (e.g. a slightly higher threshold for one objective may prevent discovery of a solution which provides a significant improvement on all other objectives). Systems which produce sets of solutions support *a posteriori* decisions about the accepted solution, which are better informed as they are based on knowledge of the trade-offs available as encapsulated by the front. Also the presentation of the front to the user may provide better insight into the relationships between the objectives. The primary disadvantage of generating multiple solutions is the increased computational cost and, for on-line learning, the increased time spent interacting with the environment.

As noted earlier, most existing MORL systems produce single solutions, but some authors have investigated generating multiple solutions. [10] describes a policy-gradient

MORL algorithm. A policy derived by applying RL independently to each objective, is improved using hill-climbing to follow gradients in the policy space which are non-negative with regards to all objectives. An approximate front is constructed by performing repeated searches with different weightings of the gradients. Whilst this approach is quite sophisticated, most other work on multiple-solution MORL has relied on the simpler combination of scalarisation and TD methods. [4] used a combination of scalarisation and Q-learning to approximate the Pareto front for a lake regulation system with twin objectives of ensuring water supply and providing flood protection. The front is found by repeated runs of the algorithm with varying objective weights. [3] applied scalarised RL to tasks where an external source of the objective weights was assumed, with periodic changes in these weights. The task of the system was to rapidly adapt to the novel weights. The key finding was that performance could be substantially improved by storing and re-using learning from earlier policies. Whilst this work did not directly consider approximating the Pareto front, it could be used for that purpose by replacing the external weight source with a loop which steps through the scalarisation weight space.

Unfortunately, whilst scalarised MORL is simple, it suffers from a fundamental flaw. Any system based on a linear combination of the objectives is incapable of finding solutions which lie in a concave region of the Pareto front [11]. No weights exist which allow a point in a concavity to produce a weighted sum higher than that achieved by the solutions at either edge of the concavity. As many multiobjective problems exhibit non-convex regions, this limitation has significantly reduced interest in scalarisation-based methods in the MOO literature [12]. The following sections of this paper will examine the significance of this limitation in the context of MORL.

3 MORL Benchmarks and Scalarisation Performance

Given the known limitations of scalarisation, why does scalarised MORL continue to be used? We argue this arises from the lack of any means for assessing the performance of a MORL system. MORL research is in its infancy, and no study has yet examined the performance of different algorithms – in fact no standard benchmarks have been established to act as a basis for such a study. In addition, the Pareto fronts were not known for the problems to which scalarised MORL has been applied. Known fronts can provide a measure of the absolute performance of an MORL system by comparing the approximate front produced against the true front. In the absence of such known fronts, it is difficult to judge the quality of the solutions produced by a learning system. Here we address these issues by presenting three benchmark tasks, along with their Pareto fronts. To our knowledge these are the first MORL tasks with known fronts. The first problem was designed specifically for this research. The second and third problems have been drawn from the single-objective RL literature and adapted to create multiobjective tasks. To facilitate future use, full details of the tasks and the data-points describing the fronts are available for download from http://uob-community.ballarat.edu.au/~pvamplew/MORL.html.

Deep Sea Treasure. This task consists of a grid of 10 rows and 11 columns (see Fig 3). The agent controls a submarine searching for treasure. There are multiple

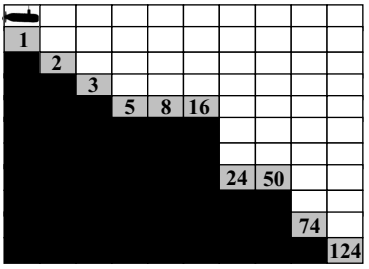


Fig. 3. Deep Sea Treasure: Black cells are the sea-floor; grey cells are treasure locations

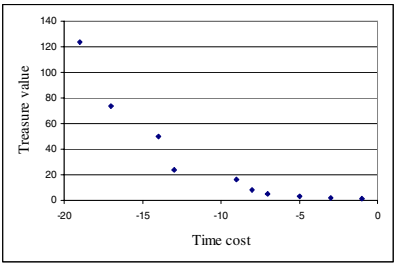


Fig. 4. The Pareto front for the Deep Sea Treasure problem

treasures with varying values. There are two objectives – to minimise the time taken to reach treasure, and to maximise the value of the treasure retrieved. This is an episodic task – each episode starts with the vessel in the top row of the first column, and ends when a treasure is reached or after 1000 actions. At each time-step, four actions are available to the agent – moving one square to the left, right, up or down. Any action which would result in the agent leaving the grid will leave its position unchanged. The reward received by the agent on each turn is a 2-element vector. The first element is a time penalty, which equals -1 on all turns. The second element is 0 on all turns except when the agent moves into a treasure location, when it is the value shown in Fig 3. There are ten non-dominated policies, each of which leads to one of the ten treasure locations. The Pareto front of these policies is shown in Fig 4. The front is globally concave, with local concavities at the second, fourth and sixth points from the left.

MO-Puddleworld. This is a 2-D environment. The agent starts each episode at a random state and must reach the goal in the top-right corner, whilst avoiding puddles. It receives its current coordinates as input, and at each step selects an action (left, right, up or down). The agent's position is bounded by the limits of the world. The reward structure for the original single-objective Puddleworld task [13] 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 applies if the agent is within a puddle, equal to 400 multiplied by the distance to the puddle's edge. To convert this problem to a multiobjective task, we present the two penalties as separate elements of a reward vector (omitting the multiplication by 400, as it is no longer relevant). To facilitate the evaluation of the Pareto front, it was necessary to make several alterations to the original problem specification. The noise added to the movement of the agent was omitted. The policies were based on a 20x20 discretisation of the state-space (although the actual position of the agent in the environment was still modeled as a continuous value). 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. 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 5. The overall shape of the front is convex, but a closer inspection of the front reveals a number of subtle local concavities and linearities, as shown in Fig 6.

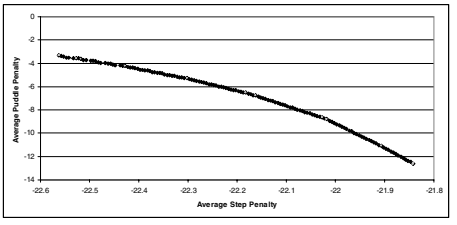


Fig. 5. The Pareto front for the MO-Puddle-world problem

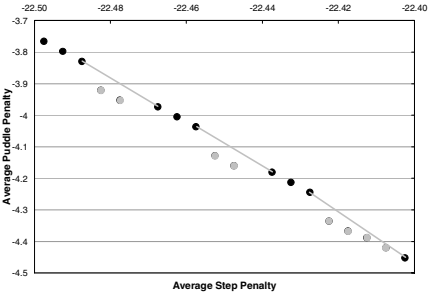
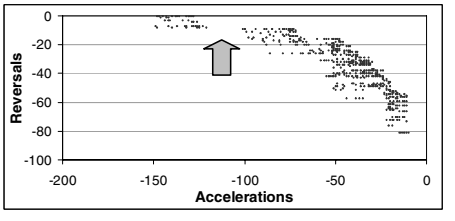
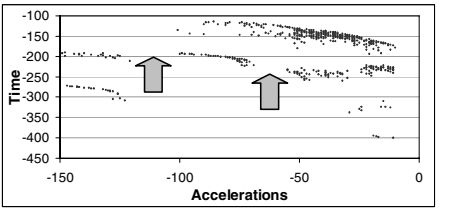
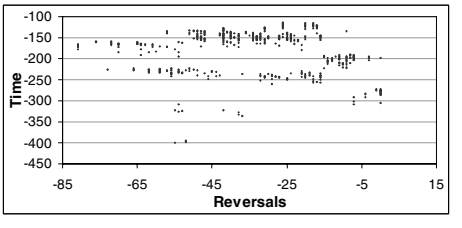


Fig. 6. A close-up view of Fig 5 (solutions in concave regions are shown in grey)







Figs. 7-9. 2-dimensional projections of the Pareto front for the Mountain-Car task, relative to each pair of objectives. The arrows indicate some of the discontinuities in this front.

MO-Mountain-Car. A car must escape from a 1-dimensional valley. The engine is less powerful than gravity, and so the car must reverse up the left side of the valley to build enough momentum to escape from the right side. The inputs are the current position and velocity, and there are 3 actions – accelerate, reverse, and zero throttle. In the single-objective case a penalty of -1 is received on all steps on which the goal-state is not reached [14]. To test the generality of MORL systems, some benchmarks should involve more than two objectives so two further objectives were added – minimising both the number of reversing and the number of acceleration actions. A penalty of -1 is received in the corresponding reward vector element when one of these actions is executed. Interestingly in the single-objective case the zero throttle action is redundant, whereas in the multiobjective task, the choice of when to choose zero throttle is a key difference between policies. As with MO-Puddleworld, it was necessary to restrict the policies to a discretised state space in order to evaluate the front – in this case a 6x6 discretisation was used. The resulting front is shown in

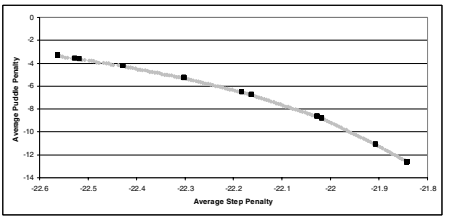


Fig. 10. The Pareto front of the MO-Puddle-world task (in grey) with the policies found by scalarisation superimposed (in black)

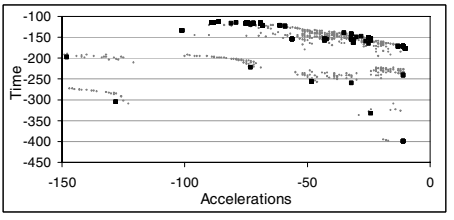


Fig. 11. The Pareto front of the MO-Mountain-Car task (in grey) relative to the acceleration and time objectives, with the policies found by scalarisation superimposed (in black)

Figs 7-9 – as it is difficult to interpret a 3-dimensional view, 2-dimensional projections have been provided. 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.

The performance of scalarisation on the benchmarks was assessed by applying scalarising weights to each known solution, and determining which returned the highest scalarised value. This was repeated for a variety of weights, and the resulting sets of solutions compared to the actual Pareto front. As noted earlier Deep Sea Treasure has a globally concave Pareto front. Whatever weight combinations are applied to this task, only the two extremal points will ever return the maximal scalarised value - the intermediate solutions will be overlooked by any system based on scalarisation. The limitations of scalarisation on MO-Puddleworld are more subtle, but nonetheless substantial. Searching the weight space in steps of 0.01 results in 101 weight sets being applied to the solutions, yet finds only 11 of the 141 unique policies present in the original Pareto front. Fig 10 compares these policies to the true front. It can be seen that the sparse set of policies found by scalarisation is a poor representation of the near continuous true front. A similar result is observed for MO-Mountain-Car. The need to assign weights for all 3 objectives increases the size of the weight-space to be searched, so using a step-size of 0.1 results in 66 weight combinations yet only detects 28 of the 470 unique solutions in the actual front. A step-size of 0.05 tests a 231 weight combinations, and locates only 41 policies. Fig 11 compares these 41 policies against the actual Pareto front. Once again, the approximate front produced by scalarisation is a poor match for the actual front. In particular the distribution is quite different from that of the actual front.

4 Conclusions and Future Work

This paper has argued for following the lead of MOO research, by developing MORL systems which approximate the Pareto front of policies, rather than producing just a single solution. Presenting the user with a range of solutions provides more information about the trade-offs between objectives, and allows an informed decision without the need to impose *a priori* biases on the nature of the solution. We have examined the use of scalarisation within Pareto-based MORL showing that it may be unable to produce a good approximation of the Pareto front for many problems, due to its inability to locate policies in non-convex regions of the front. Whilst the limitations of scalarisation are

well known amongst MOO researchers, they have been overlooked by many MORL practitioners. Hence we believe that there is a pressing need for MORL algorithms designed expressly to address the issue of deriving an approximation to the Pareto front. [11] has provided pioneering work in this area. It may also prove fruitful to extend existing methods such as [5], by performing multiple iterations of the algorithm using different values for the thresholds.

This paper also presented 3 test problems which are the first MORL tasks with identified Pareto fronts, making them valuable benchmarks for future research. To support such further use, it is important that this set of benchmarks be extended as the problems presented share a number of features such as their limited state-space dimensionality, non-stochasticity and episodic nature. Whilst these features aided in finding the Pareto fronts, they also simplify the task facing a learning agent. Therefore these benchmarks need to be augmented with a more diverse range of problems to better represent the challenges posed to MORL systems in real-world tasks.

References

1. Coello Coello, C.A.: Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. In: 2000 Congress on Evolutionary Computation, vol. 1, pp. 30–37 (2000)
2. Tesauro, G., Das, R., Chan, H., Kephart, J.O., Lefurgy, C., Levine, D.W., Rawson, F.: Managing power consumption and performance of computing systems using reinforcement learning. Neural Information Processing Systems (2007)
3. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. In: 22nd International Conference on Machine Learning, Bonn, Germany, pp. 601–608 (2005)
4. Castelletti, A., Corani, G., Rizzolli, A., Soncinie-Sessa, R., Weber, E.: 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 (2002)
5. Gabor, Z., Kalmar, Z., Szepesvari, C.: Multi-criteria reinforcement learning. In: The Fifteenth International Conference on Machine Learning, pp. 197–205 (1998)
6. Geibel, P.: Reinforcement learning with bounded risk. In: Proceedings of the 18th International Conference on Machine Learning (2001)
7. Geibel, P., Wysotzki, F.: Risk-sensitive reinforcement learning applied to control under constraints. Journal of Artificial Intelligence Research 24, 81–108 (2005)
8. Mannor, S., Shimkin, N.: The steering approach for multi-criteria reinforcement learning. In: Neural Information Processing Systems, Vancouver, Canada, pp. 1563–1570 (2001)
9. Mannor, S., Shimkin, N.: A geometric approach to multi-criterion reinforcement learning. Journal of Machine Learning Research 5, 325–360 (2004)
10. Shelton, C.R.: Importance sampling for reinforcement learning with multiple objectives, Massachusetts Institute of Technology AI Lab Technical Report No. 2001-003 (2001)
11. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: region-based selection in evolutionary multiobjective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 283–290 (2001)
12. Coello Coello, C.A., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithm MOP Approaches (Chapter Two). In: Evolutionary Algorithms for Solving Multiobjective Problems. Kluwer Academic Publishers, Dordrecht (2002)
13. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: Safely approximating the value function. In: Neural Information Processing Systems (1995)
14. Sutton, R.S.: Generalisation in reinforcement learning: Successful examples using sparse coarse coding. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems, pp. 1038–1044 (1996)