Synergies between Reinforcement Learning and Evolutionary Dynamic Optimisation

Aman Soni, Peter R. Lewis, and Anikó Ekárt

Aston Labs for Intelligent Collectives Engineering (ALICE)
Aston University, Birmingham, United Kingdom,
sonia2@aston.ac.uk,
http://alice.aston.ac.uk

Abstract. A connection has recently been drawn between dynamic optimization and reinforcement learning problems as subsets of a broader class of sequential decision-making problems. We present a unified approach that enables the cross-pollination of ideas between established communities, and could help to develop rigorous methods for algorithm comparison and selection for real-world resource-constrained problems.

1 Introduction

Decision problems are often encountered in system design and control. If the underlying environment changes, the problem is dynamic and requires new decisions over time. There is a need for robust problem-independent search algorithms in cases where knowledge of optimization function can only be gained through sampling, or if there are insufficient resources to construct a problem-dependent algorithm [1]. Current approaches to construct problem-independent algorithms are to frame the problem as a Dynamic Optimisation Problem (DOP) [2] or as a Reinforcement Learning Problem (RLP) [3].

DOPs, often tackled using Evolutionary Dynamic Optimisation (EDO) algorithms, are usually seen as tracking moving optima problems [2]. On the other hand, RLPs are often defined as Markov Decision Processes (MDPs) [3]. Fu et al. [4] proposed Sequential Decision-Making Problems (SDMPs) as a class of problem that includes DOPs and RLPs. This is useful as EDO and RL algorithms specialise in different types of SDMPs [5] and the SDMP perspective creates an opportunity to cross-pollinate ideas across well-established research.

To realise this opportunity, SDMP definitions should include sufficient information for EDO and RL algorithms to be substituted or hybridised. This creates a need for methods to compare performance across EDO and RL algorithms. To understand how an algorithm will perform on real-world problems with limited resources, comparisons should be made under varying resource constraints. These comparisons can be used to create a rigorous method for algorithm selection. There may be further applications for meta-heuristics at run-time.

This research has immediate applicability to challenging real world problems in domains of robotics [6], space [7], vehicle routing [8], traffic control [9], image recognition [10], inventory management [11] and game play [12].

2 DOPs versus RLPs

Based on Fu et al. [4], we define a DOP as: *Given an optimization problem F, an optimization algorithm G to solve F, and an optimization period $[t_0, t_e]$, F is a DOP if during $[t_0, t_e]$ the underlying fitness landscape changes and G has to react by providing new solutions where E calculates the estimated reward at each time step.* In contrast to Fu et al. [4] we consider fitness function f to be static. When the state of the environment at time t is a set of variables s_t, and the action taken at time t is a_t, a DOP can be defined[1] as:

$$max \sum_{t=0}^{t_e} E(f(s_t, a_t)). \tag{1}$$

Sutton and Barto [3] define the general form of the RLP as a problem where an agent, for a sequence of discrete time steps, t, at each t evaluates its representation of the environment's state, $s_t \in \mathbb{S}$, where $\mathbb{S}$ is the set of all possible states, and selects an action to perform, $a_t \in \mathbb{A}(s_t)$, where $\mathbb{A}(s_t)$ is the set of available actions for the state s_t. As a consequence of the action, a_t, the environment moves to a new state, s_{t+1} and provides a numerical reward to the agent for the next time step, $r_{t+1} \in \mathbb{R}$. The agent builds a *policy* π for each t, where $\pi_t(s, a)$ is the probability that the selected action $a_t = a$ if $s_t = s$. The solution is a policy π that maximises[2] the total sum of expected rewards. If r denotes the reward function, an RLP can be defined as:

$$\sum_{t=0}^{t_e} r(\text{arg max}[\pi_t(s_t, a_t)], s_t). \tag{2}$$

The similarities between the definitions of DOP (1) and RLP (2) are striking and support the view that DOPs and RLPs are subsets of a broader class of SDMPs [4]. In both definitions the reward (or fitness) returned is determined by the effects of an action on the current state of the environment. Each problem seeks to maximise the accumulated sum of the returned value and algorithm performance is determined by the values accumulated over the interval $[t_0, t_e]$.

Two significant differences relate to time-linkage between states. The first is that DOP (1) accumulates the maximum reward at every t while RLP (2) maximises the accumulated sum of discounted future rewards. The second difference is that the state representation in DOP (1) is a part of the problem specification while RLP (2) tracks the received rewards using state-action pairs (s_t, a_t) in the learning policy π. This implies that DOP (1) does not cater for time-linkage between states, while the policy π in RLP (2) tracks state transitions.

Another similarity is, EDO algorithms compare individuals in a population set, while RL algorithms select from a policy-value set. These actions occur at at every at time step.

[1] Notation has been adjusted to aid comparison to RLP (2).
[2] Maximization problems are considered without a loss of generality.

3 Towards a Unified Approach

EDO algorithms assume a fitness function [4], and typically use an offline environment before the solution is applied to the problem instance [13]. On the other hand, RL algorithms assume some observability of state and a reward function [3]. A unified approach can provide guidance so that current tacit community knowledge is made explicit in problem definition. This makes synthetic and real-world problems available to both EDO and RL algorithms.

RL algorithms can learn online under dynamic conditions without a need for an *a priori* model or simulator [3]. This is a major benefit when the model is unknown or too complex to simulate accurately [6]. Additionally, RL algorithms exploit time-linkage between states, making convergence to an optima quicker if this information is available [3]. However, RL algorithms are considered both slow and resource intensive [14]. On the other hand, EDO algorithms need only evolve policies that directly map states to actions, a major advantage in problems with large or continuous action spaces [15]. EA policies need only specify an action for each state, which can be simpler to represent than learning methods that specify the value of each state-action pair [15]. Suitable policy representations can be evolved rather than needing to be design beforehand [14]. When state information is uncertain, it may be advantageous to use EDO algorithms [6].

Fig. 1 shows the unified problem definition space, typical algorithms that overlap, and corresponding gaps in both problems and algorithms. A unified approach provides further opportunities to develop new algorithms, based on the strengths of both EDO and RL, to add to the work that currently spans these research communities.

In this section we present four key open research questions that we see as necessary to realise in order to gain the benefits of a unified SDMP framework.

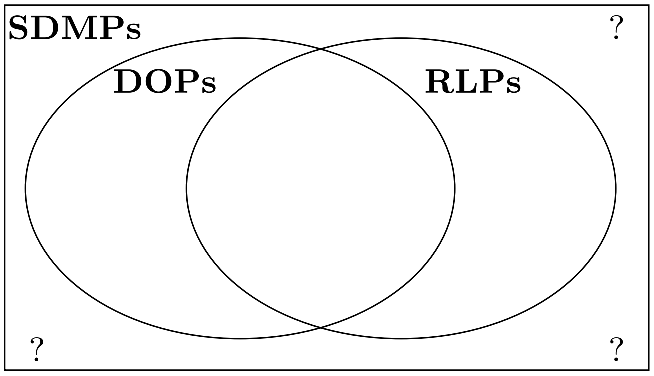

Fig. 1: Unified sequential decision-making problem definition space with corresponding typical algorithms and techniques. Source [16].

What are the overarching dimensions of the SDMP space? We propose an approach that divides characteristics of SDMPs into two high-level categories, namely the environment state information and the dynamics of the environment. Environment state information would define SDMP dimensions for the number, range and type of state variables. The dynamics on the environment include the degree [17], rate and pattern of change [2] along with affect of time-linkage between states [3] and certainty in sensing and state transition prediction [6]. Positioning of problem instances in the SDMP space will then be determined by dynamics of the environment state, the size of the state-action space, and the effect of time-linkage between states.

There is a need for configurable synthetic problems that provide good coverage of the SDMP definition space (Fig. 1) to test algorithm performance on different dimension-value combinations. For some problems, dimension values may be static for the instance, while for others they may change.

How should we measure algorithm performance? The solution requirements have a determining role in measuring algorithm performance. Navigation problems require an always available solution, although it can change during navigation, while a manufacturing robot requires control systems with a high level of precision that should not change during operation. Collision avoidance requires real-time responses in a potentially rapidly changing environment. Whether an optimum is known, and the desired minimum solution quality with respect to the optimum, should be considered for algorithm selection and required training, or for bounding algorithm run-times. Other considerations are certainty of time between decision points and whether the relationship of the current state to the life-time of the problem is known.

We argue that it is that we can only *meaningfully compare algorithm in terms of the solution requirements.*

When can we compare performance across algorithms? Measures, such as generations-to-convergence or best-of- generation are used to compare population-based EDO algorithms [15] and do not allow for comparison to non-population-based algorithms [18]. EDO algorithms currently ignore simulation time [4] for performance measures. Furthermore it is not clear how the solution is applied to the environment [2][4]. While many RL algorithms operate online, there are cases where prior knowledge is gained from training [19], or updates to the learning policy are delegated [6]. *SDMPs require methods to compare the online performance of algorithms.*

How do resource constraints effect algorithm performance? In the real world, resources including sensors, memory, power and time are limited. Recent work in this area aims to reduce resource usage [20], or for algorithms to learn to reduce the computational resource and runtime [21]. There is a range of techniques to analyse algorithm time complexity [1] or build models of the run-time [22] that can help gain an understanding of how resource constraints effect solution quality. Use of these techniques could *help design effective algorithms for cases that require specific resources (e.g. battery life) to be conserved.*

A Rigorous Method for Algorithm Selection In practice, there is a necessary trade-off between performance, solution quality and resource usage. Some considerations for algorithm selection include where the algorithm performs well with high-dimensional and continuous states, or if it is know that an algorithm copes well with change in the environment state dynamics. Experimental work on methods to estimate algorithm performance under constrained resources should be measured against the solution requirements.

This research will allow us to gather information to *develop a rigorous method for algorithm selection*. A further application could be to run-time meta-heuristics.

4 Conclusions

We have presented a unifying perspective of RLPs and DOPs as SDMPs. We have argued that a unified SDMP framework allows for the cross-pollination of ideas between well establish research communities. The SDMP framework will be supported by configurable synthetic problems that can model constrained resources, as is the case in the real world.

We proposed the mapping of problems in the SDMP space. This requires both theoretical and experimental research on algorithm performance. The performance should be measured against the solution requirements, over varying resource constraints.

This work would help develop a rigorous method for algorithm selection balanced between solution quality, algorithm efficiency and resource usage. By framing the selection method as an SDMP, this research has potential application to run-time meta-heuristics.

References

1. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing **4**(3) (2007) 281–293
2. Nguyen, T., Yang, S., Branke, J.: Evolutionary dynamic optimization: A survey of the state of the art. Swarm and Evolutionary Computation **6** (2012) 1–24
3. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. Volume 1. MIT press Cambridge (1998)
4. Fu, H., Lewis, P.R., Sendhoff, B., Tang, K., Yao, X.: What are dynamic optimization problems? In: IEEE Congress on Evolutionary Computing (CEC). (2014) 1550–1557
5. Fu, H., Lewis, P.R., Yao, X.: A Q-learning Based Evolutionary Algorithm for Sequential Decision Making Problems. In: Parallel Problem Solving from Nature (PPSN), VUB AI Lab (2014)
6. Wiering, M., van Otterlo, M.: Reinforcement Learning: State-of-the-art. Volume 12. Springer Science & Business Media (2012)
7. Myers, P.L., Spencer, D.B.: Application of a multi-objective evolutionary algorithm to the spacecraft stationkeeping problem. Acta Astronautica **127** (2016) 76–86
8. Tan, K.C., Cheong, C.Y., Goh, C.K.: Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. European Journal of operational research **177**(2) (2007) 813–839

9. Münst, W., Dannheim, C., Gay, N., Malnar, B., Al-mamun, M., Icking, C., Hagen, F.: Managing intersections in the cloud. (2015) 329–334
10. Fergus, R., Fei-Fei, L., Perona, P., Zisserman, A.: Learning object categories from google's image search. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Volume 2., IEEE (2005) 1816–1823
11. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary computation **10**(3) (2002) 263–282
12. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587) (2016) 484–489
13. Jin, Y.J.Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation **9**(3) (2005) 303–317
14. Drugan, M.M.: Synergies between evolutionary algorithms and reinforcement learning. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. GECCO Companion '15, ACM (2015) 723–740
15. Eiben, A.E., Schoenauer, M.: Evolutionary computing. Information Processing Letters **82**(1) (2002) 1–6
16. Soni, A., Lewis, P.R., Ekárt, A.: Offline and online time in sequential decision-making problems. In: IEEE CIDUE, IEEE Press (2016)
17. Uzor, C.J., Gongora, M., Coupland, S., Passow, B.N.: Real-world dynamic optimization using an adaptive-mutation compact genetic algorithm. In: Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014 IEEE Symposium on, IEEE (2014) 17–23
18. Cruz, C., González, J.R., Pelta, D.A.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft Computing **15**(7) (2011) 1427–1448
19. Dearden, R., Friedman, N., Andre, D.: Model based bayesian exploration. In: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc. (1999) 150–159
20. Piatkowski, N., Lee, S., Morik, K.: Integer undirected graphical models for resource-constrained systems. Neurocomputing **173** (2016) 9–23
21. Graves, A.: Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983 (2016)
22. Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K.: Algorithm runtime prediction: Methods & evaluation. International Joint Conference on Artificial Intelligence (IJCAI) (January 2015) 4197–4201