A Survey on Discrete Multi-Objective Reinforcement Learning Benchmarks

Thomas Cassimon
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium
thomas.cassimon@uantwerpen.be

Reinout Eyckerman
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium

Siegfried Mercelis
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium

Steven Latré
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium

Peter Hellinckx
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Antwerp, Belgium

ABSTRACT

In this paper, we investigate the current state of the art in Multi-Objective Reinforcement Learning (MORL) benchmarking for problems with discrete state and action spaces. Our investigation finds that there exist a large number of different MORL benchmarks, but there seems to be little standardization among the MORL community. Through a quantative comparison, it becomes clear that many of the MORL benchmarks being used today do not have the same complexity as their Single-Objective Reinforcement Learning (SORL) counterparts. Based on the existing benchmarks for MORL problems, the authors propose an extended version of the commonly used Deep Sea Treasure (DST) benchmark. The new benchmark is shown to be flexible enough to cover a wide range of complexities, from an image-based alteration of the DST benchmark, to complexities rivalling those used in contemporary SORL research. Finally, the authors also provide an implementation of their benchmark.

KEYWORDS

Multi-Objective Optimization, Multi-Objective Reinforcement Learning, Reinforcement Learning, Deep Learning, Deep Sea Treasure

1 INTRODUCTION

1.1 Multi-Objective Reinforcement Learning

Multi-Objective Reinforcement Learning (MORL) is the subfield of Reinforcement Learning (RL) that attempts to find optimal policies for problems with at least two objectives. Within this field, various techniques have been proposed, such as Non-Stationary Policy Gradients [7], Q-Learning with Lexicographical Thresholding [10] and outer-loop approaches like Deep Optimistic Linear Support Learning (DOL) [19]. A property of MORL or even Multi-Objective Optimization (MOO) in general is that each of the objectives is in conflict with the others. When two objectives can be optimized concurrently, the problem becomes equivalent to optimizing the sum of both objectives, effectively reducing the number of objectives by one.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2022), Cruz, Hayes, da Silva, Santos (eds.), May 9-10, 2022, Online, https://ala2022.github.io/. 2022.

MORL problems are usually formalized using Multi-Objective Markov Decision Processes (MOMDPs), a multi-objective variation of a traditional Markov Decision Process (MDP). In [11], Hayes et al. define a MOMDP as a 6-tuple $\langle S, A, T, \gamma, \mu, \boldsymbol{R} \rangle$:

- S is the state space
- A is the action space
- $T : S \times A \times S \rightarrow [0, 1]$ is a probabilistic transition function
- $\gamma \in [0, 1)$ is a discount factor
- $\mu : S \rightarrow [0, 1]$ is a distribution over initial states
- $\boldsymbol{R} : S \times A \times S \rightarrow \mathbb{R}^d$ is a vector-valued reward function, giving the immediate reward for d different objectives.

1.2 Optimality Criteria

When considering MORL approaches, contrary to Single-Objective Reinforcement Learning (SORL), there are two different optimality criteria. Which optimality criterion applies to a specific problem, depends on how the optimal policy is intended to be used. In their survey on sequential multi-objective decision making, Roijers et al. [22] described these two optimality criteria: Expected Scalarized Return (ESR) and Scalarized Expected Return (SER). In their paper, they formulate ESR as shown in equation 1, where $u()$ is the utility function:

$$V_u^\pi = \mathbb{E}\left[u\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) | \pi, \mu_0\right] \tag{1}$$

The mathematical formulation of SER, on the other hand, is shown in equation 2:

$$V_u^\pi = u\left(\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | \pi, \mu_0\right]\right) \tag{2}$$

Under ESR, a user derives utility from a single roll-out of its policy, while under SER utility is derived from the expected outcome (mean over multiple roll-outs). In the context of MOO, the notions of ESR and SER are valuable, since they describe two fundamentally different types of solutions. It is important to note that, under a linear utility function, ESR and SER are equivalent. A practical example of ESR is selecting the appropriate medical treatment for a patient. Since this treatment will only be carried out once, it is important that the policy selecting the treatment optimizes for this

one policy execution [23]. An example of SER is demonstrated by Khamis et al. in their paper on MORL for traffic signal control [17]. The policy found by Khamis et al. is executed many times, and is only optimal if it can consistently route traffic in an optimal way, under different traffic conditions.

1.3 Many-Objective Optimization

While there is no exact definition of when a Multi-Objective Optimization problem becomes a Many-Objective Optimization (MaOO) problem, the line is typically drawn around 4 objectives, with anything with 4 or more objectives considered a Many-Objective Optimization problem. Saxena et al. [24] define a MaOO problem to have 'significantly more than five' objectives, while [18], [33] and [39] say MaOO problems are any problem with more than three objectives. Despite these differences, it is generally accepted that MaOO problems are significantly more challenging and less intuitive than MOO problems. While the authors thought it important to point out the difference between MOO and MaOO, this paper will restrict itself to MOO problems.

In section 2, the authors discuss existing MOO and MORL benchmarks, and compare them based on how often they have been used, their complexity and the availability of implementations for other researchers to use. Section 3 details a newly proposed benchmark based on gaps identified in the state-of-the-art. Next, section 4 discusses the accompanying Pareto front, and some guidelines when using this benchmark. Finally, the authors provide some potential directions for future research in section 5.

2 STATE OF THE ART

2.1 Existing Benchmark Suites

In MOO literature, there exist numerous benchmarks, such as the Walking Fish Group (WFG) Toolkit [13], ZTD Benchmark [40] and the DTLZ Benchmark [8]. When analyzing benchmarks like ZTD, DTLZ and WFG, we notice that most of them are not tailored towards sequential decision making problems. While this provides a nice playground for Multi-Objective Evolutionary Algorithms (MOEAs), it does not suit many RL problems well as they are usually built as sequential decision making algorithms. Since this publication focuses on MORL, rather than MOEAs, we will omit these benchmarks in further comparisons.

In their 2011 paper [26], Vamplew et al. included a link to a list of benchmarks in their original publication[1]. We will discuss the benchmarks mentioned in [26] here again, evaluating them in light of todays algorithms and challenges.

One of the most popular benchmarks among MORL researchers is the Deep Sea Treasure (DST) benchmark. This benchmark was originally proposed by Vamplew et al. [26] in 2011, due to a lack of standardized, rigorous benchmarking methods for newly developed MORL techniques. Since this benchmark was proposed, the capabilities of RL algorithms have expanded significantly, reducing the level of challenge offered by this benchmark. The main reason for this is the limited size of the state and action space, the fact that it only has two objectives, the simplicity of the underlying transition function, and its determinism.

One possible way of avoiding the triviality of problems like the DST benchmark is by converting their observations from simple, scalar observations to image-based observations. This has been done before in the field of SORL, for example, when using images of Atari games as benchmarks [4]. In the field of MORL, this was a technique used by Mossalam et al. [19] to raise the complexity of the DST problem they used to evaluate their DOL algorithm. While the problem complexity was increased, it was still not as complex as contemporary SORL benchmarks.

The multi-objective mountain car problem (MO-MountainCar) is another common benchmark in the field of MORL. It was also mentioned in Vamplew et al. [26], and used as a benchmark by, among others, Mossalam et al [19]. The single-objective problem that it is derived from is originally continuous, but in their paper, Vamplew et al. propose a discretized version. In Vamplew et al.'s version, the state space is discretized to a 6×6 grid of the original state space, with the standard discrete set of 3 possible actions (forward, idle, reverse). Similar to its SORL counterpart, the MO-MountainCar offers a relatively simple challenge to modern RL algorithms.

MO-Puddleworld is another one of the benchmark problems proposed by Vamplew et al. in their 2011 paper [26]. It is a two-objective problem in which an agent must reach the top-right square of a grid as quickly as possible, without hitting any of the puddles present in the world. These two criteria are the two objectives that must be optimized for in this problem. The original, SORL version of the problem used a continuous state and action space, but similar to MO-MountainCar, Vamplew et al. discretized this, to reduce the number of policies that need to be evaluated to find a complete Pareto front. We were not able to find any papers that used MO-PuddleWorld for evaluation of their methodology.

The Resource Gathering problem originally proposed by Barrett and Narayanan [2] is another MORL benchmark discussed by Vamplew et al. [26]. In the Resource Gathering task, an agent moves along a discrete grid, gathering different resources (gold and gems in the original). These resources can be gathered on specific spaces. There are also spaces occupied by enemies which may attack an agent with a certain probability. Agents are rewarded based on three criteria: Enemy attacks, gold gathered, gems gathered. Rewards for resources are only awarded to the agent if it manages to reach the starting square carrying resources without being attacked along the way.

Another, more recently proposed problem, is the fruit tree navigation task, proposed by Yang et al. in their paper "A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation" [38]. In this task, an agent must navigate a full binary tree of depth d, to arrive at a leaf node. Each leaf node has a 6-objective reward associated with it, and leaf nodes are constructed in a such a way that there exists a scalarization for which that leaf node is optimal. Similar to the original DST problem, the fruit tree navigation problem is completely discrete. While tasks like the Fruit Tree Navigation problem can serve to check if a new MORL algorithms could feasibly work, when compared to current SORL problems, the complexity they offer is limited.

[1] https://web.archive.org/web/20120324223820/http://uob-community.ballarat.edu.au/ pvamplew/MORL.html

In their recent paper "A Practical Guide to Multi-Objective Reinforcement Learning and Planning" [11], Hayes et al. argue that many of the currently used MORL benchmarks are quite simple. One of the possible solutions they propose is the multi-objectification of SORL benchmarks. One such example is Yang et al.'s use of a Super Mario game in their paper [38]. Yang et al. modified the SORL version of Super Mario Bros. [16] to include five objectives: x-position, time, deaths, coins, and enemies. This creates a benchmark which is on par with most contemporary SORL benchmarks in terms of complexity (when used in an image-based manner), and offers a relatively large number of objectives. We note that, since this benchmark has more than three objectives, it is technically considered a MaOO benchmark.

Hayes et al. [10], in their paper on Dynamic Thresholded Lexicographic Ordering, use a problem called Dynamic Economic Emissions Dispatch (DEED). DEED was originally proposed by Basu [3], and optimized using Nondominated Sorting Genetic Algorithm - II (NSGA-II). In DEED a MORL agent must manage nine power generators in such a way that running costs, emissions and a penalty function capturing constraint violations are all balanced according to a set of preferences. In their paper Hayes et al. treat DEED as a multi-agent reinforcement learning problem, as well as a MORL problem, which can create a number of complications not present in single-agent systems.

Bonus World is a relatively recent benchmark, introduced by Vamplew et al. as part of their research for "Softmax exploration strategies for multiobjective reinforcement learning" [27]. It features an agent moving around a grid. Certain squares of the grid can not be visited, while others can activate or deactivate a bonus state, that doubles the agents reward at the end of the episode. Depending on where the agent ends the episode, it gets a different reward for two objectives, with the time to complete an episode being a third objective.

Alongside Bonus World, Vamplew et al. also introduced the Space Exploration benchmark in their work [27]. Similar to Bonus World, Space Exploration involves an agent navigating a 2-dimensional grid. Agents can move to all 8 neighbouring squares, and must avoid asteroids and pockets of radiation, while navigating to one of several planets. The Space Exploration benchmark is a two-objective benchmark, where an agent must minimize its radiation exposure, while attempting to find the planet with the highest desirability.

Many of the benchmarks mentioned in previous paragraphs were implemented as part of the MORL-Glue benchmark suite [29] and can be found on GitHub[2].

2.2 Popularity Analysis

In Table 1 we show which publications used which discrete benchmarks to evaluate the performance of their methodologies. Unfortunately, due to time and space constraints, this table is far from exhaustive and several benchmarks were omitted (Pyramid MDP [32], Task-Oriented Dialog Policy Learning [38], Linked Rings [28], ...). Due to similar considerations, we also did not go into detailed descriptions of some benchmarks like the (Pressurized) Bountyful Sea Treasure [32]. Although some benchmarks were not included or discussed in detail, we feel like the selected subset is an accurate representation of the most commonly used MORL benchmarks.

From Table 2, we can see that, despite its relative simplicity, the DST benchmark is still the most widely used.

We would also like to note that many MORL techniques have not been tested on standardized benchmarks, but rather, were tested directly on the problem they intend to solve. This allows each algorithm's designers to solve the problem at hand and move on, but it does further complicate the process of assessing the merits of different MORL algorithms, and hampers general MORL research. Examples of this include MoTiAC [37], NPG-NAS [7] and MnasNet [25].

Another conclusion that can be drawn from this enumeration of MORL benchmark problems is the strong similarities between many of them. When comparing, for example, the Resource Gathering task to the Space Exploration task and the Bonus World task, as well as the Puddle World task and the DST task, it becomes apparent that all five of these tasks are variations of navigating a two-dimensional grid, with positive and negative rewards being given for certain grid squares. While the details such as the exact size of the state space, action space and Pareto front vary between these benchmarks, the underlying transition functions are largely the same.

2.3 Complexity Analysis

In section 2.1 we listed a number of MORL benchmarks, and looked at their popularity, noting which publications used which benchmarks to evaluate their methodologies. In this section, we will analyse the same set of benchmarks, based on their complexity. Each benchmark's complexity will be assessed in terms of the size of its state space, action space and the number of objectives that can be optimized for. While this is not an ideal measure of complexity for a RL benchmark, it is a complexity measure that can be easily quantified and compared. In this comparison, we also included 2 SORL benchmarks to allow readers to compare the complexity of MORL benchmarks to their SORL counterparts. The chosen reference benchmarks are `space-invaders-v0` and `space-invaders-ram-v0`. Both of these are games from the Atari 2600 benchmark, which contains a number of emulated games from the similarly named video game console. Space invaders is a game in which the player must shoot up at a number of enemies traversing the screen left-to-right, alternating direction and dropping down when they reach the edge of the screen. The players wins when all enemies are defeated, and loses if one of the enemies manages to reach the bottom of the screen. `space-invaders-ram-v0` uses the console's Random-Access Memory (RAM) as an observation, while `space-invaders-v0` allows the agent to observe the screen's pixels directly.

From the shapes of the observation spaces of various benchmarks in Table 2, it can be seen that image-based RL benchmarks have gained some popularity over the past couple of years. While this is certainly an option to raise the complexity of these benchmarks, it does not necessarily complicate the underlying task. It merely makes the process of feature extraction more challenging. Given the current state of the art in computer vision, and the pace at which it continues to advance, we would like to argue against the use of image-based observations to increase problem complexity,

[2] https://github.com/FedUni/MORL

	Abels et al. [1]	Barrett and Narayanan [2]	Basu [3]	Chen et al. [6]	Hayes et al. [10]	Horie et al. [12]	Mossalam et al. [19]	Vamplew et al. [27]	Van Moffaert et al. [30]	Van Moffaert et al. [31]	Van Moffaert and Nowé [32]	Wang et al. [34]	Wiering et al. [35]	Yang et al. [38]	Sum
MO-Super Mario Brothers [38]				✓[1]										✓	2
DEED [10]			✓		✓										2
Image-based DST [19]	✓						✓								2
MO-Puddleworld [26]															0
Bonus World [27]								✓							1
Space Exploration [27]								✓							1
Resource Gathering [2]		✓				✓			✓						3
Fruit Tree Navigation [38]														✓	1
Pressurized Bountyful Sea Treasure [32]											✓				1
DST [26]				✓			✓	✓		✓	✓	✓	✓	✓	8
MO-MountainCar [26]							✓			✓					2

Table 1: Table showing which publications tested their algorithms on which benchmarks, along with the total number of publications using each benchmark.

[1] Chen et al.'s version of MO-Super Mario Brothers only used two objectives.

Environment	Objectives	Obs. Shape	$\lvert\mathcal{S}\rvert$	Act. Shape	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{S}\times\mathcal{A}\rvert$	Known Pareto front
MO-Super Mario Brothers	5	$240 \times 256 \times 3$	4.72×10^7	1	7	3.30×10^8	
space-invaders-v0[4]	1	$210 \times 160 \times 3$	2.58×10^7	1	6	1.55×10^8	N/A
DEED	3	2	2.01×10^4	1	101	2.03×10^6	✓
Image-based DST	2	$10 \times 11 \times 3$	8.45×10^4	1	4	3.38×10^5	✓
space-invaders-ram-v0[4]	1	128×256	3.28×10^4	1	6	1.97×10^5	N/A
MO-Puddleworld	2	2	4.00×10^2	1	4	1.60×10^3	✓
Bonus World	3	2	1.62×10^2	1	4	6.48×10^2	✓
Space Exploration	2	2	6.50×10^1	1	8	5.20×10^2	✓
Resource Gathering	3	2	1.00×10^2	1	4	4.00×10^2	✓
Fruit Tree Navigation	6	2	1.27×10^2	1	2	2.54×10^2	✓
Pressurized Bountyful Sea Treasure	3	2	6.10×10^1	1	4	2.44×10^2	✓
DST	2	2	6.10×10^1	1	4	2.44×10^2	✓
MO-MountainCar	3	2	3.60×10^1	1	3	1.08×10^2	✓

Table 2: Comparison of state and action space sizes for several single- and multi-objective RL benchmarks. Benchmarks are ordered in descending order of complexity. SORL benchmarks are written in italics.

since advancements in computer vision are likely to render these complexity increases redundant in a number of years, when techniques like Vision Transformer (ViT) [9] become more mainstream and accessible.

Table 2 shows that there do exist MORL benchmarks that rival contemporary SORL benchmarks in terms of complexity. However, some of these come with some complications. Most notably, the DEED benchmark is a multi-agent problem, which can introduce a number of added complexities that only serve to complicate single-agent MORL research. While the image-based DST benchmark just about matches the `space-invaders-ram-v0`, even this is considered fairly simple by current SORL standards. We also note that our comparison ignores various pre-processing techniques often used in SORL publications targetting image-based benchmarks like frame-stacking, frame-skipping and grayscale conversions. While these could be applied to all image-based benchmarks, we focus on the original problems, rather than the pre-processed ones.

2.4 Availability Comparison

In this section, we will look at the various benchmarks discussed in previous sections, and see for which of these benchmarks (open-source) implementations are available. The importance of the availability of the source code for benchmarks has already been highlighted in [11]. We believe that Table 1 further shows the importance of this, since it is clear that, even 10 years after Vamplew et al.'s original paper pushing for standardized benchmarks [26], there is still little standardization among MORL users.

When it comes to benchmarking RL algorithms, a commonly used Application Programming Interface (API) is that of OpenAI's gym [5]. Gym proposes a simple software API that encapsulates the underlying MDP. Through this API, gym allows for algorithms to be easily tested on multiple different problems to show an algorithms' capability to generalize to different problems. While gym's API has been adopted by the vast majority of SORL publications, this is not necessarily the case for MORL publications, which further hampers easy comparisons between algorithms and benchmarks.

Since DST is one of the most commonly used benchmarks, there are a couple of implementations already available. Van Moffaert and Nowé provide an implementation of the DST problem as part of their paper on Pareto-Q Learning [32]. Their implementation[3] is written in Python, and compliant with the gym interface. They also provide an alternative variant of the DST problem, the Bountyful Sea Treasure variant [30].

Another implementation is that of Nguyen et al. [20], as part of their Multi-Objective Deep Reinforcement Learning (MODRL) framework. Their framework[4] contains both problems and solution strategies, and is written in Python. The DST environment is not fully compatible with the OpenAI gym interface (The implementation is missing certain attributes such as `action_space` and `observation_space`, and likely doesn't implement all functionality present in a regular gym environment, since it does not inherit from `gym.Env` class or any of its subclasses). While the API doesn't match that of gym, it does present an interface that is similar enough to allow research to rapidly adapt it to a gym-compatible environment. This implementation is much more customizable, and while it doesn't allow for the specification of arbitrary Pareto fronts, it does have the built-in capability to present a convex, concave, linear and mixed (convex and concave) Pareto front.

In Table 3 we list all the investigated benchmarks, and compare them based on the availability of an (open-source) implementation. We consider implementations in multiple programming languages, but note that the vast majority of RL research currently is performed in Python, thus, any implementation that doesn't use Python poses a significant disadvantage for anyone hoping to use it. In some cases, converting a SORL benchmark to a MORL benchmark is a reasonable option, because of this, we list both SORL and MORL implementations of the discussed benchmarks.

Table 3 notes a lack of DEED implementations. While Basu [3] provided all necessary data and formulae in their paper, they did not provide any implementation of this problem, and neither did Hayes et al. [10].

[3] https://gitlab.ai.vub.ac.be/mreymond/deep-sea-treasure
[4] https://personal-sites.deakin.edu.au/ thanhthi/drl.htm

3 PROPOSED BENCHMARK

Following the comparisons from section 2.1, 2.3 and 2.4, and some of the conclusions from [26] and [11] we now propose a new benchmark. The aim of this new benchmark is to fill the complexity gap between image-based DST environments, and the MO-Super Mario Brothers environment for single-agent MORL techniques. We will aim to achieve this complexity, not by increasing the complexity of feature extraction, but by increasing the size of the Pareto front, and making the underlying dynamics somewhat more complex. We will also be making this benchmark available for other researchers, hoping to help improve the state of benchmarking in MORL.
Similar to the Pressurized, Bountyful Deep Sea Treasure [32], our version of the DST problem involves the optimization of three conflicting objectives: Time, Treasure and Fuel. The observation and action space for our benchmark are also larger than the original DST problem. Similar to the DST problem, we define this problem to be a finite-horizon MDP, limited to 1000 time steps.

3.1 State Space

The state space of this new DST problem is significantly larger than the original one [26]. The new state space consists of eleven two-element column vectors of integers:

$$\mathcal{S} \in \mathbb{Z}^{2\times 11} \tag{3}$$

The first column vector represents the agent's current velocity, expressed as a separate x- and y-component. The following ten column vectors each represent the agent's relative coordinates to each of the treasures. This observation differs from the original DST problem in that it directly tells the agent where each of the potential solutions are. By changing the formulation of the observation space, the agent's task changes from memorizing the treasure locations and a path to them, to learning how to traverse the ocean towards whatever treasure the agent finds most interesting.

In the standard DST environment, we know that there are 61 positions the agent can visit. In the proposed environment, the agent's velocity can not exceed 5 in any direction, meaning that there are a total of 121 different velocity vectors the agent can achieve ($(5 + 1 + 5)^2$). While not every velocity vector is achievable in every position, this still allows us to place an upper bound on the size of our state space of 7381, which is already significantly larger than the original DST problem, which had a state space of 60 elements. We note that our implementation also allows users to define arbitrarily large Pareto fronts, to further increase the problem complexity.

3.2 Action Space

The action space consists of a two-element vector. The elements of this vector represent the respective x- and y-components of the acceleration the agent would like to make. In the new DST environment, the agent no longer takes discrete steps in one of the cardinal directions, but rather, the agent can accelerate in one or both dimensions, allowing for more efficient ways of reaching each treasure at the cost of a higher fuel consumption. The use of accelerations, rather than unit velocity in cardinal directions also increases the complexity of the underlying dynamics, since agents must now account for built-up inertia. An agent can accelerate up

Benchmark	SORL Implementations	MORL Implementations
MO-Super Mario Brothers	[16]	[38]
DEED		
Image-based DST		[19][1]
MO-Puddleworld	[14]	[15][2]
Bonus World		[29][5]
Space Exploration		[29][5]
Resource Gathering		[29][5]
Pressurized Bountyful Sea Treasure[4]		
DST		[15][2][19][1][20][3][32][38] [29][5]
Fruit Tree Navigation		[38]
MO-MountainCar	[5]	[15][2][20][3][29][5]

Table 3: Comparison of the different benchmarks, based on the availability of open-source implementations

[1] Mossalam et al.'s implementation of these benchmarks is written in Lua.
[2] Issabekov et al.'s implementation of these benchmarks is written in in Java.
[3] Nguyen et al.'s implementation of these benchmarks is written in Python, but does not adhere to the gym API.
[4] Van Moffaert and Nowé do provide an implementation of the similar Bountyful Sea Treasure [32]
[5] Part of the MORL-Glue Benchmark Suite, written in Java

to three units in any direction yielding a total size of our action space of 49 ($(3+1+3)^2$).

3.3 Dynamics

At a high-level, the dynamics of the new MDP are similar to that of the original [26]: If the agent attempts to make a move that would result in a collision, the state is left unchanged, otherwise, the agent's desired action is executed. We will only show the dynamics function for the x-component for brevity, noting that a set of identical operations are executed for the y-component. We start off by defining our action a:

$$a \doteq (a_x, a_y) \tag{4}$$

Next, we determine a preliminary next velocity and position, which will be used for collision checking.

$$v'_{t+1,x} = v_{t,x} + a_x \tag{5}$$

$$x'_{t+1} = x_t + v'_{t+1,x} \tag{6}$$

Knowing the position of the submarine if the action was executed, we can check for collisions, and update our actual velocity and position accordingly. We also set a maximum velocity for the submarine, v_{max}, that it can never exceed.

$$v_{t+1,x} = \begin{cases} min\left(v_{max}, v_{t,x} + a_x\right) & \text{if } collides\left(x'_{t+1}\right) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$x_{t+1} = x_t + v_{t+1,x} \tag{8}$$

We note that in this case, the $collides()$ function should not only check for collisions in the cardinal directions, but also in diagonal directions. Another important note is that the agent's velocity is reduced to zero if an action would cause a collision, which is a behaviour an agent could attempt to exploit to arrest movement without consuming fuel.

3.4 Rewards

In terms of rewards, the new DST problem inherits the two objectives from the old DST problem (time, r_b and treasure, r_p), and adds a third objective, fuel. The fuel objective was designed in such a way that it conflicts with the two already existing objectives. When the agent attempts to find far-away treasures, it will need to spend more fuel to cover the distance, and when the agent wants to cover a given distance quicker, it can spend more fuel accelerating and decelerating.

Formally, we define the fuel objective r_f as the negative sum of the squares of the acceleration in both dimensions, if the action does not cause a collision, or 0 if the action would cause a collision:

$$r_f = \begin{cases} -\left(a_x^2 + a_y^2\right) & \text{if } collides\left(x'_{t+1}\right) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

We note that this function creates the exploit mentioned earlier in section 3.3. The formulation of this reward function also does not necessarily make collisions sub-optimal, since coasting (Moving without accelerating using built-up inertia) uses the same amount of fuel as colliding.

This leads us to $\boldsymbol{R}(s, a, s')$ as:

$$\boldsymbol{R}(s, a, s') = \left(r_b(s, a, s'), r_p(s, a, s'), r_f(s, a, s')\right) \tag{10}$$

3.5 Complexity Analysis

Knowing the observation and state space for our newly proposed benchmark, we can now perform the same complexity calculations that we did for other benchmarks. In this complexity comparison, we will compare two configurations of the proposed benchmark,

the most simple configuration, using the values from section 3.1 and 3.2, and a much more complicated configuration, that rivals the MO-Super Mario Brothers benchmark in complexity. To allow for an easier comparison, we will repeat the two closest, single-agent MORL benchmarks here from Table 2.

The first configuration (Configuration A) we propose is the one already touched on in section 3.1 and 3.2. It involves the same set of treasures as the original DST environment, with a maximum velocity of 5 in any direction and a maximum acceleration of 3 in any direction. The total size of the state-action space is 3.62×10^5, which is comparable to the image-based DST's complexity of 3.38×10^5.

The next configuration that we propose (Configuration B) is significantly more complex than the first. It requires adding 48 treasures to the existing Pareto front, for a total of 58. This significantly increases the number of states available to the agent, as well as the size of the observation. In order to estimate the total size of the state space, we assume treasures are added at linearly increasing depths for these 48 new treasures. Next, we also allow the agent to accelerate more rapidly, allowing it to cover the larger space much quicker. The agent is allowed to accelerate up to 10 units in any direction, and can accumulate up to 12 units worth of velocity. In its totality, this configuration allows us to visit 1231 spaces, with a total of 625 $((12 + 1 + 12)^2)$ different velocities, leading to a total state space size of 7.69×10^5. On top of that, our agent can accelerate in 441 $((10 + 1 + 10)^2)$ different ways. This leaves us with a total combined state-action space size of 3.39×10^8, which slightly exceeds that of MO-Super Mario Brothers (3.30×10^8).

4 IMPLEMENTATION

We provide an implementation of both the proposed benchmark, and the original DST problem [26], written in Python[5] and compatible with gym [5]. The benchmark was also published as a PyPI package[6].

4.1 Pareto Front

In this section, we provide the Pareto front for the proposed benchmark, using Configuration A from section 3.5. While we do not discuss it here, the Pareto front data for the original DST benchmark is also available in our repository. The Pareto front of the proposed benchmark was determined empirically, using exhaustive graph search. Solving this environment took 33.22 hours using 28 Intel E5-2680v4 Broadwell CPUs. The numerical data for the Pareto front can be found in the same repository that contains the code for the environment. Figure 1 shows the obtained Pareto front for the new benchmark. It is important to note that one optimal solution was omitted from this Pareto front for clarity. This solution involved the agent idling for the complete duration of the episode, in order to minimize fuel consumption, it is however present in the numerical dataset in the repository. Each point on the Pareto front has been projected onto the time-treasure plane for clarity, and horizontal lines have been drawn for all solutions that lead to the same treasure.

Looking at Figure 1, the first notable property of this Pareto front is that the number of points on the Pareto front (25 + 1) no longer

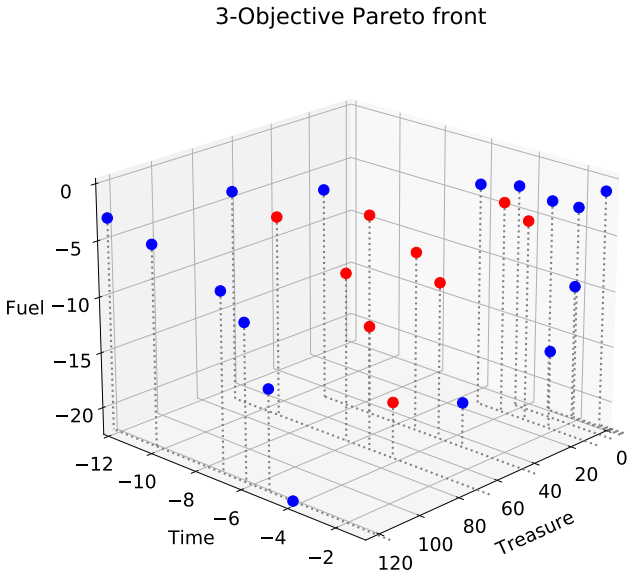

Figure 1: 3-Objective Pareto front. Points in blue lie on the convex hull, while points in red are contained inside the convex hull.

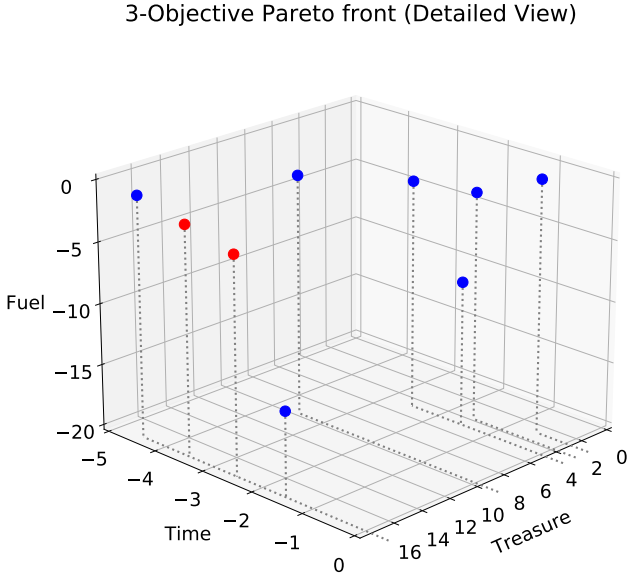

Figure 2: Detailed view of the low-treasure region of the 3-Objective Pareto front. Points in blue lie on the convex hull, while points in red are contained inside the convex hull.

corresponds to the number of treasures (10). In our analysis, we will ignore the 26th point generated by the submarine performing the "idle" action 1000 times. While this point is theoretically optimal, we argue that it does not produce a useful solution, and hence will

[5]https://github.com/imec-idlab/deep-sea-treasure
[6]https://pypi.org/project/deep-sea-treasure/

Environment	Objectives	Obs. Shape	$\lvert\mathcal{S}\rvert$	Act. Shape	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{S}\times\mathcal{A}\rvert$	Known Pareto front
Configuration B	3	2×58	7.69×10^5	2	441	3.39×10^8	
MO-Super Mario Brothers	5	$240 \times 256 \times 3$	4.72×10^7	7	7	3.30×10^8	
Configuration A	3	2×11	7.38×10^3	2	49	3.62×10^5	✓
Image-based DST	2	$10 \times 11 \times 3$	8.45×10^4	1	4	3.38×10^5	✓

Table 4: Comparison of the proposed state and action space sizes against two MORL benchmarks. Benchmarks are ordered in descending order of complexity. Proposed benchmarks are marked in bold.

disregard it in further analysis. We can see that for each treasure, there exist different trade-offs between fuel and time. Note, that not all treasures are part of the Pareto front, the treasures that make up the Pareto front are those with values $\{1, 2, 3, 8, 16, 50, 74, 124\}$, while the treasures with values $\{5, 24\}$ are not part of the Pareto front, as can be seen in the detailed view of the Pareto front in Figure 2.

By looking at Figure 1 it can also be easily ascertained that the Pareto front for this environment is not convex. In this Pareto front, 9 out of 25 solutions lie inside the convex hull, rather than on it. In the original DST problem, 3 out of 10 points formed local concavities. Similar to the original DST problem, this presents an interesting challenge for MORL algorithms, testing their ability to handle non-convex Pareto fronts.

4.2 Guidelines

While the implementation is highly configurable, we recommend anyone using our implementation, to always at least report performance using the default settings (Configuration A), to ensure that the results of a specific algorithm can be fairly compared to those of other algorithms. The implementation and Pareto front dataset provided by the authors is independently citable through Zenodo[7].

5 DISCUSSION & FUTURE WORK

One major omission in this paper is the concept of constraints. Constraints in the context of MOO are usually formulated as a set of (in)equalities that check certain properties of a solution. Constraints pose an interesting challenge to the API proposed by gym [5], since the current API provides no natural way of indicating that certain solutions are “unacceptable" or “invalid". While it is usually possible to work around this – e.g., by ending episodes early, and modifying reward functions to make constraint violations guaranteed to be sub-optimal – this is not always easy to achieve, and prompts the question of whether or not gym provides the most well-suited API for solving constrained optimization problems using RL agents. One such work-around is used by `safety-gym`, introduced in OpenAI’s 2019 whitepaper [21]. Safety gym is completely API-compatible with regular gym environments and simply passes constraint information through the `info` dictionary that is returned by an environment’s `step()` method.

Besides omitting constraints, we also chose to omit continuous benchmark problems from this paper. While there are numerous continuous, MORL problems that represent interesting challenges to state-of-the-art MORL research, such as the Minecart problem [1], or multi-objective versions of numerous Mujoco benchmarks [36], we were unable to sufficiently address these within the confines of this paper.

Another potential point of improvement, whose benefits could extend beyond just the field of MORL, is better methodologies for examining and comparing the complexity of benchmarks. In this paper, the size of the state-action space was used as a measure of problem complexity, but one of the most important, and often complex parts of any MDP, its transition function, is completely ignored in this. We believe that, in order to be able to do a fair comparison of RL benchmarks, some way of capturing the complexity of transition functions is a requirement.

While this paper examined individual benchmarks, creating a larger benchmark suite, composed of several different benchmarks could help circumvent the weaknesses of individual benchmarks. Similar efforts have already been made in the field of MOO [8] [13] [40] and SORL [4] [5]. After assembling such a benchmark suite, MORL algorithms could be tested on a range of standardized test problems, providing researchers with a more accurate way to evaluate the performance of their algorithms. One possible starting point for such a benchmark suite could be MORL-Glue [29]. This includes a number of MORL benchmarks that cover a broad range of problems. MORL-Glue is written in Java, which can hamper adoption; however, it is likely that a similar benchmark suite could be created in Python using (some of) the implementations listed in table 3.

In this paper, a number of MORL benchmarks were compared. Based on this comparison, a new MORL benchmark was proposed, and an implementation was provided. The newly proposed benchmark was compared to existing benchmarks, showing that it filled a need for medium to high-complexity benchmarks.

ACKNOWLEDGMENTS

This research received funding from the Flemish Government (AI Research Program). This work was supported by the Research Foundation Flanders (FWO) under Grant Number 1SC8821N. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government – department EWI.

REFERENCES

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 11–20. https://proceedings.mlr.press/v97/abels19a.html

[2] Leon Barrett and Srini Narayanan. 2008. Learning All Optimal Policies with Multiple Criteria. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML ’08)*. Association for Computing Machinery, New York, NY, USA, 41–47. https://doi.org/10.1145/1390156.1390162

[7] https://doi.org/10.5281/zenodo.5227091

[3] M. Basu. 2008. Dynamic economic emission dispatch using nondominated sorting genetic algorithm-II. *International Journal of Electrical Power Energy Systems* 30, 2 (2008), 140–149. https://doi.org/10.1016/j.ijepes.2007.06.009

[4] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2012. The Arcade Learning Environment: An Evaluation Platform for General Agents. *CoRR* abs/1207.4708 (2012). arXiv:1207.4708 http://arxiv.org/abs/1207.4708

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersoson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). http://arxiv.org/abs/1606.01540

[6] Diqi Chen, Yizhou Wang, and Wen Gao. 2020. Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. *Applied Intelligence* 50, 10 (01 Oct 2020), 3301–3317. https://doi.org/10.1007/s10489-020-01702-7

[7] Zewei Chen, Fengwei Zhou, George Trimponias, and Zhenguo Li. 2020. Multi-objective neural architecture search via non-stationary policy gradient. *arXiv preprint arXiv:2001.08437* (2020).

[8] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2001. Scalable Test Problems for Evolutionary Multiobjective Optimization. *Evolutionary Multiobjective Optimization* (2001), 105 – 145. https://doi.org/10.1007/1-84628-137-7_6

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

[10] Conor F. Hayes, Enda Howley, and Patrick Mannion. 2020. Dynamic Thresholded Lexicographic Ordering. (2020). https://www.researchgate.net/profile/Conor-Hayes/publication/348779254_Dynamic_Thresholded_Lexicographic_Ordering/links/6010134a299bf14088c0f595/Dynamic-Thresholded-Lexicographic-Ordering.pdf

[11] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2021. A Practical Guide to Multi-Objective Reinforcement Learning and Planning. arXiv:2103.09568 [cs.AI]

[12] Naoto Horie, Tohgoroh Matsui, Koichi Moriyama, Atsuko Mutoh, and Nobuhiro Inuzuka. 2019. Multi-objective safe reinforcement learning: the relationship between multi-objective reinforcement learning and safe reinforcement learning. *Artificial Life and Robotics* 24, 3 (01 Sep 2019), 352–359. https://doi.org/10.1007/s10015-019-00523-3

[13] Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. 2005. A Scalable Multi-Objective Test Problem Toolkit. *Evolutionary Multi-Criterion Optimization* (2005), 280 – 295. https://doi.org/10.1007/978-3-540-31880-4_20

[14] Ehsan Imani. 2020. gym-puddle. https://github.com/EhsanEI/gym-puddle, http://web.archive.org/web/20201204203830/https://github.com/EhsanEI/gym-puddle.

[15] Rustam Issabekov and Peter Vamplew. 2012. An Empirical Comparison of Two Common Multiobjective Reinforcement Learning Algorithms. In *AI 2012: Advances in Artificial Intelligence*, Michael Thielscher and Dongmo Zhang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 626–636.

[16] Christian Kauten. 2018. Super Mario Bros for OpenAI Gym. GitHub. https://github.com/Kautenja/gym-super-mario-bros

[17] Mohammed A. Khamis and Walid Homaa. 2014. Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Engineering Applications of Artificial Intelligence* 29 (2014), 134 – 151. https://doi.org/10.1016/j.engappai.2014.01.007

[18] Sandeep U. Mane and M. R. Narasinga Rao. 2017. Many-Objective Optimization: Problems and Evolutionary Algorithms – A Short Review. *International Journal of Applied Engineering Research* 12, 20 (2017), 9774 – 9793. https://www.ripublication.com/ijaer17/ijaerv12n20_72.pdf

[19] Hossam Mossalam, Yannis M. Assael, Diederik M. Roijers, and Shimon Whiteson. 2016. Multi-Objective Deep Reinforcement Learning. arXiv:1610.02707

[20] Thanh Thi Nguyen, Ngoc Duy Nguyen, Peter Vamplew, Saeid Nahavandi, Richard Dazeley, and Chee Peng Lim. 2018. A Multi-Objective Deep Reinforcement Learning Framework. *CoRR* abs/1803.02965 (2018). http://arxiv.org/abs/1803.02965

[21] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. (2019).

[22] D. M. Roijers, Vamplew P., Whiteson S., and Dazeley R. 2013. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research* 48 (18 Oct 2013). https://doi.org/10.1613/jair.3987

[23] Roxana Rădulescu, Patrick Mannion, Diederik M. Roijers, and Ann Nowé. 2019. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems* 34 (2019). https://doi.org/10.1007/s10458-019-09433-x

[24] Dhish Kumar Saxena, Tapabrata Ray, Kalyanmoy Deb, and Ashutosh Tiwari. 2009. Constrained many-objective optimization: A way forward. In *2009 IEEE Congress on Evolutionary Computation*. 545–552. https://doi.org/10.1109/CEC.2009.4982993

[25] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. 2019. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2815–2823. https://doi.org/10.1109/CVPR.2019.00293

[26] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning* 84 (2011), 51 – 80. https://doi.org/10.1007/s10994-010-5232-5

[27] Peter Vamplew, Richard Dazeley, and Cameron Foale. 2017. Softmax exploration strategies for multiobjective reinforcement learning. *Neurocomputing* 263 (2017), 74–86. https://doi.org/10.1016/j.neucom.2016.09.141 Multiobjective Reinforcement Learning: Theory and Applications.

[28] Peter Vamplew, Rustam Issabekov, Richard Dazeley, Cameron Foale, Adam Berry, Tim Moore, and Douglas Creighton. 2017. Steering approaches to Pareto-optimal multiobjective reinforcement learning. *Neurocomputing* 263 (2017), 26–38. https://doi.org/10.1016/j.neucom.2016.08.152 Multiobjective Reinforcement Learning: Theory and Applications.

[29] Peter Vamplew, Dean Webb, Luisa M. Zintgraf, Diederik M. Roijers, Richard Dazeley, Rustam Issabekov, and Evan Dekker. 2017. MORL-Glue: A Benchmark Suite for Multi-Objective Reinforcement Learning. In *BNAIC 2017 Preproceedings*. 389 – 390. https://pure.rug.nl/ws/portalfiles/portal/107483223/preproceedings_1_.pdf#page=405

[30] Kristof Van Moffaert, Tim Brys, and Ann Nowé. 2014. *Efficient Weight Space Search in Multi-Objective Reinforcement Learning*. Technical Report. Vrije Universiteit Brussel. https://www.researchgate.net/publication/261499155_Efficient_Weight_Space_Search_in_Multi-Objective_Reinforcement_Learning

[31] Kristof Van Moffaert, Madalina M. Drugan, and Ann Nowé. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 191–199. https://doi.org/10.1109/ADPRL.2013.6615007

[32] Kristof Van Moffaert and Ann Nowé. 2014. Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies. *Journal of Machine Learning Research* 15 (2014), 3663 – 3692. https://jmlr.org/papers/v15/vanmoffaert14a.html

[33] Handing Wang, Licheng Jiao, and Xin Yao. 2015. Two_Arch2: An Improved Two-Archive Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 19, 4 (2015), 524–541. https://doi.org/10.1109/TEVC.2014.2350987

[34] Weijia Wang and Michèle Sebag. 2012. Multi-objective Monte-Carlo Tree Search. In *Proceedings of the Asian Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 25)*, Steven C. H. Hoi and Wray Buntine (Eds.). PMLR, Singapore Management University, Singapore, 507–522. https://proceedings.mlr.press/v25/wang12b.html

[35] Marco A. Wiering, Maikel Withagen, and Mădălina M Drugan. 2014. Model-based multi-objective reinforcement learning. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 1–6. https://doi.org/10.1109/ADPRL.2014.7010622

[36] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. 2020. Prediction-Guided Multi-Objective Reinforcement Learning for Continuous Robot Control. In *Proceedings of the 37th International Conference on Machine Learning*.

[37] Chaoqi Yang, Junwei Lu, Xiaofeng Gao, Haishan Liu, Qiong Chen, Gongshen Liu, and Guihai Chen. 2020. MoTiAC: Multi-Objective Actor-Critics for Real-Time Bidding. *CoRR* abs/2002.07408 (2020). arXiv:2002.07408 https://arxiv.org/abs/2002.07408

[38] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4a46fbfca3f1465a27b210f4bdfe6ab3-Paper.pdf

[39] Shengxiang Yang, Miqing Li, Xiaohui Liu, and Jinhua Zheng. 2013. A Grid-Based Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 17, 5 (2013), 721–736. https://doi.org/10.1109/TEVC.2012.2227145

[40] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computing* 8 (2000), 173 – 195. https://doi.org/10.1162/106365600568202