Average Reward Adjusted Discounted Reinforcement Learning

Manuel Schneckenreither
University of Innsbruck
Innsbruck, Austria
manuel.schneckenreither@student.uibk.ac.at

Georg Moser
University of Innsbruck
Innsbruck, Austria
georg.moser@uibk.ac.at

ABSTRACT

Despite astonishing results in the application of reinforcement learning to game-like domains, for instance chess, Go and various Atari games, to the best of our knowledge no such success has been reported for non-episodic operations research problems yet, despite their ubiquitous presence in real-world applications. In this paper, we introduce a novel adaptation of the discounted reinforcement method, dubbed *average reward adjusted discounted reinforcement learning* (*ARAL* in short) as remedy. Our approach is based on the Laurent Series expansion of the discounted state value and a subsequent reformulation of the target function guiding the learning process. We compare ARAL to competing approaches and give ample evidence of its usefulness.

KEYWORDS

machine learning, reinforcement learning, average reward learning, operations research, applications.

1 INTRODUCTION

Reinforcement Learning (*RL*) is most often applied to control problems, games and other sequential decision making tasks [28], where in almost all steps the reward function returns 0. However, in real-world (economic) applications, where success is usually measured in terms of profit or costs, the function intuitively returns a non-zero reward in almost all steps as it continuously reports the actions results. In this work, we focus on such problems that produce an average reward per step that cannot be approximated by 0. We introduce *ARAL*, a novel adaption of the discounted reinforcement method and give ample example of the usefulness of ARAL in this context. Application areas of such a model could be, e.g. (i) an agent that periodically decides on buying and selling instruments at the stock market; (ii) an agent performing daily replenishment decisions; or (iii) many other decision problems that arise in hierarchical supply chain and production planning systems, which are strongly capacity-oriented [33] and with aggregated feedback (see for example [10, 24]).

The reason for the small number of RL applications to non-episodic operations research problems, i.e. our original field of motivation, and the fact that none of the available applications report tremendous success, as in game-like areas, can be explained by the average reward (per step) that the agent receives. The average reward is usually by far the greatest part, when the discounted state values are decomposed into its sub-components. This results from the fact, that in contrast to the other parts, the average reward contributes to the learned state-values in an exponentially up-scaled fashion. However, for most applications the average reward is equal among all states and thus the agent has issues in choosing the best action. Additionally, iteratively shifting all state values to the exponentially up-scaled average reward easily causes the learning process to fail, finding itself in sub-optimal maxima and with that complicating the hyperparameterisation process tremendously.

We argue that this is the primal reason that discounted reinforcement learning is inappropriate for these kind of applications. Earlier this has already been emphasised by Mahadevan. In a series of papers, Mahadevan investigate *average reward reinforcement learning*, cf. [13–17]. Our contribution aims to combine discounted and average reward RL, to get the best things of both worlds: the stability and simplicity of standard discounted RL and the idea of assessing the average reward separately and aiming for broader optimality criteria of average reward RL. Additionally, we contribute to the operations research domain by providing a new machine learning algorithm and applying it to a well studied M/M/1 admission control queuing system. In summary, we make the following contributions in this paper. (i) First we analyse and illustrate why standard discounted reinforcement learning is inappropriate for real-world applications, which often impose an average reward that cannot be approximated with 0, (ii) Second, we establish a novel near-Blackwell-optimal reinforcement learning algorithm and analytically prove its optimality, and (iii) Third, show the viability of the algorithm by experimentally applying it, where, due to lack of space, we focus on the well-studied M/M/1 admission control queuing system.

Related Work. Only a very limited number of operations research optimisation papers that are using RL and imposing this structure are available. In [25], the first author and Haeussler present an RL algorithm which optimises order release decisions in production control environments. They use sophisticated additions to allow the agent to link the actions to the actual rewards. The results were compared to static order release measures only, which were outperformed. Gijsbrechts et al. [8] apply RL on a dual sourcing problem for replenishment of a distribution center using rail or road, and compare their results to optimal solutions if tractable or otherwise established heuristics. They found that hyperparameter tuning is effort-intensive and the resulting policies are often not optimal, especially for larger problems. Balaji et al. [1] use out-of-the-box RL algorithms with simple 2 layer neural networks to tackle *stochastic bin packing*, *newsvendor* and *vehicle routing problems* (*VRP*). VRP is a generalised *travelling salesperson problem* (*TSP*), where one or more vehicles have to visit nodes in a graph. They report to sometimes beat the benchmarks and find sensible solutions. The capacitated VRP is also tackled with RL by Nazari et al. [21]. They minimise the total route length and compare the results to optimal solutions for small instances. Although the optimum is not reached, one instance, which keeps track of the most probable paths, performs better than

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.

well established heuristics. Also for larger problem sizes this technique seems to outperforms the other tested methods. Vera and Abad [31] extend this model to a multi-agent algorithm and thus tackle the capacitated multi-vehicle routing problem with a fixed fleet size. They also report better results compared to the heuristics, especially for large problem sizes, but in contrast to [21] are outperformed by Google's OR-Tools[1]. These applications, like other TSP applications [3, 12, e.g.], of RL to the VRP, except for [1], however, calculate the reward according to the length to the finished route. Thus, the average reward for long routes can be approximated with 0 as the decision problem is episodic and therefore perform well. Finally, there are applications to the beer distribution game [7, 22], but the problem sizes and thus complexities are small.

All of the above cited papers use highly sophisticated methods, but essentially remain routed in standard discounted RL frameworks. Further, only small problem sizes are handled or far-from-optimal solutions are reported. A different approach was taken by Mahadevan. In a series of papers the authors investigate average reward RL, cf. [13–17]. We emphasise that in average reward RL no discount factor is used but rather nested constraint problems are approximated iteratively. The latter immediately leads to computational intensive calculations.

The rest of the paper is structured as follows. The next section provides a high-level overview of ARAL. Section 4 introduces discounted RL and average reward RL, and provides the linkage between the two frameworks via the Laurent Series Expansion of discounted state values. The insights gained of the expansion form the foundations for the developed average reward adjusted discounted RL method established in Section 5. Section 7 proves the viability of the algorithm, while Section 8 concludes the paper.

2 OVERVIEW OF ARAL

The overarching goal of businesses is an increase of value. One key factor for reaching this goal is the maximisation of profit [2, p.18]. Accordingly, the most intuitive decision taking in economic optimisations problems is to choose decisions that return the highest expected long-term profit at every decision point. Translated to RL this means that the most intuitive reward function quotes the accumulated profit of the last decision period. However, most RL algorithms [e.g. 19, 20, 27] are not designed to cope with such reward functions and therefore perform poorly on such problems.

Therefore, our approach estimates the average reward value and uses *average reward adjusted discounted state values* to be able to selectively choose the best actions. Best refers to collecting as much reward as possible, and only afterwards, to choose paths that collect the reward earlier. The Markov Decision Process (MDP) in Figure 1 consists of two possible definite policies, with reward of 2 for going left and 0 for moving right. Both policies π_A (going left in 1) and π_B (going right) result in an the average reward of 1. However, only π_A is optimal, as the actions with non-zero rewards are selected earlier. To clarify consider starting in state 1. Under the policy π_A the reward sequence is $(2, 0, 2, 0, \ldots)$, while for π_B it is $(0, 2, 0, 2, \ldots)$. Hence, simply learning the *average reward* is insufficient. Basically, there are two more values to consider. First, the *bias values*, which intuitively is the additional reward

[1]See https://developers.google.com/optimization.

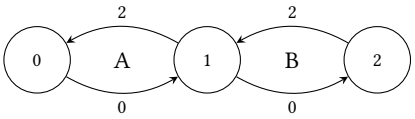

Figure 1: A MDP with a single choice (adapted from [14])

received when starting in a specific state. And second, the *error term*, which incorporates the number of steps until the reward is collected. The analytically inferred bias values under policy π_A are $V^{\pi_A}((A, \text{left})) = V^{\pi_A}((A, \text{right})) = 0.5$. For policy π_B these amount to $V^{\pi_B}((A, \text{left})) = V^{\pi_B}((A, \text{right})) = -0.5$. Therefore, π_A is preferable.

Our algorithm infers i) the *average reward* based on the state values of consecutively observed states and the returned reward, ii) approximates the *bias values*, and iii) uses a smaller discount factor to infer the *error term* values. The latter only exist for discount factors strictly less than 1. With a discount factor of $0.999 \in (0, 1]$ our implementation automatically infers policy π_A in under $10k$ iterations. In particular the average reward adjusted discounted state values $X^{\pi_A}_{0.999}((A, \text{left})) = 0.493$ and $X^{\pi_A}_{0.999}((A, \text{right})) = 0.492$ are an estimate of the bias values. The bias values slightly differ from the analytical values due to the error term, as the discount factor is strictly less than 1. The third decision level is comprised of average adjusted discounted state values with a smaller discount factor. The inferred values for a discount factor of 0.8 are $X^{\pi_A}_{0.8}((A, \text{left})) = 0.555$ and $X^{\pi_A}_{0.8}((A, \text{right})) = 0.145$. Thus, action left is preferred over action right.

3 PRELIMINARIES

Like Miller and Veinott [18] we are concerned with problems that are observed in a sequence of points in time labeled $0, 1, 2, \ldots$ and can be modelled using a finite set of states $\mathcal{S}$, labelled $1, 2, \ldots, |\mathcal{S}|$. At each point t in time the system is in a state $s_t \in \mathcal{S}$ and by choosing an action a_t, of a finite set of possible actions $\mathcal{A}_s$, the system returns a reward $r_t = r(s_t, a_t)$ and transitions to another state $s_{t+1} \in \mathcal{S}$ at time $t + 1$ with conditional probability $p(s_{t+1}, r_t \mid s_t, a_t)$. Hence, we expect the system to possess the Markov property [28, p.63]. RL processes that possess the Markov property are referred to as Markov decision processes (MDPs) [28, p.66]. *Terminal states* are absorbing states and are followed by a reset of the system, which starts a new episode [28, p.58]. *Recurrent states* are revisited with probability 1 [14]. States that are not recurrent are called *transient*. For *periodic states* the greatest common divisor of all path lengths to itself is greater than 1. A state that is not periodic is termed *aperiodic*. The *action space* is defined as $F = \times_{s=1}^{|\mathcal{S}|} \mathcal{A}_s$. A *policy* is a sequence $\pi_\gamma = (f_0, f_1, f_2, \ldots)$ of elements $f_t \in F$, where γ is the discount factor. Using the policy π_γ means that, if the system is in state s at time t, the action $f_t(s)$, i.e. the s-th component of f_t, is chosen. As the optimal policy $\pi^\star$ for a given MDP is different to the optimal achievable policy $\pi^\star_\gamma$ with chosen discount factor γ, we rigorously write π_γ in this paper. A *stationary policy* $\pi_\gamma = (f, f, \ldots)$ does not depend on time. In the sequel we are concerned with stationary policies only. An *episodic MDP* includes terminal states, while *non-episodic MDPs* do not [28, p.58]. An *ergodic MDP* consists of a single

set of recurrent states under all stationary policies, that is, all states are revisited with probability 1 [14]. A MDP is termed *unichain* if under all stationary policies the transition matrix contains a single set of recurrent states and (a possible) empty set of transient states. For *multichain MDPs* there exists a policy with at least two recurrent classes [14]. The aim in RL is to find the optimal stationary policy $\pi^\star$ of the underlying MDP.

4 THE LAURENT SERIES EXPANSION OF DISCOUNTED STATE VALUES

This section investigates the discounted RL framework and provides the Laurent series expansion of its state values, which play a crucial role in the development of the presented algorithm.

4.1 Discounted Reinforcement Learning

In standard discounted RL the value of a state is defined as the expected discounted sum of future rewards [e.g. 14, 28].

Definition 4.1. The discounted state value under policy π_γ and when starting in state s is defined as

$$V_\gamma^{\pi_\gamma}(s) = \lim_{N\to\infty} \mathbb{E}[\sum_{t=0}^{N-1} \gamma^t R_t^{\pi_\gamma}(s)] \ ,$$

with discount factor $0 \leqslant \gamma < 1$ and $R_t^{\pi_\gamma}(s) = \mathbb{E}_{\pi_\gamma}[r(s_t, a_t) \mid s_t = s, a_t = a]$ the reward received at time t by following policy π_γ.

The aim is to find an optimal policy $\pi_\gamma^\star$, which when followed, maximises the state value for all states s as compared to any other policy π_γ: $V_\gamma^{\pi_\gamma^\star} - V_\gamma^{\pi_\gamma} \geqslant 0$. This criteria is usually referred to as *discounted-optimality* (or γ-*optimality*) as the discount factor γ is fixed [15]. Note that most works omit the index γ in the policies $\pi_\gamma, \pi_\gamma^\star$ and thus incorrectly indicate that $\pi_\gamma^\star = \pi^\star$, where $\pi^\star$ is the optimal policy for the underlying problem.[2] We identified four major issues of standard discounted RL for MDPs with non-zero average reward per step.

(1) Standard discounted RL infers suboptimal policies due to the discount factor being strictly less than one, i.e. $\gamma < 1$.
(2) With standard discounted RL it is very difficult to specify a desired balance between short-term and long-term rewards.
(3) The average reward per step in the decomposed state value of terminal states is ignored as there are no future states. This leads to poor policies, even when converged.
(4) The average reward per step is independently assessed for each state, even though this part of the state value is shared between more than one or, for unichain MPDs, all states. This usually leads to an exponentially increased number of learning steps required for policies to converge.

In the sequel we briefly elaborate on these disadvantages.

Add 1. The (converged optimal) policy depends on the discount factor γ. Consider the MDP of the upper part in Figure 2. It returns reward 5, or 20 respectively, upon traversing to state 1 from state 5, or $10'$ resp. In all other cases the reward is 0. The average reward is 1 for the printer-loop and 2 for the mail-loop. Thus, the optimal policy is to choose the mail-loop, but if $\gamma < 3^{-\frac{1}{5}} \approx 0.8027$ a standard discounted RL agent selects the printer loop.

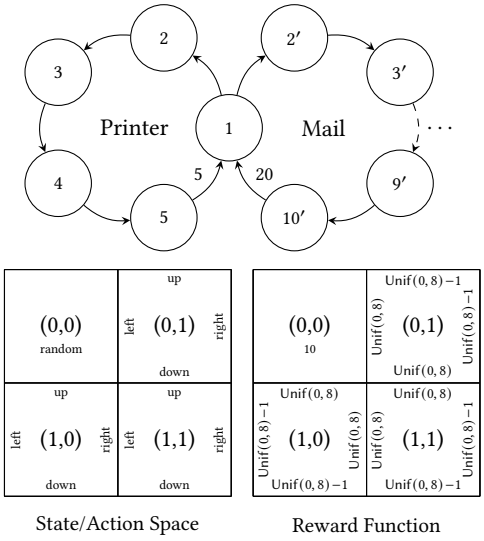

Figure 2: A MDP with one action choice (top, adapted[3], [15]) and a gridworld MDP with 4 states and 5 actions (bottom).

Add 2. As we will see only the long-term reward (average reward), out of three addends (bias value, error term), of the state-value is exponentially scaled. For non-zero average reward MDPs this is problematic, as average reward is multiplied by $1/(1-\gamma)$. To clarify, consider the gridworld MDP on the lower part of Figure 2 with states $\mathcal{S} = \{(x, y) \mid x, y \in \{0, 1\}\}$. $(0, 0)$ is the goal state, after which the agent is placed uniformly on the grid by using the action random (see left side). In all other states the agent can choose to go up, right, down, or left. When moving out of the grid the state is unchanged. Action random gives a reward of 10, while any other action returns a reward of $\mathsf{Unif}(0, 8)$. Moving out of the grid adds a punishment of -1 (see right side). Reaching state $(0, 0)$ with as little steps as possible is optimal. As by definition the average reward for unichain MDPs is equal among all states [14], for the optimal policy it is 7. Hence, the addend of the average reward equals 700 for each single state in case of $\gamma = 0.99$. The other parts of the state value are diminutive in that regard.

Add 3. For episodic MDPs standard discounted RL is unable to evaluate the average reward of terminal states correctly, as there is no consecutive state at the end of an episode.

Add 4. As each state is assessed separately, so is the average reward. States coincidentally visited more often at the beginning of the iterative learning process are better evaluated than the others, regardless of the specific reward feedback. The later is due to the shared average reward, cf. Add 2. This increases the likeliness that clusters form, with cyclic paths between the states (e.g. going back and fourth between some states). Setting a very high exploration rate is usually no remedy by the same argument. Further, recall that the average reward increases once the policy improves, e.g. for smaller exploration rates. This implies that finding the correct

[2] We use the notations π_γ and π_A, where Greek letters indicate the discount factor of the policy, while Latin symbols refer to specific policies.

[3] [15] incorrectly claims that for $\gamma < 0.75$ the policy is sub-optimal.

hyperparameter setting (e.g. exploration decay) requires a lot of effort and experience, as for instance reported in [8].

Clearly under these circumstances optimal policies are difficult to obtain for non-zero average reward MDPs. Therefore, in the sequel we present a more refined RL approach for operations research, which overcomes these issues by separately assessing the average reward.

4.2 The Laurent Series Expansion of Discounted State Values

The Laurent series expansion of the discounted state values [18, 23] provide important insights by giving rise to basically three addends. For a given discount factor γ the first addend is solely determined by the average reward[4], the second is the bias value and the third one, actually consisting of infinitely many sub-terms, is the error term. In the sequel we present the definitions of the average reward and bias value, before providing the Laurent series expansion.

Definition 4.2. Due to Howard [11] for an aperiodic[5] MDP the *gain* or *average reward* $\rho^{\pi_\gamma}(s)$ of a policy π_γ and a starting state s is defined as

$$\rho^{\pi_\gamma}(s) = \lim_{N\to\infty} \frac{\mathbb{E}[\sum_{t=0}^{N-1} R_t^{\pi_\gamma}(s)]}{N} .$$

Clearly $\rho^{\pi_\gamma}(s)$ expresses the expected average reward received per action taken when starting in state s and following policy π_γ. In the case of unichain MDPs, in which only a single set of recurrent states exists, the average reward $\rho^{\pi_\gamma}(s)$ is equal for all states s [14, 23]. Thus, in the sequel we may simply refer to it as ρ^{π_γ}.

Definition 4.3. For an aperiodic MDP problem the average adjusted sum of rewards or *bias value* is defined as

$$V^{\pi_\gamma}(s) = \lim_{N\to\infty} \mathbb{E}[\sum_{t=0}^{N-1} (R_t^{\pi_\gamma}(s) - \rho^{\pi_\gamma}(s))] .$$

Note that the bias values are bounded due to the subtraction of the average reward. Thus, the bias value describes the reward that additionally sums up in case the process starts in state s. Finally, a state value $V_\gamma^{\pi_\gamma}(s)$ in standard discounted reinforcement learning can be decomposed in its average reward, the bias value and an additional error term $e_\gamma^{\pi_\gamma}(s)$, which actually consists of infinitely many subterms.

Lemma 4.4. *Due to Miller and Veinott [18] the Laurent Series Expansion of the discounted state value for a state s, a discount factor γ and a policy π_γ is given by*

$$V_\gamma^{\pi_\gamma}(s) = \frac{\rho^{\pi_\gamma}(s)}{1-\gamma} + V^{\pi_\gamma}(s) + e_\gamma^{\pi_\gamma}(s) , \tag{1}$$

where Puterman [23, p.341] shows that $\lim_{\gamma\to 1} e_\gamma^{\pi_\gamma}(s) = 0$.

The error term $e_\gamma^{\pi_\gamma}(s)$ incorporates the amount and time until the reward is collected. The higher the discount factor the more long-sighted is the agent. Note how the first term depending on the average reward $\rho^{\pi_\gamma}(s)$ converges to infinity as γ converges to 1, as well as the second addend does not depend on the discount factor. If the average reward is non-zero and for large γ-values we can assume $\rho^{\pi_\gamma}(s)/(1-\gamma) \gg V^{\pi_\gamma}(s) + e_\gamma^{\pi_\gamma}(s)$ which explains the behaviour of the standard discounted RL agent, cf. the MDP in the lower part of Figure 2.

[4]We assume a positive average reward and the objective for maximisation in this work.
[5]In the periodic case the Cesaro limit of degree 1 is required to ensure stationary state transition probabilities and values [23]. To ease readability we concentrate on unichain aperiodic MDPs. The theory applies to period unichain MDPs by replacing the limits.

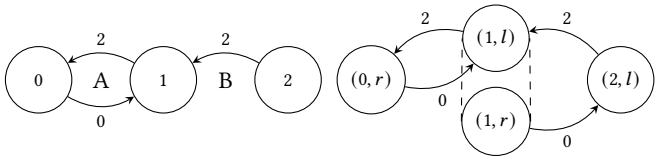

Figure 3: The (Blackwell-)optimal policy π_A of the MDP of Figure 1 as model-based (left) and model-free (right) version

Reconsider the task of Figure 1. The optimal MDP π_A, that is, the one that chooses the A-loop, is shown on the left of Figure 3. The right side depicts the same MDP for the model-free version. In model-free reinforcement learning state-action pairs as opposed to state values are estimated. The dashed line indicates that these states are connected and the agent has to choose among them. In this case the bias values are $V^{\pi_A}((0,r)) = -0.5$, $V^{\pi_A}((1,l)) = V^{\pi_A}((1,r)) = 0.5$, and $V^{\pi_A}((2,l)) = 1.5$.

When using standard discounted reinforcement learning, that is, estimating $V_\gamma^{\pi_\gamma}(s)$ the average reward of 1 scales the state-action values. E.g. for a discount factor of $\gamma = 0.99$ the inferred values for a converged system are $V_{0.99}^{\pi_A}((0,r)) = 99.497$, $V_{0.99}^{\pi_A}((1,l)) = 100.502$, $V_{0.99}^{\pi_A}((1,r)) = 100.482$, and $V_{0.99}^{\pi_A}((2,l)) = 101.497$. For all states the discounted state-value consists of the scaled average reward $1/0.01 = 100$ and the bias value plus the error term. The duration of the process of learning the state values is unintentionally increased as the scaled average reward has to be learned in an iterative manner and by every single state separately. Furthermore, the values itself are hard to interpret and the marginal difference of 0.02 between optimal and non-optimal action as compared to their actual values increase the likelihood of choosing sub-optimal actions, especially when function approximation is used to represent $V_\gamma^{\pi_\gamma}(s)$.

5 AVERAGE REWARD ADJUSTED DISCOUNTED REINFORCEMENT LEARNING

This section establishes the average reward adjusted discounted reinforcement learning algorithm, the Bellman Equations and discusses optimality. For the rest of the paper we assume unichain MDPs, i.e. we restrict our method to MDPs that posses a scalar average reward value ρ^{π_γ}.

Definition 5.1. The *average reward adjusted discounted state value* $X_\gamma^{\pi_\gamma}(s)$ of a state s under policy π_γ and with discount factor $0 \leqslant \gamma \leqslant 1$ is defined as

$$X_\gamma^{\pi_\gamma}(s) \coloneqq V^{\pi_\gamma}(s) + e_\gamma^{\pi_\gamma}(s) .$$

The reformulation $X_\gamma^{\pi_\gamma}(s) = V_\gamma^{\pi_\gamma}(s) - \frac{\rho^{\pi_\gamma}}{1-\gamma}$ depicts that $X_\gamma^{\pi_\gamma}(s)$ describes the discounted state value adjusted by the average reward.

In average reward RL the bias values are not uniquely defined without solving the first set of constraints defined by the error term

addends (see [16, 23, p.346]). This can be overcome by simply requiring γ to be strictly less than 1. However, our algorithm does not require the exact solution for $V^{\pi_\gamma}(s)$. Clearly this observation reduces the required iteration steps tremendously as finding the exact solution, especially for large discount factors, is tedious. Therefore, we allow to set $\gamma = 1$, which induces $X_\gamma^{\pi_\gamma}(s) = V^{\pi_\gamma}(s) + u$, where u is a scalar value [23, p.346]. If we are interested in correct bias values, i.e. γ is close but strictly less than 1, our approach is a tremendous advantage over average reward RL. This is due to a reduced number of iterative learning steps by requiring only a single constraint per state plus one for the scalar average reward value. For a MDP with N states only one more constraint ($N+1$) has to be solved in ARAL as compared to (at least) $2N+1$ nested constraints for average reward RL. Therefore, it is cheap to compute $X_\gamma^{\pi_\gamma}(s)$, while it is rather expensive to find the correct values of $V^{\pi_\gamma}(s)$ directly. Especially as RL estimates constraints iteratively.

5.1 Bellman Equations

We derive the Bellman equation based on the corresponding counterpart of $V_\gamma^{\pi_\gamma}(s)$ (see e.g. [28, p.70]). As in [28, p.66ff] we use the notation $R^a_{s,s'} = \mathbb{E}[r_t \mid s_t = s, a_t = a, s_{t+1} = s']$ for the expected reward to receive when traversing from any current state s to state s' using action a and denoting $\pi_\gamma(a \mid s)$ the probability of taking action a in state s as given by policy π_γ.

$$V_\gamma^{\pi_\gamma}(s) = \mathop{\mathbb{E}}_{\pi_\gamma}[r_t + \gamma V_\gamma^{\pi_\gamma}(s_{t+1}) \mid s_t = s]$$

$$V^{\pi_\gamma}(s) + e_\gamma^{\pi_\gamma}(s) = \mathop{\mathbb{E}}_{\pi_\gamma}[r_t + \gamma(V^{\pi_\gamma}(s_{t+1}) + e_\gamma^{\pi_\gamma}(s_{t+1})) + \rho^{\pi_\gamma} \cdot \frac{\gamma - 1}{1 - \gamma} \mid s_t = s]$$

$$X_\gamma^{\pi_\gamma}(s) = \mathop{\mathbb{E}}_{\pi_\gamma}[r_t + \gamma X_\gamma^{\pi_\gamma}(s_{t+1}) - \rho^{\pi_\gamma} \mid s_t = s]$$

$$X_\gamma^{\pi_\gamma}(s) = \sum_a \pi_\gamma(a \mid s) \sum_{s'} p(s' \mid s_t = s, a_t = a)(R^a_{s,s'} + \gamma X_\gamma^{\pi_\gamma}(s') - \rho^{\pi_\gamma})$$

Thus, we can compute the average reward adjusted discounted state value $X_\gamma^{\pi_\gamma}(s)$ of a state s by the returned reward, the adjusted discounted state value $X_\gamma^{\pi_\gamma}(s_{t+1})$ of the next state s_{t+1} and the average reward ρ^{π_γ}. Further, note that we use the equivalence $\rho^{\pi_\gamma}(s) = \mathbb{E}_{\pi_\gamma}[\rho^{\pi_\gamma}(s)]$ described by the first addend of the Laurent series expansion (see [18]). In the same manner we derive the Bellman optimality equation for average reward adjusted discounted reinforcement learning, cf. [28, p.76].

$$V_\gamma^{\pi_\gamma^\star}(s) = \max_a \mathop{\mathbb{E}}_{\pi_\gamma^\star}[r_t + \gamma V_\gamma^{\pi_\gamma^\star}(s_{t+1}) \mid s_t = s]$$

$$V^{\pi_\gamma^\star}(s) + e_\gamma^{\pi_\gamma^\star}(s) = \max_a \mathop{\mathbb{E}}_{\pi_\gamma^\star}[r_t + \gamma(V^{\pi_\gamma^\star}(s) + e_\gamma^{\pi_\gamma^\star}(s)) + \rho^{\pi_\gamma} \cdot \frac{\gamma - 1}{1 - \gamma} \mid s_t = s]$$

$$X_\gamma^{\pi_\gamma^\star}(s) = \max_a \mathop{\mathbb{E}}_{\pi_\gamma^\star}[r_t + \gamma X_\gamma^{\pi_\gamma^\star}(s) - \rho^{\pi_\gamma^\star} \mid s_t = s]$$

$$X_\gamma^{\pi_\gamma^\star}(s) = \max_a \sum_{s'} p(s' \mid s_t = s, a_t = a)(R^a_{s,s'} + \gamma X_\gamma^{\pi_\gamma^\star}(s') - \rho^{\pi_\gamma^\star})$$

As in standard discounted RL the optimal policy is computed by the expected value of the best action from that state [28, p.76], where the average reward is subtracted accordingly.

5.2 Near-Blackwell-Optimal Algorithm

The reinforcement learning algorithm is printed in Algorithm 1. In model-free methods, which is what we aim for, state-action pairs are computed instead of state values only. Therefore, the algorithm operates on state-action tuples, where for simplicity we write $X_\gamma^{\pi_\gamma}(s,a)$ instead of $X_\gamma^{\pi_\gamma}((s,a))$. Further, the Bellman optimality equation is adopted as in [28, p.76], s.t. the agent is able to selectively choose actions among multiple states, cf. Figure 3. In the action selection process (step 3) we utilise an ϵ-sensitive lexicographic order $\preccurlyeq_\epsilon$ defined as $(a_1, \ldots, a_n) = a \preccurlyeq_\epsilon b = (b_1, \ldots, b_n)$ if and only if $|a_j - b_j| \leqslant \epsilon$ for all $j < i$ and $|a_i - b_i| > \epsilon$. Note that the resulting sets of actions may not be disjoint. Although unusual in programming, finding the set of maximizing values as in our algorithm is straightforward and thus cheap to compute. In case the resulting set of actions contains more than one element a random action of this set of actions is chosen. All state values are updated using exponential smoothing, where we write $x \overset{\alpha}{\leftarrow} y$ for $x \leftarrow (1-\alpha)x + \alpha y$. To prevent introducing inaccuracies (cf. Theorem 3 of [18]) only when actions are chosen greedy the average reward is updated (step 5). The update formula $\rho^{\pi_\gamma}(s) + V^{\pi_\gamma}(s) - \mathbb{E}[V^{\pi_\gamma}(s)] = R_t(s)$ origins from the Laurent series expansion from [18] [see also 23, p.346] and has been identified as superior than exponentially smoothing the rewards [29]. It infers solid average reward predictions in few steps and thus quickly leads to better policies. Furthermore, we additionally prevent retreating to worse policies by adding an exponentially smoothed[6] bound from below for the average reward value. The Bellman optimality equation is the basis of the ARAL state-value update (step 6).

Algorithm 1 Model-free tabular near-Blackwell-optimal RL algorithm for unichain MDPs

1: Set exploration rate p_{expl}, exponential smoothing rates α, γ, and discount factors $0 < \gamma_0 < \gamma_1 \leqslant 1$.
2: **while** the stopping criterion is not fulfilled **do**
3: With probability p_{expl} choose a random action and prob. $1 - p_{\text{expl}}$ one that fulfills $\max_a \preccurlyeq_\epsilon (X_{\gamma_1}^{\pi_{\gamma_1}}(s_t,a), X_{\gamma_0}^{\pi_{\gamma_0}}(s_t,a))$.
4: Carry out action a_t, observe reward r_t and new state s_{t+1}.
5: **if** a non-random action was chosen **then**

$$\rho^{\pi_{\gamma_1}} \overset{\alpha}{\leftarrow} r_t + \max_a X_{\gamma_1}^{\pi_{\gamma_1}}(s_{t+1}, a) - X^{\pi_{\gamma_1}}\gamma_1(s_t, a_t)$$

6: Update average reward adjusted discounted state-values

$$X^{\pi_{\gamma_0}}(s_t, a_t) \overset{\gamma}{\leftarrow} r_t + \gamma_0 \max_a X^{\pi_{\gamma_0}}(s_{t+1}, a) - \rho^{\pi_{\gamma_1}}$$

$$X_{\gamma_1}^{\pi_{\gamma_1}}(s_t, a_t) \overset{\gamma}{\leftarrow} r_t + \gamma_1 \max_a X_{\gamma_1}^{\pi_{\gamma_1}}(s_{t+1}, a) - \rho^{\pi_{\gamma_1}}$$

7: Set $s \leftarrow s'$, $t \leftarrow t+1$ and decay parameters

5.3 Optimality Criteria

We consider the notion of n-discount optimality, as, for a sufficiently large n, it is known that there exists a policy which is optimal [6, 30]. For a comprehensive discussion of optimality criteria in reinforcement learning we refer to [15].

Definition 5.2 ([30]). For MDPs a policy $\pi_\gamma^\star$ is *n-discount-optimal* for $n = -1, 0, 1, \ldots$ for all states $s \in \mathcal{S}$ with discount factor γ if and only if $\lim_{\gamma \to 1}(1-\gamma)^{-n}\,(V_\gamma^{\pi_\gamma^\star}(s) - V_\gamma^{\pi_\gamma}(s)) \geqslant 0$.

[6] With rate $\frac{1}{50}$ and update of 97.5% of the current reward in every period.

Hence, the correct solution to the underlying constraint problem for the optimal policy $\pi^\star$ of the MDP requires $\pi_{1.0}$. Nonetheless, there exists a discount factor $\gamma < 1$ that finds the optimal policy [6, 16]. A policy can only be m-discount optimal if it is n-discount-optimal for all $n < m$ [23, 30]. This leads to the componentwise comparison when greedily choosing actions, cf. the algorithm. If a policy is ∞-discount-optimal then it is said to be *Blackwell-optimal* [6]. Blackwell-optimal policies are the optimal policy $\pi^\star$ for the underlying MDP. Put differently, they are the, in the sense of n-discount-optimality, best achievable policies. They first optimise for the highest gain, as we have for $n = -1$ a measure for gain-optimality [14]. Then for $n = 0$ for bias-optimality [14]. And for $n \geqslant 1$ they maximise the error term, where for increasing n more distant steps are considered. For an agent that either expects to have infinitely many time to collect rewards, or one that is unaware when the system will halt, this order is the most sensible approach.

In the Literature there are mainly two solution strategies to RL. First, to use a single discounted state value, and, second, decomposing the state value into infinitely many addends. The later in general incorporates solving infinitely many constraints. The following definition separates these kinds of algorithms.

Definition 5.3. If an algorithm infers for any MDP bias-optimal policies, and for any given MDP can in theory be configured to infer ∞-discount-optimal policies, but in practise this ability is naturally limited due to finite memory availability and accuracy of floating-point representation of modern computer systems, it is said to be *near-Blackwell-optimal* under the given computer system. An according to a near-Blackwell-optimal algorithm inferred Blackwell-optimal policy is called near-Blackwell-optimal.

This definition is of practical relevance, as it defines a group of algorithms that are by far less computationally expensive in comparison to ones that solve infinitely many constraints, but are able to deduce sufficiently optimal policies.

5.4 Optimality Analysis

In the sequel we show that ARAL is indeed near-Blackwell-optimal. For $n = -1$ gain-optimality is required [14].

Theorem 5.4. *For $\gamma_1 = 1$ and unichain MDPs ARAL infers gain-optimal policies, i.e. it finds a policy $\pi^\star_\gamma$ with $\rho^{\pi^\star_\gamma}(s) - \rho^{\pi_\gamma}(s) \geqslant 0$ for any other policy π_γ.*

Proof. With $\gamma_1 = 1$ and as in unichain MDPs the average reward is equal among all states [14], we have $X^{\pi_\gamma}_\gamma(s) = V^{\pi_\gamma}(s) + u$, where u is a scalar value [23, p.346]. All bias values $V^{\pi_\gamma}(s)$ are assessed with the same average reward ρ^π_γ. As by definition the bias value $V^{\pi_\gamma}(s)$ depends on the returned reward and average reward ρ^π_γ only. By learning the average reward greedy only and choosing actions with the maximum bias value, the theorem follows. □

An interesting insight is, that by definition, for incorrectly fixed average reward value, the bias values are estimated according to the given policy induced by the average reward. Furthermore, the values shift similar as in the standard discounted RL. Hence, the need to infer the average reward automatically. Reconsider the MDP of Figure 2 and the discount factor $\gamma = 0.99$. When fixing the average reward to $\rho^{\pi_\gamma}_{\text{fix}} = 1$, instead to the correct value of 2, the algorithm infers values $X^{\pi_\gamma}_{\text{fix},\gamma}(s) = \frac{\rho^{\pi_\gamma} - \rho^{\pi_\gamma}_{\text{fix}}}{1-\gamma} + X^{\pi_\gamma}_\gamma(s) = 100 + X^{\pi_\gamma}_\gamma(s)$ instead.

In the case of *0*-Discount-Optimality we have bias-optimality with $V^{\pi^\star_\gamma}(s) - V^{\pi_\gamma}(s) \geqslant 0$ for all policies π_γ and states $s \in \mathcal{S}$ [14].

Theorem 5.5. *For a sufficiently large $\gamma_1 \leqslant 1$, such that for all states s we have $|e^{\pi_{\gamma_1}}_{\gamma_1}(s)| \leqslant \epsilon$, ARAL infers bias-optimality policies, i.e. it finds a policy $\pi^\star_\gamma$ with $V^{\pi^\star_\gamma}(s) - V^{\pi_\gamma}(s) \geqslant 0$.*

Proof. Recall that $\lim_{\gamma\to 1} e^{\pi_\gamma}_\gamma(s) = 0$ and $\rho^{\pi^\star_\gamma} = \rho^{\pi_\gamma}$. We have

$$\lim_{\gamma\to 1}(1-\gamma)^0 (V^{\pi^\star_\gamma}_\gamma(s) - V^{\pi_\gamma}_\gamma(s)) \geqslant 0$$

$$\lim_{\gamma\to 1}\left(\frac{\rho^{\pi^\star_\gamma} - \rho^{\pi_\gamma}}{1-\gamma} + V^{\pi^\star_\gamma}(s) - V^{\pi_\gamma}(s) + e^{\pi^\star_\gamma}_\gamma(s) - e^{\pi_\gamma}_\gamma(s)\right) \geqslant 0$$

$$V^{\pi^\star_\gamma}(s) - V^{\pi_\gamma}(s) \geqslant 0$$

meaning that a 0-discount-optimal policy π_γ has to maximise the bias values $V^{\pi_\gamma}(s)$ for all states s. By definition of the ϵ-sensitive lexicographic order ($a \preccurlyeq_\epsilon b$ if and only if $|a_j - b_j| \leqslant \epsilon$), we have for a sufficiently large γ_1-value $|e^{\pi_{\gamma_1}}_{\gamma_1}(s)| \leqslant \epsilon$ and thus $|V^{\pi^\star_\gamma}(s) - X^{\pi_{\gamma_1}}_{\gamma_1}(s)| \leqslant \epsilon$ for all states s. Thus the claim follows. □

Finally, for $n \geqslant 1$, the agent has to choose actions that satisfy $e^{\pi^\star_\gamma}_\gamma(s) - e^{\pi_\gamma}_\gamma(s) \geqslant 0$ once $\gamma \to 1$. Hence, we are focusing on the cases with $\gamma < 1$, i.e. how long-sighted the agent shall be. Therefore, the number of actions to reach a desired goal or path is taken into account, as well as when and how much rewards are collected. However, as the error term depends on infinitely many sub-terms simply estimating these and summing up does not work.

Hence, the RL algorithm depicted in Algorithm 1 is unable to generally deduce Blackwell-optimal policies. This is due to the non-linearity of the error term values in regard to the discount factor. The MDP of Figure 4 explains the idea. The only action choice is in state S, where the agent either decides to do the top- or bottom-loop. For both loops the same amount of rewards are collected, thus regardless of the chosen policy the average reward (0.75) and bias values are equal (0.25 for the policy that goes up, 0.375 for the policy that chooses down). But only going down is Blackwell-optimal, as the full amount of rewards is collected earlier. This also manifests in a higher bias value. However, recall that the agent is unable to separate the actions by the bias values in case the same amount of rewards are collected. If we set $\gamma_0 = 0.50$ the agent deduces state-values $V^{\pi_{\text{Top}}}_{\gamma_0}(S) = -0.480$ and $V^{\pi_{\text{Bottom}}}_{\gamma_0}(S) = -0.746$ and thus like for any other value $\gamma_0 < 0.84837$ chooses the top-loop. Due to the fact that the state values are adjusted, γ_0 of our algorithm functions exactly as the discount factor in standard RL is supposed to do: It can be used to balance expected short-term and long-term rewards, without changing the main optimisation objective, i.e. maximise for the highest average reward and bias values, before taking path lengths into account. Especially for highly volatile systems, e.g. stochastic production and control systems, being able to set the long-sightedness can be an advantage.

Theorem 5.6. *For any given MDP ARAL can be configured such that it infers a Blackwell-optimal policy, i.e. it finds a policy $\pi^\star_\gamma$ with $e^{\pi^\star_\gamma}_\gamma(s) - e^{\pi_\gamma}_\gamma(s) \geqslant 0$ for any other policy π_γ.*

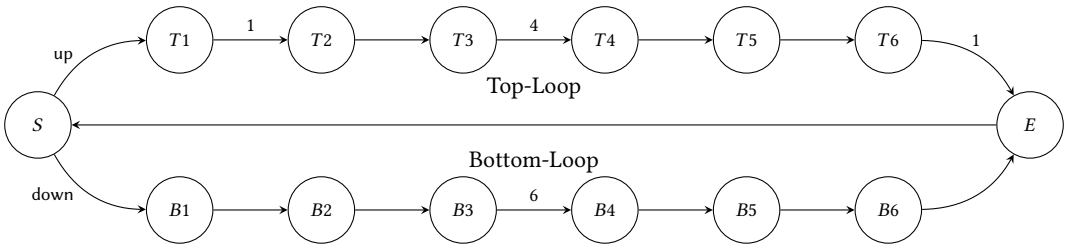

Figure 4: An MDPs in which the discount factor γ_0 is used to balance short- and long-sightedness of ARAL.

Proof. Due to 0-discount-optimality we have $\rho^{\pi^\star_\gamma} = \rho^{\pi_\gamma}$ and $V^{\pi^\star_\gamma}(s) = V^{\pi_\gamma}(s)$ for all states s. Therefore,

$$\lim_{\gamma\to 1}(1-\gamma)^{-n}(V^{\pi^\star_\gamma}_\gamma(s) - V^{\pi_\gamma}_\gamma(s)) \geqslant 0$$

$$\lim_{\gamma\to 1}(1-\gamma)^{-n}\left(\frac{\rho^{\pi^\star_\gamma} - \rho^{\pi_\gamma}}{1-\gamma} + V^{\pi^\star_\gamma}(s) + e^{\pi^\star_\gamma}_\gamma(s) - V^{\pi_\gamma}(s) - e^{\pi_\gamma}_\gamma(s)\right) \geqslant 0$$

$$\lim_{\gamma\to 1}(1-\gamma)^{-n}(e^{\pi^\star_\gamma}_\gamma(s) - e^{\pi_\gamma}_\gamma(s)) \geqslant 0$$

Note that the error term does not depend on n. It is known that for Blackwell-optimal policies $\pi^\star_\gamma$ there exists a discount factor $\gamma^\star < 1$ such that $V^{\pi^\star_\gamma}_\gamma(s) \geqslant V^{\pi_\gamma}_\gamma(s)$ for all $\gamma \geqslant \gamma^\star$ and under all policies π_γ [6, 16]. Hence for sufficiently large γ_0 and small ϵ we have $|e^{\pi^\star_{\gamma_0}}_{\gamma_0}(s) - X^{\pi_{\gamma_0}}_{\gamma_0}(s)| \leqslant \epsilon$ for all states s. □

In the Literature the error term is often split into its subterms by $e^{\pi_\gamma}_\gamma(s) := \sum_{m=1}^{\infty}(\frac{1-\gamma}{\gamma})^m y^{\pi_\gamma}_m$ (for the definition see [18]). Then for any $n \geqslant 1$ the terms evaluate to maximising $y^{\pi_\gamma}_n$. This leads to the approaches of average reward dynamic programming, where n-nested sets of constraints are solved [23, e.g. p.511ff]. Clearly, this straightforward approach is computationally very expensive and infinite. We refrain on adapting this strategy and rather let the user choose an appropriate γ_0 value for the provided situation.

Thus, for any given MDP and under the assumption of correct approximations, as well as wisely chosen parameters ARAL infers Blackwell-optimality policies, but this is bounded by the characteristics of the computer system. Therefore, the presented reinforcement learning algorithm ARAL is **near-Blackwell-optimal**.

6 BENCHMARK

6.1 Benchmark Algorithm

We compare the algorithm to standard discounted RL, where we choose the widely applied Q-Learning [28, 32] technique as appropriate model-free comparison method. This decision is due to the fact that ARAL is based on the ideas of Q-Learning and thus well comparable. The implementation details with adapted parameter names to match the corresponding counterparts of ARAL are given Algorithm 2.

6.2 Problem Set

Due to the lack of space, in this version we focus on an admission control queuing problem. Nonetheless, we have tested the algorithm with success on various problems, like the printer-mail and gridworld problems of Figure 2, but using a 5×5 grid. In general we found that our algorithm outperforms Q-Learning on the tested examples, as the latter struggles with the average reward. We selected the admission control queuing problem as it is key to solve it with Blackwell-optimality, i.e. bias-optimality is insufficient.

Algorithm 2 Watkins Q-Learning algorithm [32]. Adapted from the version by Sutton [28, p.149].

1: Set an exploration rate $0 \leqslant p_{\text{expl}} \leqslant 1$ and $0 < \gamma, \gamma_1 < 1$.
2: **while** the stopping criterion is not fulfilled **do**
3: With probability p_{expl} choose a random action and otherwise one that fulfills $\max_a Q^\pi_{\gamma_1}(s_t, a)$ at the current state s_t.
4: Carry out action a_t, observe reward r_t and new state s_{t+1}.
5: Update the discounted state-values.

$$Q^{\pi_{\gamma_1}}(s_t, a_t) \overset{\gamma}{\leftarrow} r_t + \gamma_1 \max_{a'} Q^{\pi_{\gamma_1}}(s_{t+1}, a')$$

6: Set $s \leftarrow s'$, $t \leftarrow t+1$ and decay parameters

Admission Control Queuing System. Like Mahadevan [13, 16] we evaluate the algorithm on a simple M/M/1 admission control queuing system assuming one server that processes jobs, which arrive by an exponential (Markov) interarrvial time distribution. The processing duration is assumed to be exponentially distributed. The arrival and service rate are modeled by parameter λ and μ respectively. On each new arrival the agent decides whether to accept the job and thus add it to the queue, or reject the job. In case of acceptance an immediate reward is received, which also incurs a holding cost depending on the current queue size. The goal is to maximise the reward by balancing the admission allowance reward and the holding costs. The discrete time MDP is depicted in Figure 5 and is the result of a transformation using the uniformisation technique from a continuous time problem (see [5, 23] for a description of uniformisation). The set of states consists of elements (l, Arr) with queue length $l \in \mathbb{N}$ and a Boolean variable $\text{Arr} \in \{\text{T}, \text{F}\}$, where Arr symbolises an arrival (T), or no arrival (F). The edges are labelled with the transition probability and the corresponding action, that is accept and reject in case of a new arrival, or continue when no new job arrived. The reward function r is defined as in [13, 16]:

$$\begin{aligned}
r((0,\text{F}),\text{continue}) = r((0,\text{T}),\text{reject}) &= 0 && \text{if } s = 0\,,\\
r((l,\text{F}),\text{continue}) &= -f(l+1)(\lambda+\mu) && \text{if } l \geqslant 1\,,\\
r((l,\text{T}),\text{reject}) &= -f(l+1)(\lambda+\mu) && \text{if } l \geqslant 1\,,\\
r((l,\text{T}),\text{accept}) &= [R - f(l+1)](\lambda+\mu)\,.
\end{aligned}$$

The factor $\lambda+\mu$ is an artifact of the uniformisation of the continuous time problem to the discrete time MDP.

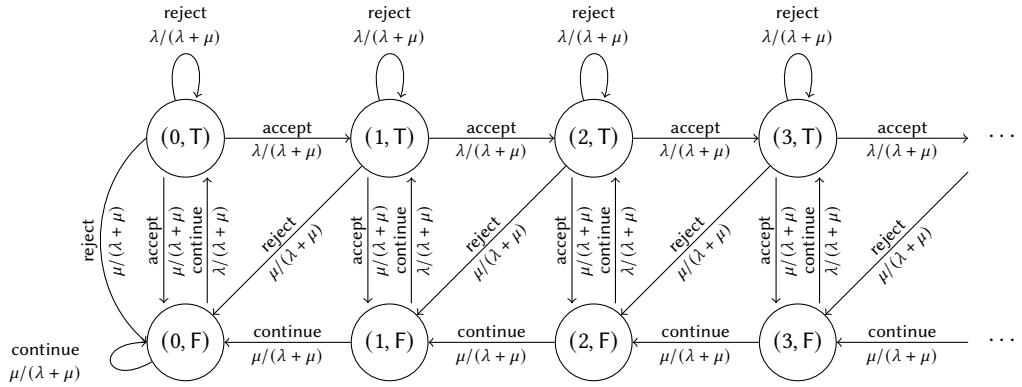

Figure 5: This diagram illustrates a simple M/M/1 admission control queuing system (adapted from Mahadevan [13, 16])

Haviv and Puterman [9] show that, if the cost function has the shape $f(l) = c \cdot l$, there are at most two gain-optimal control limit policies. Namely to admit L or $L+1$ jobs. However, only the policy that admits $L+1$ jobs is also bias-optimal as the extra reward received offsets the additional cost of the extra job.

We use exactly this cost function in our experiment, with the challenging problem setup $\lambda = 5$, $\mu = 5$, $R = 12$, $c = 1$, and a maximum queue length of 20 as in [13, 16]. To compute the optimal solution we implement the constraint formulation of the Laurent series expansion [18, 23, p.346] using mixed integer linear programming (MILP). The optimum is $L = 2$, i.e. both policies of admitting 2 or 3 jobs to the queue are gain-optimal with average reward of $\rho^{\pi^\star_{\gamma_1}} = 30$, but only admitting 3 jobs is also bias-optimal. And as such also Blackwell-optimal, as it's the only gain-optimal policy that is left. This makes sense, as R is only collected when an order is accepted, which is for admitting 3 jobs immediate in contrast to the policy of admitting 2. The correct queue length for admitting 2 jobs is 0.67, while for 3 it is 1.12.

The initial setup is $\alpha = \gamma = 0.01$, $\gamma_0 = 0.80$, $\epsilon = 5$, $p_{\text{expl}} = 1.00$. The learning rates and exploration are exponentially decayed as follows. α with a rate of 0.50 in $50k$ steps and a minimum of 10^{-5}, γ with rate of 0.50 in $150k$ steps and 10^{-3}, and finally the exploration with rate 0.50 in $100k$ steps and a minimum of 0.01. We execute 10^6 learning steps before doing an evaluation run of $100k$ steps for which exploration and learning is disabled. The experiment, with learning process, is repeated 40 times with the same random number streams over the setups.

7 EXPERIMENTAL EVALUATION

The results are depicted on the right of Table 1. Gray colored cells are statistically indistinguishable (Friedman and Conover test [4], $p = 0.05$). The ARAL variants with $\gamma_1 = 1.0$ and $\gamma_1 = 0.999$ infer the Blackwell-optimal policy of admitting 3 jobs, accumulating a reward of about 29.88 and 29.77 per step over all 40 evaluation replications. This clearly shows the stability of the ARAL algorithm, especially when $\gamma_1 = 1.0$. ARAL $\gamma_1 = 0.99$ is unable to find the optimal policy in 12 replications and therefore performs worse as compared to the other ARAL instances. Statistically the optimal queue length of 1.12 is matched by ARAL $\gamma_1 = 0.999$ and ARAL $\gamma_1 = 1.0$. In contrast all

	Sum Reward		Queue Length	
Algorithm	Mean	StdDev	Mean	StdDev
ARAL $\gamma_1 = 1.0$	**2988054.750**	23977.493	1.075	0.157
ARAL $\gamma_1 = 0.999$	2976862.250	64688.399	**1.122**	0.184
ARAL $\gamma_1 = 0.99$	2683089.250	1021845.424	1.545	1.263
Q-Learning $\gamma_1 = 0.99$	45360.750	13434.404	0.174	0.057
Q-Learning $\gamma_1 = 0.999$	32609.250	16216.020	0.181	0.080
Q-Learning $\gamma_1 = 0.5$	24917.500	1467.415	0.002	0.000

Table 1: Experiment Results. Inferred avg. rewards for $\gamma_1 = 1.0, 0.999$, and 0.99 are $\rho^{\pi_{\gamma_1}} = 29.965, 30.272$ and 28.734 resp.

Q-Learning setups are unable to find even close-to-optimal policies, resulting in rather small amount of collected rewards and very short mean queue lengths. Furthermore, we investigated how to get Q-Learning $\gamma_1 = 0.99$ to find better solutions. Using ϵ-sensitive comparison, instead of the max-operator, for the action selection process we could infer the gain-optimal policy which admits 2 jobs. However, we were unable to infer the Blackwell-optimal policy of admitting 3 jobs. Similarly by hyperparameter tuning of ARAL $\gamma_1 = 1.0$, namely increasing the decay rates of α and γ to 0.8 and omitting the minimum values, we were able to find more stable state-action values such that setting $\epsilon \leqslant 1$ is possible while still finding the optimal policy and the average reward stabilizes even more. We found that the stability of the average reward value is of major importance for the stability of ARAL.

8 CONCLUSION AND FUTURE WORK

In this work, we focus on such problems that produce an average reward per step that cannot be approximated by 0. We introduce ARAL, a novel adaption of the discounted reinforcement method and give ample evidence of the usefulness of ARAL in this context. The algorithm is specifically designed for commonly-used unichain (and thus ergodic) MDPs. ARAL explicitly handles the average reward per step, which is used to adjust the discounted state values. Using two different discount factors, the agent infers bias-optimal policies and for any given MDP can be configured to infer Blackwell-optimal policies. In the future we plan to use neural networks for function approximation (cf. [26]). Adding to this we plan to investigate the further adaption to an actor-critic version of ARAL.

REFERENCES

[1] Bharathan Balaji, Jordan Bell-Masterson, Enes Bilgin, Andreas Damianou, Pablo Moreno Garcia, Arpit Jain, Runfei Luo, Alvaro Maggiar, Balakrishnan Narayanaswamy, and Chun Ye. 2019. ORL: Reinforcement Learning Benchmarks for Online Stochastic Optimization Problems. *arXiv preprint arXiv:1911.10641* (2019).

[2] Erik Banks. 2010. *Finance: the basics*. Routledge.

[3] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940* (2016).

[4] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[5] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. 1995. *Dynamic programming and optimal control*. Vol. 1. Athena scientific Belmont, MA.

[6] David Blackwell. 1962. Discrete dynamic programming. *The Annals of Mathematical Statistics* 344 (1962), 719–726. https://doi.org/016/j.cam.2018.05.030

[7] S Kamal Chaharsooghi, Jafar Heydari, and S Hessameddin Zegordi. 2008. A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems* 45, 4 (2008), 949–959.

[8] Joren Gijsbrechts, Robert N Boute, Jan A Van Mieghem, and Dennis Zhang. 2018. Can deep reinforcement learning improve inventory management? performance and implementation of dual sourcing-mode problems. *Performance and Implementation of Dual Sourcing-Mode Problems (December 17, 2018)* (2018).

[9] Moshe Haviv and Martin L Puterman. 1998. Bias optimality in controlled queueing systems. *Journal of Applied Probability* 35, 1 (1998), 136–150.

[10] A. C. Hax and H. C. Meal. 1973. *Hierarchical integration of production planning and scheduling*. Report. DTIC Document.

[11] Ronald A Howard. 1960. *Dynamic programming and markov processes*. John Wiley.

[12] Wouter Kool, Herke van Hoof, and Max Welling. 2018. Attention, Learn to Solve Routing Problems! *arXiv preprint arXiv:1803.08475* (2018).

[13] Sridhar Mahadevan. 1996. An average-reward reinforcement learning algorithm for computing bias-optimal policies. In *AAAI/IAAI, Vol. 1*. 875–880.

[14] Sridhar Mahadevan. 1996. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning* 22 (1996), 159–195.

[15] Sridhar Mahadevan. 1996. Optimality criteria in reinforcement learning. In *Proceedings of the AAAI Fall Symposium on Learning Complex Behaviors in Adaptive Intelligent Systems*.

[16] Sridhar Mahadevan. 1996. Sensitive discount optimality: Unifying discounted and average reward reinforcement learning. In *ICML*. 328–336.

[17] Sridhar Mahadevan, Nicholas Marchalleck, Tapas K Das, and Abhijit Gosavi. 1997. Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *Machine Learning-International Workshop Then Conference-*. Morgan Kaufmann Publishers, Inc., 202–210.

[18] B. L. Miller and A. F. Veinott. 1969. Discrete Dynamic Programming with a Small Interest Rate. *The Annals of Mathematical Statistics* 40, 2 (1969), 366–370. http://www.jstor.org/stable/2239451

[19] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[21] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. 2018. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*. 9839–9849.

[22] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence Snyder, and Martin Takáč. 2017. A Deep Q-Network for the Beer Game: A Reinforcement Learning Algorithm to Solve Inventory Optimization Problems. *arXiv preprint arXiv:1708.05924* (2017).

[23] Martin L Puterman. 1994. Markov Decision Processes. J. *Wiley and Sons* (1994).

[24] Jens Rohde. 2004. Hierarchical supply chain planning using artificial neural networks to anticipate base-level outcomes. *OR Spectrum* 26, 4 (2004), 471–492.

[25] Manuel Schneckenreither and Stefan Haeussler. 2018. Reinforcement Learning Methods for Operations Research Applications: The Order Release Problem. In *International Conference on Machine Learning, Optimization, and Data Science*. Springer, 545–559.

[26] Manuel Schneckenreither, Stefan Haeussler, and Juanjo Peiro. 2022. Average reward adjusted deep reinforcement learning for order release planning in manufacturing. *Knowledge-Based Systems* (2022), 108765. https://doi.org/10.1016/j.knosys.2022.108765

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[28] Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*. Vol. 2. MIT press Cambridge.

[29] Prasad Tadepalli and DoKyeong Ok. 1998. Model-based average reward reinforcement learning. *Artificial intelligence* 100, 1-2 (1998), 177–224.

[30] Arthur F. Veinott. 1969. Discrete dynamic programming with sensitive discount optimality criteria. *The Annals of Mathematical Statistics* 40, 5 (1969), 1635–1660. https://doi.org/10.1214/aoms/1177697379

[31] Jose Manuel Vera and Andres G Abad. 2019. Deep Reinforcement Learning for Routing a Heterogeneous Fleet of Vehicles. *arXiv preprint arXiv:1912.03341* (2019).

[32] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College.

[33] Willem HM Zijm. 2000. Towards intelligent manufacturing planning and control systems. *OR-Spektrum* 22, 3 (2000), 313–345.