

# Dynamic Strategies for $TCT$

**Manuel Schneckentreither**

Department of Computer Science,  
Computational Logic Group,  
University of Innsbruck, Austria  
email: [manuel.schneckentreither@uibk.ac.at](mailto:manuel.schneckentreither@uibk.ac.at)

The Twenty First International Workshop on Logic and  
Computational Complexity July 6, 2020

Paper & Slides: <https://github.com/schneck/i/lcc20>



# Motivation

## The Tyrolean Complexity Tool, i.e. $TCT$

- Automatic tool to infer runtime (and derivational) complexity
- Allows (first or higher-order) functional programs as generalised form of term rewrite systems as input
- In case of success:  $TCT$  outputs an asymptotic complexity and corresponding proof



[1] Landi, “Undecidability of static analysis”, 1992

# Motivation

## The Tyrolean Complexity Tool, i.e. $TCT$

- Automatic tool to infer runtime (and derivational) complexity
- Allows (first or higher-order) functional programs as generalised form of term rewrite systems as input
- In case of success:  $TCT$  outputs an asymptotic complexity and corresponding proof

## Static Analyses

- Static analyses are undecidable <sup>[1]</sup>
- Thus: Some sort of creativity is required

---

[1] Landi, “Undecidability of static analysis”, 1992

## In $T_C T$

- $T_C T$  uses Strategies to define:
  - what methods (e.g. poly./matrix interpretations) to use and
  - in which sequence these are used.
- Currently strategies are predetermined and fixed in  $T_C T$
- Idea: Reinforcement learning for dynamic strategies
  - No training data needed (no predetermined strategies)
  - Provides a much larger solution space



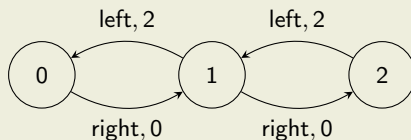
## Table of Contents

1. Motivation
2. Reinforcement Learning
  - Discounted Reinforcement Learning (Discounted RL)
  - Average Reward Adjusted Discounted RL
3. Proposed Implementation in  $TCT$



# Reinforcement Learning (RL)

## Reinforcement Learning Processes [2]

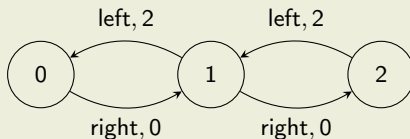


- Based on dynamic programming: states, actions, rewards

[2] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969

# Reinforcement Learning (RL)

## Reinforcement Learning Processes [2]



- Agent iteratively assesses state values  $V_{\gamma}^{\pi}(s)$ , where
- $0 < \gamma < 1$  is the discount factor,  $\pi$  the policy function, and
- for two consecutive observed states  $s_t, s_{t+1}$  with reward  $r_t$

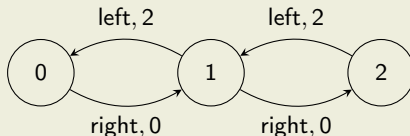
$$V_{\gamma}^{\pi}(s_t) \stackrel{\alpha}{\leftarrow} r_t + \gamma V_{\gamma}^{\pi}(s_{t+1})$$

( $\stackrel{\alpha}{\leftarrow}$  is exp. smoothed updating with rate  $\alpha$ )

[2] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969

# Reinforcement Learning (RL)

## Reinforcement Learning Processes [2]



- Thus, state values  $V_{\gamma}^{\pi}(s)$  are

$$V_{\gamma}^{\pi}(s) = \lim_{N \rightarrow \infty} E\left[\sum_{t=0}^{N-1} \gamma^t R_t^{\pi}(s)\right]$$

where  $R_t^{\pi}(s)$  the reward received at time  $t$  upon starting in state  $s$  by following policy  $\pi$ .

[2] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969



# Aim in Discounted Reinforcement Learning

## Aim of Discount Reinforcement Learning

- Finding an optimal policy  $\pi^*$
- $\pi^*$  maximises the state value for all states  $s$  as compared to any other policy  $\pi$ :

$$V_{\gamma}^{\pi^*}(s) - V_{\gamma}^{\pi}(s) \geq 0$$



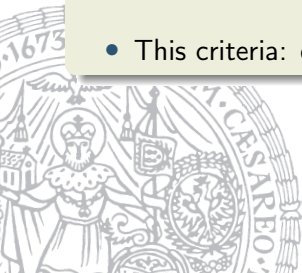
# Aim in Discounted Reinforcement Learning

## Aim of Discount Reinforcement Learning

- Finding an optimal policy  $\pi^*$
- $\pi^*$  maximises the state value for all states  $s$  as compared to any other policy  $\pi$ :

$$V_{\gamma}^{\pi^*}(s) - V_{\gamma}^{\pi}(s) \geq 0$$

- This criteria: discounted-optimality as  $\gamma$  is **fixed**



# Aim in Discounted Reinforcement Learning

## Aim of Discount Reinforcement Learning

- Finding an optimal policy  $\pi^*$
- $\pi^*$  maximises the state value for all states  $s$  as compared to any other policy  $\pi$ :

$$V_{\gamma}^{\pi^*}(s) - V_{\gamma}^{\pi}(s) \geq 0$$

- This criteria: discounted-optimality as  $\gamma$  is **fixed**

The higher  $\gamma$  the more optimal are the solutions.

# Aim in Discounted Reinforcement Learning

## Aim of Discount Reinforcement Learning

- Finding an optimal policy  $\pi^*$
- $\pi^*$  maximises the state value for all states  $s$  as compared to any other policy  $\pi$ :

$$V_{\gamma}^{\pi^*}(s) - V_{\gamma}^{\pi}(s) \geq 0$$

- This criteria: discounted-optimality as  $\gamma$  is **fixed**

The higher  $\gamma$  the more optimal are the solutions.

Usual values:  $\gamma \geq 0.99$

# Laurent Series Expansion of Discounted State Values

## Laurent Series Expansion of Discounted State Values<sup>[3]</sup>

The Laurent series expansion of  $V_\gamma^\pi(s)$  of discounted state values:

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

## Average Reward $\rho^\pi$ (simplified for unichain Processes)

The **average reward** is defined as

$$\rho^\pi = \lim_{N \rightarrow \infty} \frac{\mathbb{E}[\sum_{t=0}^{N-1} R_t^\pi]}{N}$$

where  $R_t^\pi$  is the reward received at time  $t$ .

[3] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969

# Laurent Series Expansion of Discounted State Values

## Laurent Series Expansion of Discounted State Values<sup>[3]</sup>

The Laurent series expansion of  $V_\gamma^\pi(s)$  of discounted state values:

$$V_\gamma^\pi(s) = \frac{\rho^\pi}{1-\gamma} + \textcolor{red}{V}^\pi(s) + e_\gamma^\pi(s)$$

## Bias value $V^\pi(s)$

The **bias value** is defined as

$$V^\pi(s) = \lim_{N \rightarrow \infty} \mathbb{E} \left[ \sum_{t=0}^{N-1} (R_t^\pi(s) - \rho^\pi) \right]$$

It is the additional reward that sums up when starting in state  $s$ .

[3] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969

# Laurent Series Expansion of Discounted State Values

## Laurent Series Expansion of Discounted State Values<sup>[3]</sup>

The Laurent series expansion of  $V_\gamma^\pi(s)$  of discounted state values:

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

## Error term of $e_\gamma^\pi$ <sup>[4]</sup>

**Error term**  $e_\gamma^\pi(s)$  consists of infinitely many terms, but

$$\lim_{\gamma \rightarrow 1} e_\gamma^\pi(s) = 0$$

If  $\gamma < 1$  it takes number of steps and reward values into account.

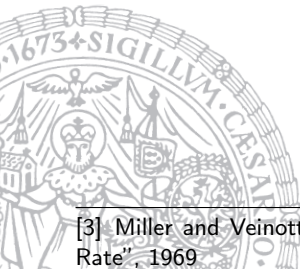
[4] Puterman, "Markov Decision Processes: Discrete Stochastic Dynamic Programming", 1994

# Laurent Series Expansion of Discounted State Values

## Laurent Series Expansion of Discounted State Values<sup>[3]</sup>

The Laurent series expansion of  $V_\gamma^\pi(s)$  of discounted state values:

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



[3] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969



# Laurent Series Expansion of Discounted State Values

## Laurent Series Expansion of Discounted State Values<sup>[3]</sup>

The Laurent series expansion of  $V_\gamma^\pi(s)$  of discounted state values:

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

### Note

- Recall: High  $\gamma \approx 1$  values required for optimal policies
- First term converges to  $\infty$  as  $\gamma \rightarrow 1$
- Recall: RL is an iterative method

[3] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969

# So What?

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



## So What?

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

Consider this simple gridworld problem [4]

(0,0) random	up	left (0,1)	right
	down	down	
up	up	left (1,0)	right
down	down	left (1,1)	right

State/Action Space

(0,0) 10	$\mathcal{U}(0,8)-1$	$\mathcal{U}(0,8)$	$\mathcal{U}(0,8)-1$
	$\mathcal{U}(0,8)$	$\mathcal{U}(0,8)$	$\mathcal{U}(0,8)-1$
$\mathcal{U}(0,8)-1$	$\mathcal{U}(0,8)$	$\mathcal{U}(0,8)$	$\mathcal{U}(0,8)-1$
$\mathcal{U}(0,8)-1$	$\mathcal{U}(0,8)-1$	$\mathcal{U}(0,8)-1$	$\mathcal{U}(0,8)-1$

Reward Function

[4] Manuel Schneckeneither, "Average Reward Adjusted Discounted Reinforcement Learning: Near-Blackwell-Optimal Policies for Real-World Applications", 2020

## So What?

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

Same problem as 5x5 grid

	23.113	162.243	354.839	398.391
(0,0) 101.260	67.160 (0,1) 61.783 49.812	49.812 (0,2) 349.515 306.761	306.761 (0,3) 399.560 399.376	399.376 (0,4) 398.444 399.539
...	...	...	...	...

- All states values completely assessed independently
- For  $\gamma \approx 1$ :  $\rho^{\pi}/(1 - \gamma) \gg V^{\pi}(s) + e_{\gamma}^{\pi}(s)$
- Thus, all values need to increase to  $\approx \rho^{\pi}/(1 - \gamma)$
- BUT: Greater  $V_{\gamma}^{\pi}(s)$ , then more likely to be picked

# Average Reward Adjusted Reinforcement Learning <sup>[4]</sup>

## Basic Idea

Separately assessing

- average reward  $\rho^\pi$
- bias value  $V^\pi(s)$



[4] Manuel Schneckeneither, "Average Reward Adjusted Discounted Reinforcement Learning: Near-Blackwell-Optimal Policies for Real-World Applications", 2020

# Average Reward Adjusted Reinforcement Learning [4]

## Basic Idea

Separately assessing

- average reward  $\rho^\pi$
- bias value  $V^\pi(s)$

## Algorithm

$$\rho^\pi \stackrel{\alpha}{\leftarrow} r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$$

$$V^\pi(s_t) \stackrel{\beta}{\leftarrow} r_t + \gamma V^\pi(s_{t+1}) - \rho^\pi$$

Note: We can set  $\gamma = 1$  here, as the subtraction of the average reward bounds the state values.

[4] Manuel Schneckeneither, "Average Reward Adjusted Discounted Reinforcement Learning: Near-Blackwell-Optimal Policies for Real-World Applications", 2020

## Discounted vs. Average Reward Adjusted RL

## Reconsider the 5x5 Gridworld [4]

Algorithm	Sum Reward	Avg. Steps
Avg. Rew. Adjusted $\gamma = 0.99$	<b>51894.094</b>	<b>5.039</b>
Avg. Rew. Adjusted $\gamma = 0.999$	51878.069	5.063
Avg. Rew. Adjusted $\gamma = 1.00$	51856.529	5.055
Standard Discounted $\gamma = 0.99$	34409.464	7661.833
Standard Discounted $\gamma = 0.999$	33931.917	7379.155
Standard Discounted $\gamma = 0.50$	30171.837	9999.000

(500k learning steps,  $40 \times 10k$  greedy evaluation steps)

[4] Manuel Schneckeneither, "Average Reward Adjusted Discounted Reinforcement Learning: Near-Blackwell-Optimal Policies for Real-World Applications", 2020

# Average Reward Adjusted Reinforcement Learning

## More Insights

Average Reward Adjusted Reinforcement Learning ...

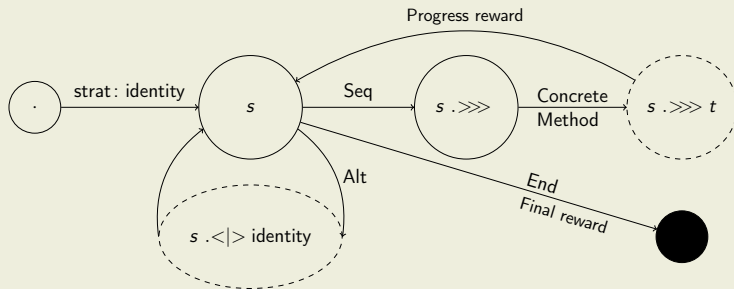
1. requires fewer number of learning steps
2. allows specifying continuous feedback





Implementation in  $TCT$ 

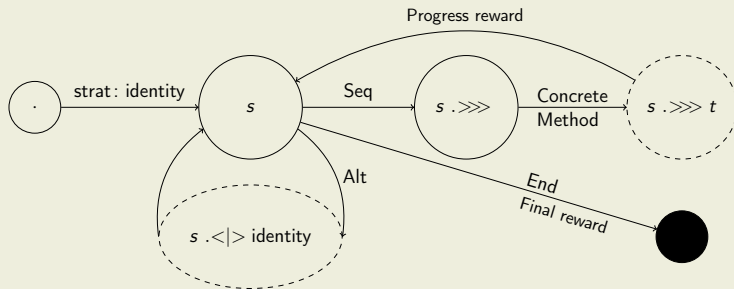
## Process Schema



- Starting strategy: identity
- Strategy combinators: Alternative and Sequence

Implementation in  $TCT$ 

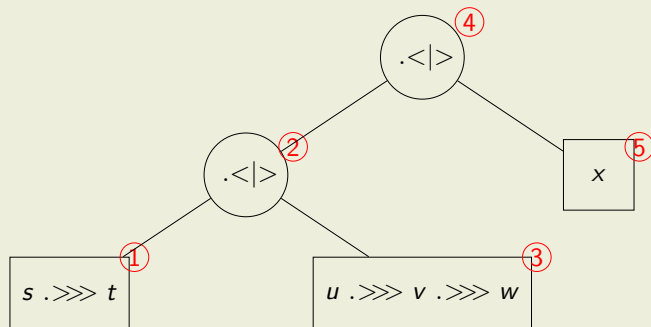
## Process Schema



- Note: Continuous feedback (Progress reward)
- Required: Exponential growth of solution space (continuous and faster learning)

Implementation in  $T_C T$ 

## Binary Strategy Tree



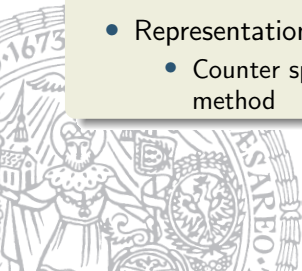
- No ambiguity by enforcing right-associativity of  $.<|>$
- Sorting of leaves on change of strategy:

$$(s .>>> t) .<|> (u .>>> v) \equiv (u .>>> v) .<|> (s .>>> t)$$

# Implementation in $T_C T$

## State Space

- Characteristics of input problem, e.g.
  - number of rules
  - number of (root) symbols
  - left-linearity, right-linearity
  - SCC (recursion and size of argument in recursive call)
  - branching degree
  - ...
- Representation of current strategy
  - Counter specifying steps until last occurrence of concrete method



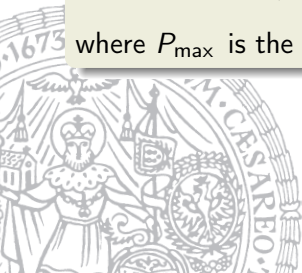
# Implementation in $T_C T$

## Reward Function

Assume current strategy  $s_t$  at step  $t$ , a timeout of  $T$  seconds, a measured execution time of  $t' \leq T$  seconds and resulting polynomial complexity  $C$ :

$$r(s_t) = \begin{cases} cT + t' & \text{if } C \text{ is } O(n^c) \text{ and } c < P_{\max} \\ P_{\max} T & \text{otherwise} \end{cases}$$

where  $P_{\max}$  is the maximum polynomial degree being considered.



# Outlook

## Reality everything is more complex

- Algorithm is more complex, e.g. incorporates the error term
- Artificial Neural Network (ANN) to approximate value function
- More strategy combinators than just  $.<|>$  and  $.>>>$
- ...



# Outlook

## Reality everything is more complex

- Algorithm is more complex, e.g. incorporates the error term
- Artificial Neural Network (ANN) to approximate value function
- More strategy combinators than just  $.<|>$  and  $.>>>$
- ...

## Current State (+ Expert info)

- Implemented to work with ANN (critic-only, experience replay memory, target+worker ANN, ...)
- We aim for an actor-critic approach (policy and value iteration)
- We plan to use a DB for caching (certain) I/O tuples
- Many open questions, as we are coding it right now

# Outlook

## Reality everything is more complex

- Algorithm is more complex, e.g. incorporates the error term
- Artificial Neural Network (ANN) to approximate value function
- More strategy combinators than just  $.<|>$  and  $.>>>$
- ...

## Current State (+ Expert info)

- Implemented to work with ANN (critic-only, experience replay memory, target+worker ANN, ...)
- We aim for an actor-critic approach (policy and value iteration)
- We plan to use a DB for caching (certain) I/O tuples
- Many open questions, as we are coding it right now



Thank you for your attention.

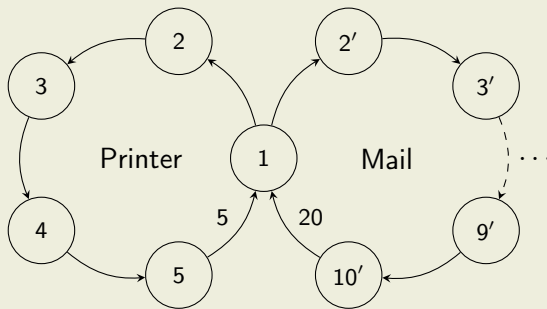
Let's discuss.

I am glad for any feedback, ideas or issues you have or see!





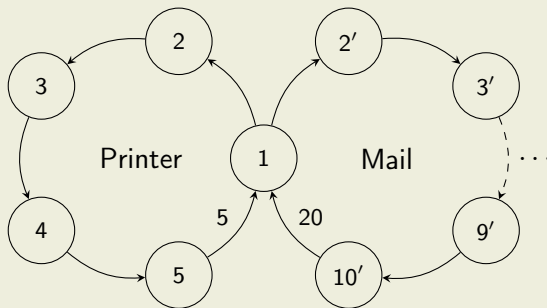
## The issue of discounted-optimality illustrated [5]



- Average reward received is
  - 1 for the printer-loop
  - 2 for the mail-loop

[5] Adapted from Mahadevan, "Optimality criteria in reinforcement learning", 1996

## The issue of discounted-optimality illustrated [5]

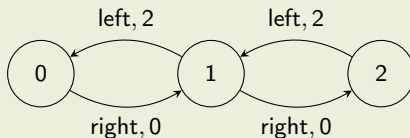


- Average reward received is
  - 1 for the printer-loop
  - 2 for the mail-loop
- BUT: if  $\gamma < 3^{-\frac{1}{5}} \approx 0.8027$  the agent prefers the printer loop

[5] Adapted from Mahadevan, "Optimality criteria in reinforcement learning", 1996

# Reinforcement Learning (RL)

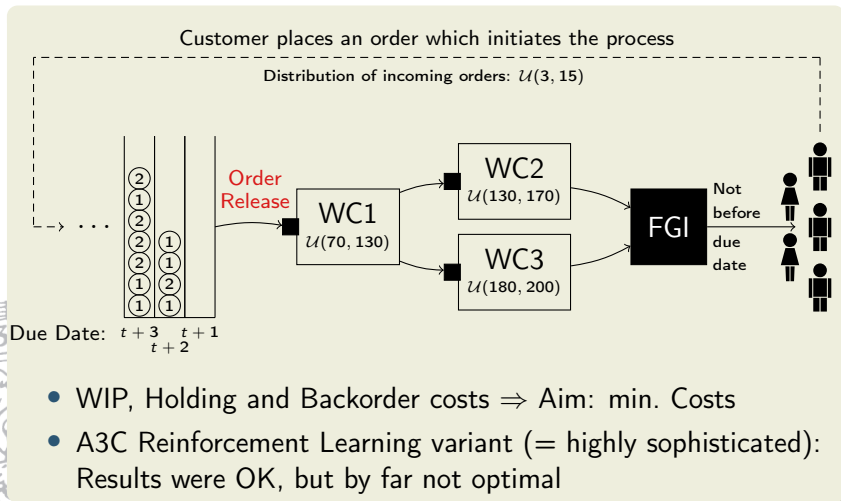
## Markov Decision Processes [6]



- Based on dynamic programming: states, actions, rewards
- Markov Prop.: transitions with probability  $p(s_{t+1}, r_t \mid s_t, a_t)$
- RL processes: *Markov decision processes* (MDPs)

[6] Miller and Veinott, "Discrete Dynamic Programming with a Small Interest Rate", 1969

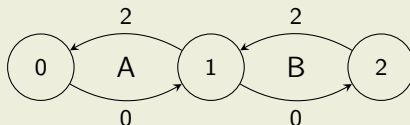
# Motivation: More Applications



M. Schneckenreither and Haeussler, "RL Methods for OR Applications", 2019

# Add: Normed state values have a higher spread (Slide 13)

## Reconsider the Example



- Both policies are have an average reward of 1 (gain-optimal).
- But only  $\pi_A$  is bias-optimal:  $V^{\pi^*}(s) - V^{\pi}(s) \geq 0$ .

$s$	$V^{\pi_A}(s)$	$V^{\pi_B}(s)$	$V_{0.99}^{\pi_A}$	$V_{0.99}^{\pi_B}$
0	-0.5	-1.5	99.4975	98.5025
1	0.5	-0.5	100.5025	99.4975
2	1.5	0.5	101.4975	100.5025