Reinforcement Learning for Dynamic Strategies in T$_C$T

Manuel Schneckenreither

(supervised by Prof. Dr. Georg Moser)

University of Innsbruck, Austria

CL Seminar   Summer 2020

Motivation

The Tyrolean Complexity Tool, i.e. $\mathrm{T_CT}$ [1]

- 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: $\mathrm{T_CT}$ outputs a certification proofing the presented asymptotic complexity

[1] Manuel Schneckenreither, "Dynamic Strategies for $\mathrm{T_CT}$", 2020
[2] Landi, "Undecidability of static analysis", 1992

Motivation

The Tyrolean Complexity Tool, i.e. $\mathsf{T}_{\mathsf{C}}\mathsf{T}$ [1]

- 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: $\mathsf{T}_{\mathsf{C}}\mathsf{T}$ outputs a certification proofing the presented asymptotic complexity

Static Analyses

- Static analyses are undecidable [2]
- Thus: Some sort of creativity is required (Strategies)
- Idea: Reinforcement learning for dynamic strategies

[1] Manuel Schneckenreither, "Dynamic Strategies for $\mathsf{T}_{\mathsf{C}}\mathsf{T}$", 2020
[2] Landi, "Undecidability of static analysis", 1992

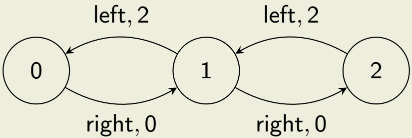
Reinforcement Learning for Dynamic Strategies in $T_C T$

Table of Contents

1. Motivation
2. Discounted Reinforcement Learning (Discounted RL)
3. Average Reward Adjusted Discounted RL
4. Proposed Implementation in $T_C T$

Reinforcement Learning (RL)

Reinforcement Learning Processes [3]

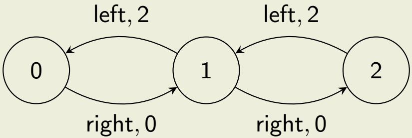


- Based on dynamic programming: states, actions, rewards

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

Reinforcement Learning (RL)

Reinforcement Learning Processes [3]



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

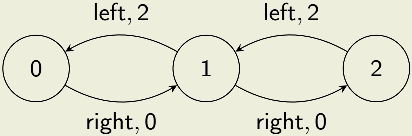
$$V_\gamma^\pi(s_t) \xleftarrow{\alpha} r_t + \gamma V_\gamma^\pi(s_{t+1})$$

($\xleftarrow{\alpha}$ is exp. smoothed updating with rate α)

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

Reinforcement Learning (RL)

Reinforcement Learning Processes [3]



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

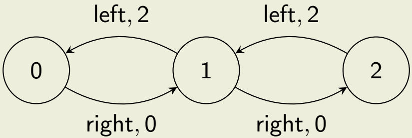
$$V^{\pi}_{\gamma}(s) = \lim_{N \to \infty} E[\sum_{t=0}^{N-1} \gamma^t R^{\pi}_t(s)]$$

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

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

Reinforcement Learning (RL)

Reinforcement Learning Processes [3]



- RL in summary: Iteratively solving constraint problems composed of $|\mathcal{S}|$ constraints and $|\mathcal{S}|$ variables, where $s \in \mathcal{S}$:

$$\begin{aligned} & V_\gamma^\pi(0) = 0 + \gamma V_\gamma^\pi(1) \qquad && V_\gamma^\pi(2) = 2 + \gamma V_\gamma^\pi(1) \\ \text{left}:\ & V_\gamma^\pi(1) = 2 + \gamma V_\gamma^\pi(0) \qquad \text{right}:\ && V_\gamma^\pi(1) = 0 + \gamma V_\gamma^\pi(2) \end{aligned}$$

($|\cdot|$ is the size of a set)

[3] 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 $\pi^\star$
- $\pi^\star$ maximises the state value for all states s as compared to any other policy π:

$$V_\gamma^{\pi^\star}(s) - V_\gamma^{\pi}(s) \geqslant 0$$

Aim in Discounted Reinforcement Learning

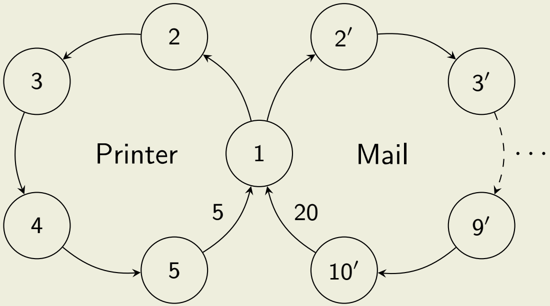
Aim of Discount Reinforcement Learning

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

$$V_\gamma^{\pi^\star}(s) - V_\gamma^{\pi}(s) \geqslant 0$$

- This criteria: discounted-optimality as γ is fixed

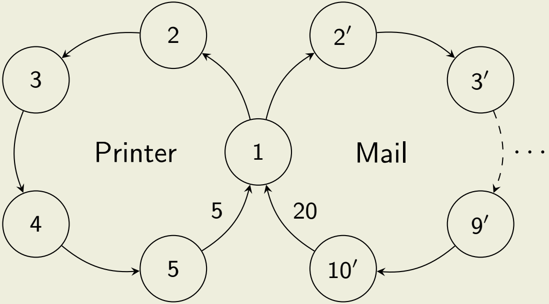
The issue of discounted-optimality illustrated [4]



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

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

The issue of discounted-optimality illustrated [4]



- 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

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

Laurent Series Expansion of Discounted State Values

Laurent Series Expansion of Discounted State Values[5]

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 ρ^π (simplified for unichain Processes)

The average reward is defined as

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

where R_t^π is the reward received at time t.

[5] 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[5]

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

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

Bias value $V^\pi(s)$

The bias value is defined as

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

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

[5] 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[5]

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_γ^π [6]

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

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

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

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

Laurent Series Expansion of Discounted State Values

Laurent Series Expansion of Discounted State Values[5]

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

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

[5] 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[5]

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 ∞ as $\gamma \to 1$
- Recall: RL is an iterative method

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

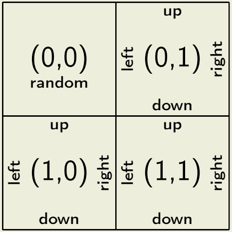
So What?

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

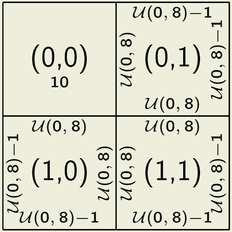
So What?

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

Consider this simple gridworld problem [6]



State/Action Space



Reward Function

[6] Manuel Schneckenreither, "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

(0,0) 101.260	23.113 67.160 (0,1) 61.783 37.331	162.243 49.812 (0,2) 349.515 153.653	354.839 306.761 (0,3) 399.560 351.907	398.391 399.376 (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

Basic Idea

Separately assessing

- average reward ρ^{π}
- bias value $V^{\pi}(s)$

Average Reward Adjusted Reinforcement Learning

Basic Idea

Separately assessing

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

Algorithm

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

$$V^\pi(s_t) \overset{\beta}{\leftarrow} r_t + \gamma_1 V^\pi(s_{t+1}) - \rho^\pi$$

Note: We can set $\gamma_1 = 1$ here, as the subtraction of the average reward bounds the state values.

Discounted vs. Average Reward Adjusted RL

Reconsider the 5x5 Gridworld [6]

Algorithm	Sum Reward	Avg. Steps
Avg. Rew. Adjusted $\gamma_1 = 0.99$	**51894.094**	**5.039**
Avg. Rew. Adjusted $\gamma_1 = 0.999$	51878.069	5.063
Avg. Rew. Adjusted $\gamma_1 = 1.00$	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)

[6] Manuel Schneckenreither, "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 ...

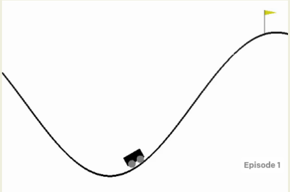
1. requires fewer number of learning steps
2. allows more natural specification of reward function (continuous feedback), which helps to
    - find the goal
    - distinguish between good and bad sequences of actions

Average Reward Adjusted Reinforcement Learning

More Insights

Average Reward Adjusted Reinforcement Learning . . .

1. requires fewer number of learning steps
2. allows more natural specification of reward function (continuous feedback), which helps to
   - find the goal
   - distinguish between good and bad sequences of actions



Source: https://gym.openai.com/

Average Reward Adjusted Reinforcement Learning

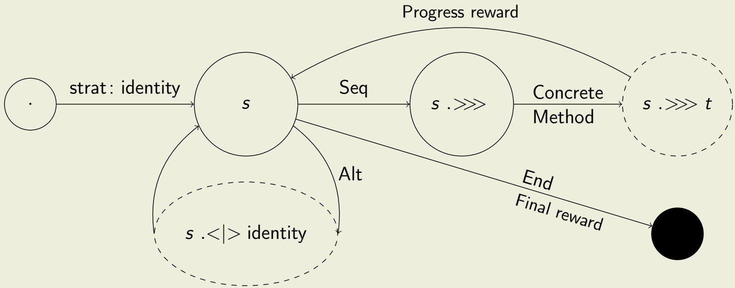
More Insights (cont'd)

Average Reward Adjusted Reinforcement Learning ...

3. state values are easier to interpret
4. normed state values have a higher spread
    - important when approximating the state-value function (ANN)

Implementation in T_CT

Process Schema [7]

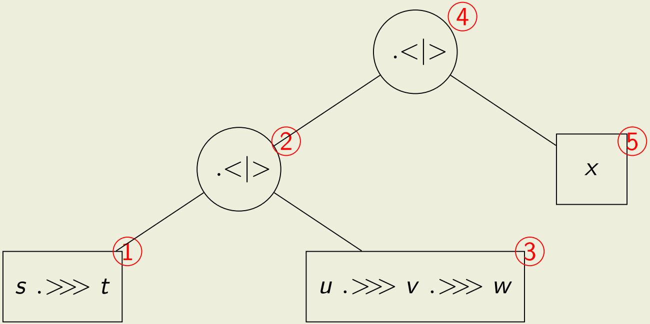


- Starting strategy: identity
- Strategy operators: Sequencing .>>> and alternative .<|>
- Two agents: decomposition and complexity analyser

[7] Manuel Schneckenreither, "Dynamic Strategies for T_CT", 2020

Implementation in T_CT

Binary Strategy Tree [8]



- No ambiguity by enforcing right-associativity of .<|>

[8] Manuel Schneckenreither, "Dynamic Strategies for T_CT", 2020

Implementation in T_CT

State Space

- Characteristics of input problem, e.g.
  - number of rules
  - number of (root) symbols
  - left-linearity, right-linearity
- Representation of current strategy
  - Counter specifying steps until last occurrence of concrete method

[8] Manuel Schneckenreither, "Dynamic Strategies for T_CT", 2020

Conclusion

Summary

- Discounted Reinforcement Learning
- Average Reward Adjusted Discounted Reinforcement Learning
- Implementation Idea for Dynamic Strategies in $\mathrm{T_CT}$

Conclusion

Summary

- Discounted Reinforcement Learning
- Average Reward Adjusted Discounted Reinforcement Learning
- Implementation Idea for Dynamic Strategies in T$_C$T

Thank you for your attention!

Implementation in T_CT

Reward Function [9]

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

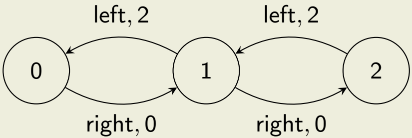
$$r(s_t) = \begin{cases} cT + t' & \text{if } C \text{ is } \mathrm{O}(n^c) \text{ and } c < P_{\max} \\ P_{\max} T & \text{otherwise} \end{cases}$$

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

[9] Manuel Schneckenreither, "Dynamic Strategies for T_CT", 2020

Reinforcement Learning (RL)

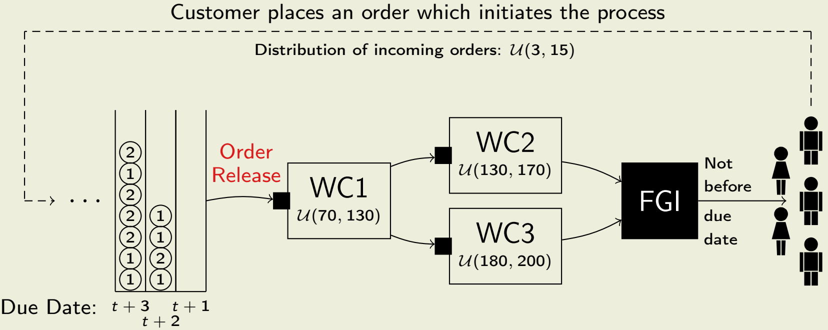
Markov Decision Processes [10]



- 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)

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

Motivation: More Applications

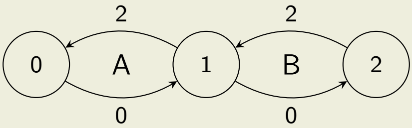


- WIP, Holding and Backorder costs $\Rightarrow$ Aim: min. Costs
- A3C Reinforcement Learning variant (= highly sophisticated): Results were OK, but by far not optimal

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 π_A is bias-optimal: $V^{\pi^\star}(s) - V^{\pi}(s) \geqslant 0$.

s	$V^{\pi_A}(s)$	$V^{\pi_B}(s)$	$V^{\pi_A}_{0.99}$	$V^{\pi_B}_{0.99}$
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