Reinforcement learning in search of optimal premium rules

Lina Palmborg, Stockholm University

based on joint work with F. Lindskog

June 16, 2023

Premium control problem for a mutual insurer

- Premium control problem in discrete time
- Non-life insurer
  - Delays between accidents and payments
  - Premium level affects whether the company attracts or loses customers (feedback)

Premium control problem for a mutual insurer

- Premium control problem in discrete time
- Non-life insurer
  - Delays between accidents and payments
  - Premium level affects whether the company attracts or loses customers (feedback)
- Mutual insurer
  - Aim: find a premium rule that generates a low, stable premium, and a low probability of default

Model of the insurance company

- Insurance economics give surplus fund dynamics

$$G_{t+1} = G_t + \mathsf{EP}_{t+1} + \mathsf{IE}_{t+1} - \mathsf{OE}_{t+1} - \mathsf{IC}_{t+1} + \mathsf{RP}_{t+1}$$

Model of the insurance company

- Insurance economics give surplus fund dynamics

$$G_{t+1} = G_t + \mathsf{EP}_{t+1} + \mathsf{IE}_{t+1} - \mathsf{OE}_{t+1} - \mathsf{IC}_{t+1} + \mathsf{RP}_{t+1}$$

- Earned premium: $\mathsf{EP}_{t+1} = (P_t N_{t+1} + P_{t-1} N_t)/2$

Model of the insurance company

- Insurance economics give surplus fund dynamics

$$G_{t+1} = G_t + \mathsf{EP}_{t+1} + \mathsf{IE}_{t+1} - \mathsf{OE}_{t+1} - \mathsf{IC}_{t+1} + \mathsf{RP}_{t+1}$$

- Earned premium: $\mathsf{EP}_{t+1} = (P_t N_{t+1} + P_{t-1} N_t)/2$

- Define the state S_t so that (S_t) is Markovian given the premium rule (policy) π, e.g.

$$S_t = (G_t, P_{t-1}, N_t, \ldots)$$

$\implies$ Markov decision process (MDP)

The control problem

$$\underset{\pi}{\text{minimise}}\ \mathbb{E}_{\pi}\Big[\sum_{t=0}^{T}\gamma^t f(P_t, S_t, S_{t+1}) \mid S_0 = s\Big],$$

where γ is a discount factor.

The control problem

$$\underset{\pi}{\text{minimise}}\ \mathbb{E}_\pi\Big[\sum_{t=0}^{T}\gamma^t f(P_t, S_t, S_{t+1}) \mid S_0 = s\Big],$$

where γ is a discount factor.

$$f(P_t, S_t, S_{t+1}) := \begin{cases} c(P_t), & \text{if } G_{t+1} \geq G_{\min}, \\ c(\max \mathcal{A})(1+\eta), & \text{if } G_{t+1} < G_{\min}, \end{cases}$$

- c an increasing, strictly convex function $\implies$ premiums (P_t) will be averaged
- $T := \min\{t : G_{t+1} < G_{\min}\} \implies$ termination (default)
- $\eta > 0 \implies$ high cost in case of default

Solving the control problem

- Explicit transition probabilities not available in a realistic setting
  $\Longrightarrow$ cannot use dynamic programming
  $\Longrightarrow$ need to use reinforcement learning (e.g. SARSA)

Solving the control problem

- Explicit transition probabilities not available in a realistic setting
  $\implies$ cannot use dynamic programming
  $\implies$ need to use reinforcement learning (e.g. SARSA)
- State space too large in a realistic setting
  $\implies$ need to use function approximation

Solving the control problem

- Explicit transition probabilities not available in a realistic setting
  $\Longrightarrow$ cannot use dynamic programming
  $\Longrightarrow$ need to use reinforcement learning (e.g. SARSA)
- State space too large in a realistic setting
  $\Longrightarrow$ need to use function approximation
- SARSA learns from real or simulated experience
  $\Longrightarrow$ need a lot of data!
  $\Longrightarrow$ simulate data from a suitable stochastic environment

SARSA with function approximation

- The action-value function

$$q_\pi(s,a) := \mathbb{E}_\pi\Big[\sum_{t=0}^{T} \gamma^t(-f(P_t, S_t, S_{t+1})) \mid S_0 = s, P_0 = a\Big]$$

is approximated by a parameterised function $\hat{q}(s,a;\theta)$

SARSA with function approximation

- The action-value function

$$q_\pi(s,a) := \mathbb{E}_\pi\Big[\sum_{t=0}^{T} \gamma^t(-f(P_t, S_t, S_{t+1})) \mid S_0 = s, P_0 = a\Big]$$

  is approximated by a parameterised function $\hat{q}(s,a;\theta)$
- Given a behaviour policy π that generates actions we can sample $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$
  - Action A_t (here premium P_t)
  - Reward R_{t+1} (here $-f(P_t, S_t, S_{t+1})$)

SARSA with function approximation

- The action-value function

$$q_\pi(s,a) := \mathbb{E}_\pi\Big[\sum_{t=0}^{T}\gamma^t(-f(P_t,S_t,S_{t+1})) \mid S_0=s, P_0=a\Big]$$

  is approximated by a parameterised function $\hat{q}(s,a;\theta)$
- Given a behaviour policy π that generates actions we can sample $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$
  - Action A_t (here premium P_t)
  - Reward R_{t+1} (here $-f(P_t,S_t,S_{t+1})$)
- Iterative update for the weight vector

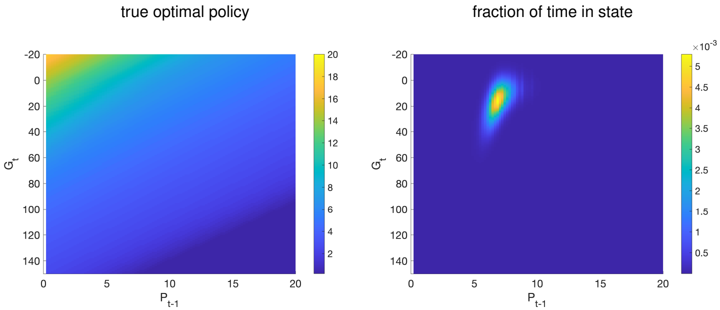
$$\theta_{t+1} = \theta_t + \alpha_{t+1}\big(R_{t+1} + \gamma\hat{q}(S_{t+1},A_{t+1};\theta_t) - \hat{q}(S_t,A_t;\theta_t)\big)\nabla\hat{q}(S_t,A_t;\theta_t)$$

Illustration - simple model

- N_t fixed, finite state space, state $S_t = (G_t, P_{t-1})$
- $\implies$ can be solved by dynamic programming

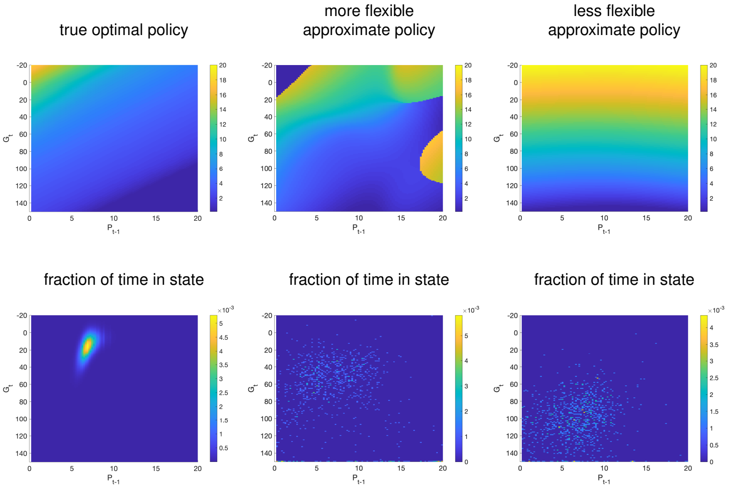
Illustration - simple model

- N_t fixed, finite state space, state $S_t = (G_t, P_{t-1})$
- $\implies$ can be solved by dynamic programming

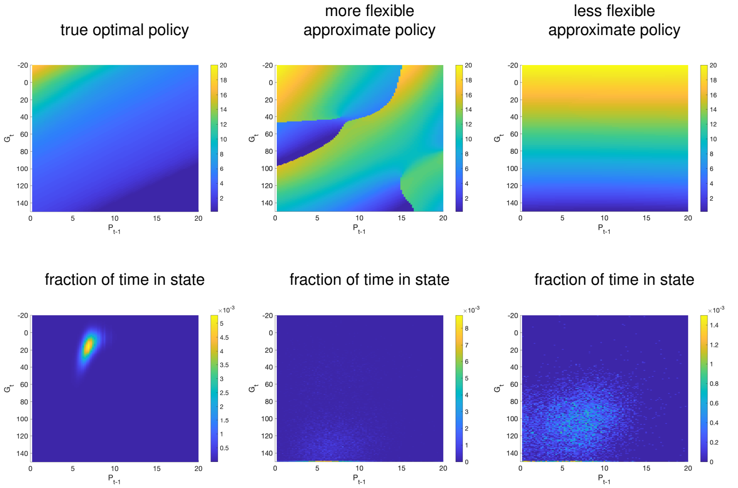


Function approximation

Number of episodes: 10 $\quad$ (episode length $= \min\{100, T\}$)

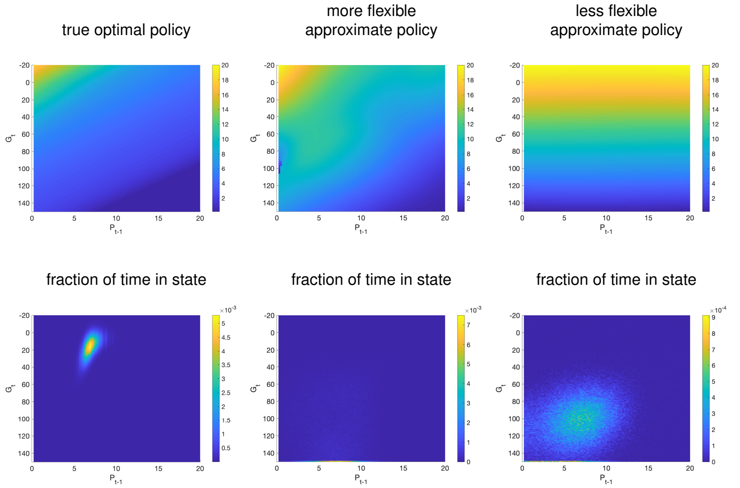


Number of episodes: 100



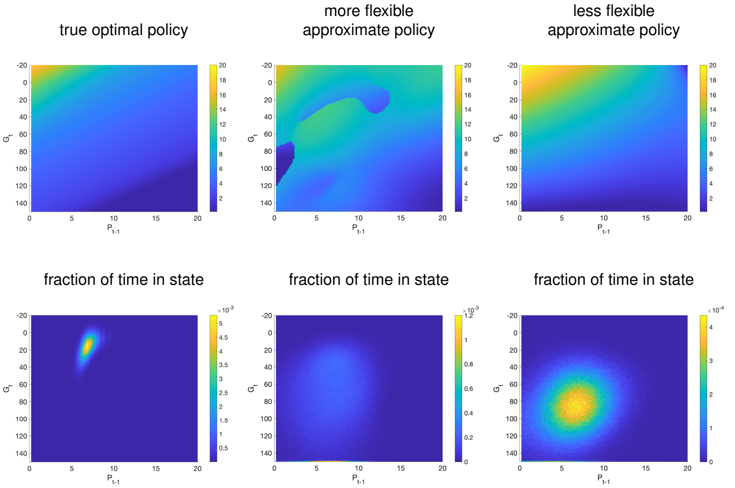
Function approximation

Number of episodes: 1000



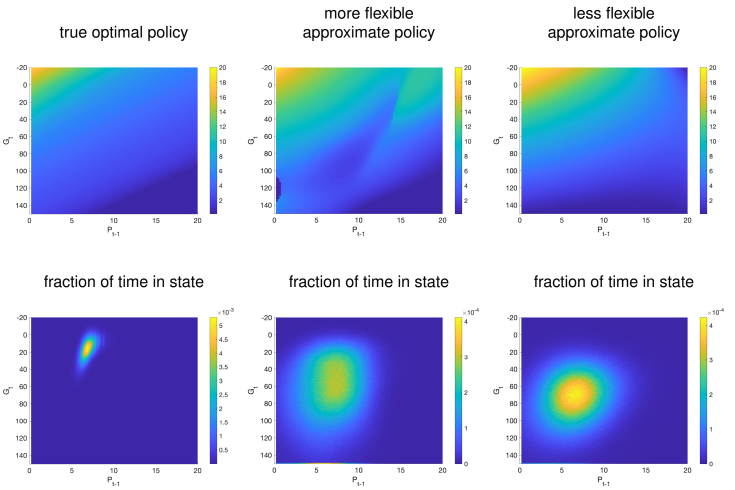
Function approximation

Number of episodes: 10 000



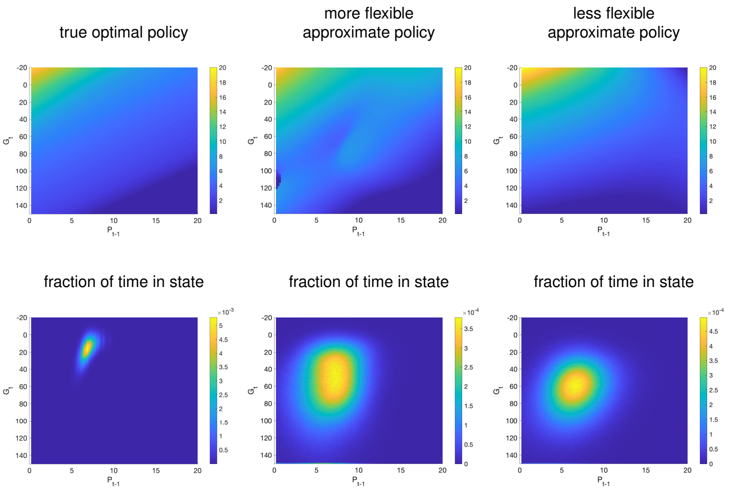
Function approximation

Number of episodes: 50 000



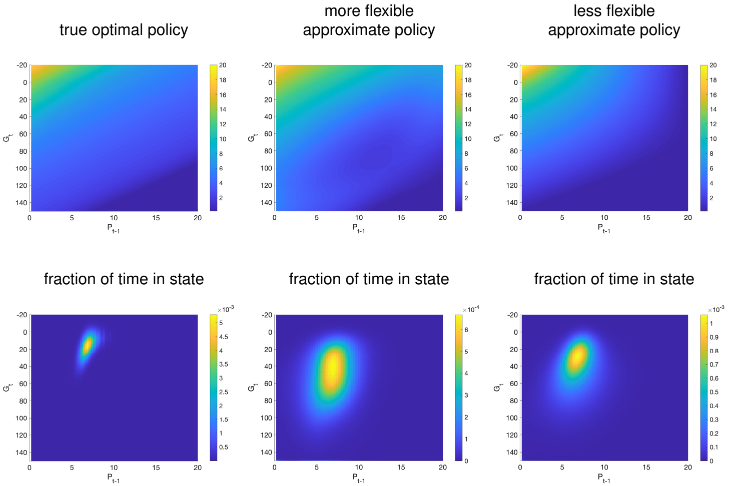
Function approximation

Number of episodes: 100 000



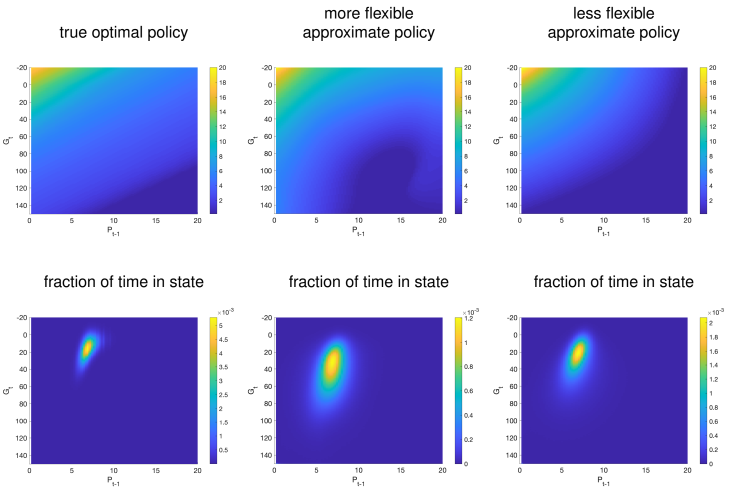
Function approximation

Number of episodes: 300 000



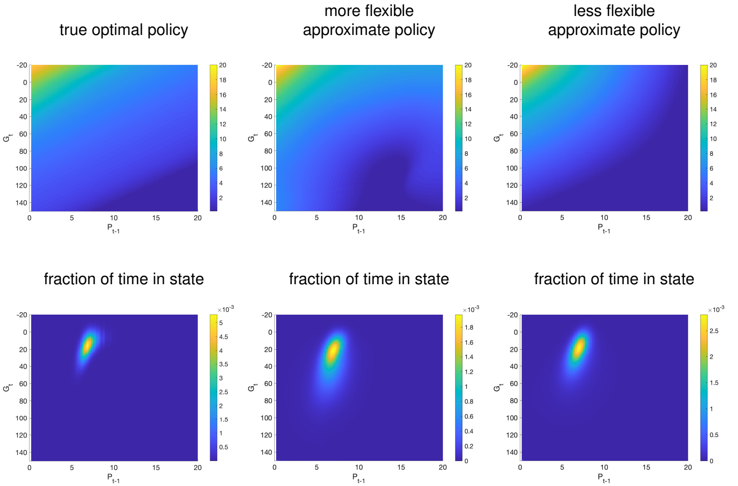
Function approximation

Number of episodes: 600 000



Function approximation

Number of episodes: 1 000 000



Behaviour policy

- Needs to both explore and exploit

Behaviour policy

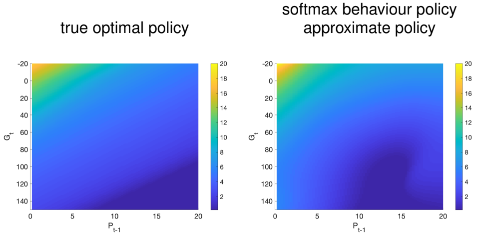
- Needs to both explore and exploit
  - ε-greedy policy: $\pi(a|s) = \begin{cases} 1-\varepsilon, & \text{if } a = \underset{a}{\operatorname{argmax}}\, \hat{q}(s,a;\theta), \\ \dfrac{\varepsilon}{|\mathcal{A}|-1}, & \text{otherwise.} \end{cases}$

Behaviour policy

- Needs to both explore and exploit
  - ε-greedy policy: $\pi(a|s) = \begin{cases} 1-\varepsilon, & \text{if } a = \underset{a}{\operatorname{argmax}}\, \hat{q}(s,a;\theta), \\ \dfrac{\varepsilon}{|\mathcal{A}|-1}, & \text{otherwise.} \end{cases}$
  - Softmax policy: $\pi(a|s) = \dfrac{\exp\{\hat{q}(s,a;\theta)/\tau\}}{\sum_{\bar{a}\in\mathcal{A}} \exp\{\hat{q}(s,\bar{a};\theta)/\tau\}}$

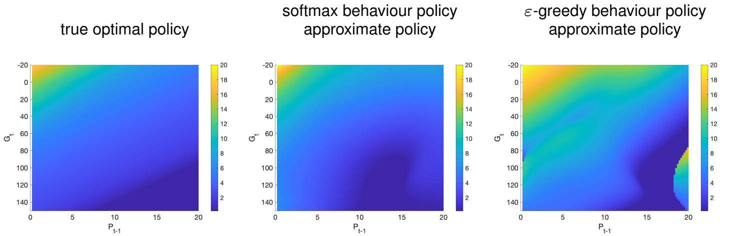
Behaviour policy

- Needs to both explore and exploit
  - ε-greedy policy: $\pi(a|s) = \begin{cases} 1-\varepsilon, & \text{if } a = \operatorname{argmax}_a \hat{q}(s,a;\theta), \\ \frac{\varepsilon}{|\mathcal{A}|-1}, & \text{otherwise.} \end{cases}$
  - Softmax policy: $\pi(a|s) = \dfrac{\exp\{\hat{q}(s,a;\theta)/\tau\}}{\sum_{\bar{a}\in\mathcal{A}} \exp\{\hat{q}(s,\bar{a};\theta)/\tau\}}$



Behaviour policy

- Needs to both explore and exploit
  - ε-greedy policy: $\pi(a|s) = \begin{cases} 1-\varepsilon, & \text{if } a = \operatorname{argmax}_a \hat{q}(s,a;\theta), \\ \dfrac{\varepsilon}{|\mathcal{A}|-1}, & \text{otherwise.} \end{cases}$
  - Softmax policy: $\pi(a|s) = \dfrac{\exp\{\hat{q}(s,a;\theta)/\tau\}}{\sum_{\bar{a}\in\mathcal{A}} \exp\{\hat{q}(s,\bar{a};\theta)/\tau\}}$



Realistic model

- In more realistic settings, we derive approximate optimal premium rules that outperform several benchmark policies

- For more details on this, and the full design of the reinforcement learning algorithm, see

  L. Palmborg, F. Lindskog (2023), Premium control with reinforcement learning. *ASTIN Bulletin*. Open access
  `https://doi.org/10.1017/asb.2023.13`

Main references

- A. Martin-Löf (1983), Premium control in an insurance system, an approach using linear control theory. *Scandinavian Actuarial Journal*, 1983(1):1–27.
- A. Martin-Löf (1994), Lectures on the use of control theory in insurance. *Scandinavian Actuarial Journal*, 1994(1):1–25.
- R. S. Sutton and A. G. Barto (2018), *Reinforcement learning: An introduction*. MIT press.
- A. Benveniste, M. Métivier, and P. Priouret (1990), *Adaptive algorithms and stochastic approximations*. Springer.
- F. S. Melo, S. P. Meyn, and M. I. Ribeiro (2008), An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th international conference on Machine learning*, pages 664-671.
- T. J. Perkins and D. Precup (2003), A convergent form of approximate policy iteration. In *Advances in neural information processing systems*, pages 1627-1634.