Autonomous Control of a Particle Accelerator using Deep Reinforcement Learning

Xiaoying Pang, Sunil Thulasidasan, Larry Rybarcyk

Introduction

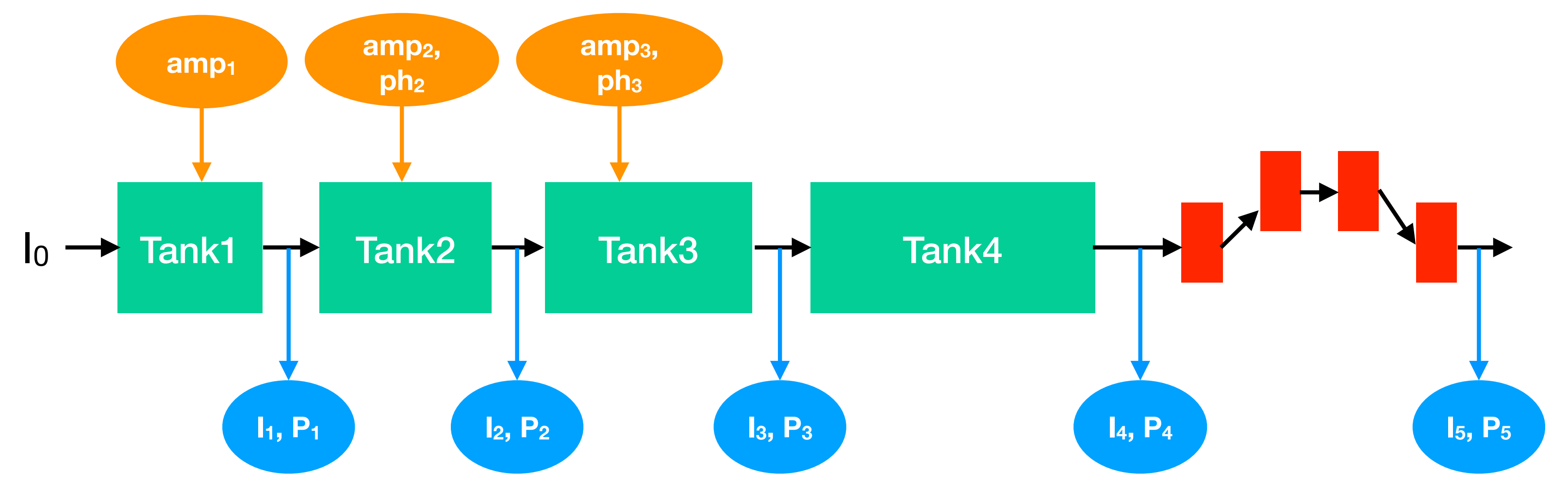
Large particle accelerators are complex systems. LANSCE proton linear accelerator :

- 800 meters
- ~5000 accelerating structures
- ~400 focusing structures
- ~200 free tuning parameters

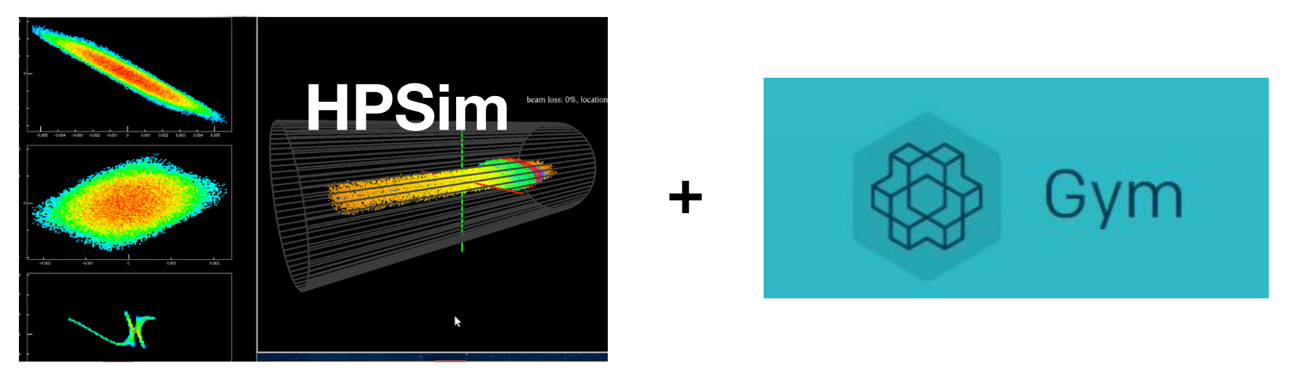
Because of the high-dimensional parameter space and dynamic nature, it is very challenging to tune up a large accelerator from scratch. In this work, we explore an approach to learn optimal accelerator control policies using deep reinforcement learning coupled with a high-fidelity physics engine.

Problem statement

As a proof-of-concept, we only focus on controlling a small section of the entire accelerator. We adjust the control parameters of the first three radio frequency tanks of the LANSCE linear accelerator to let as much beam through as possible.



Schematics of the beam line, action variables (orange) and states (blue).



Markov Decision Process

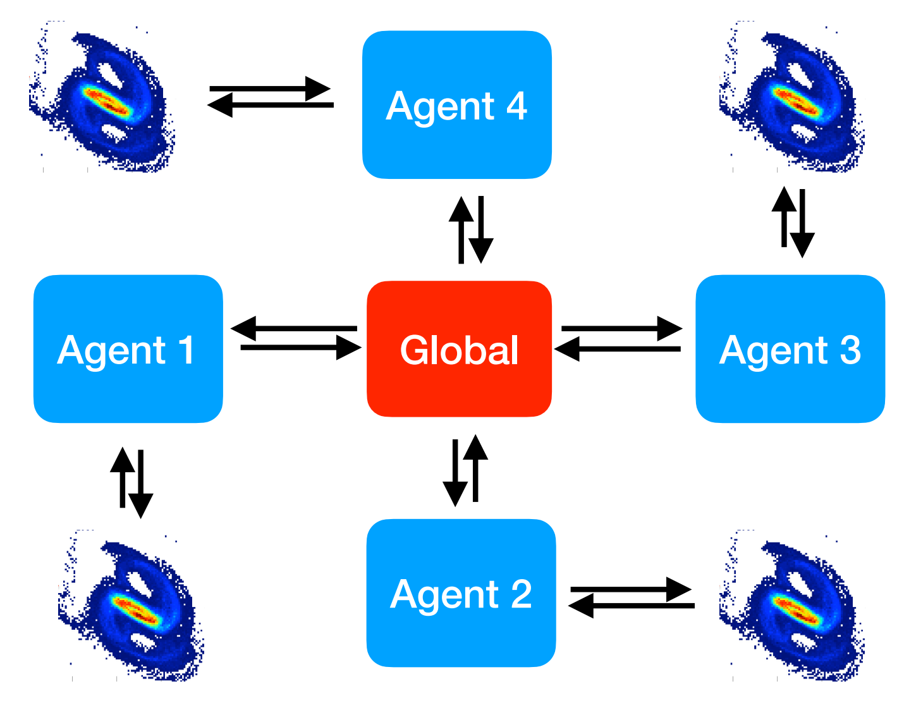
- Action: tank 1 amplitude, tank 2, 3 amplitude & phase. Continuous and bounded (with min & max values).
- State: beam current and power of lost particles at 5 locations. Continuous.
- Reward: (goal: >=85% of beam make it to the end, minimum radiation)

$$r = \begin{cases} +1000, & \text{if } I[5]/I_0 >= 0.85. \\ -\sum_{n=1}^{5}[A_n \cdot \min(0, I[n]/I_0 - 0.85)^2 + B \cdot P[n]^2], & \text{otherwise.} \end{cases}$$

$I[n]$ is current at the nth monitor. $P[n]$ is power of the lost beam at the nth monitor.

Method

We adopt Asynchronous Advantage Actor-Critic (A3C). It's a multi-agent system. Each agent holds a copy of the actor-critic model and the physics engine (HPSim) and explores a different path/trajectory to find an optimal policy. A global model that serves as the accumulated knowledge of the entire system is maintained and updated asynchronously by all the agents.

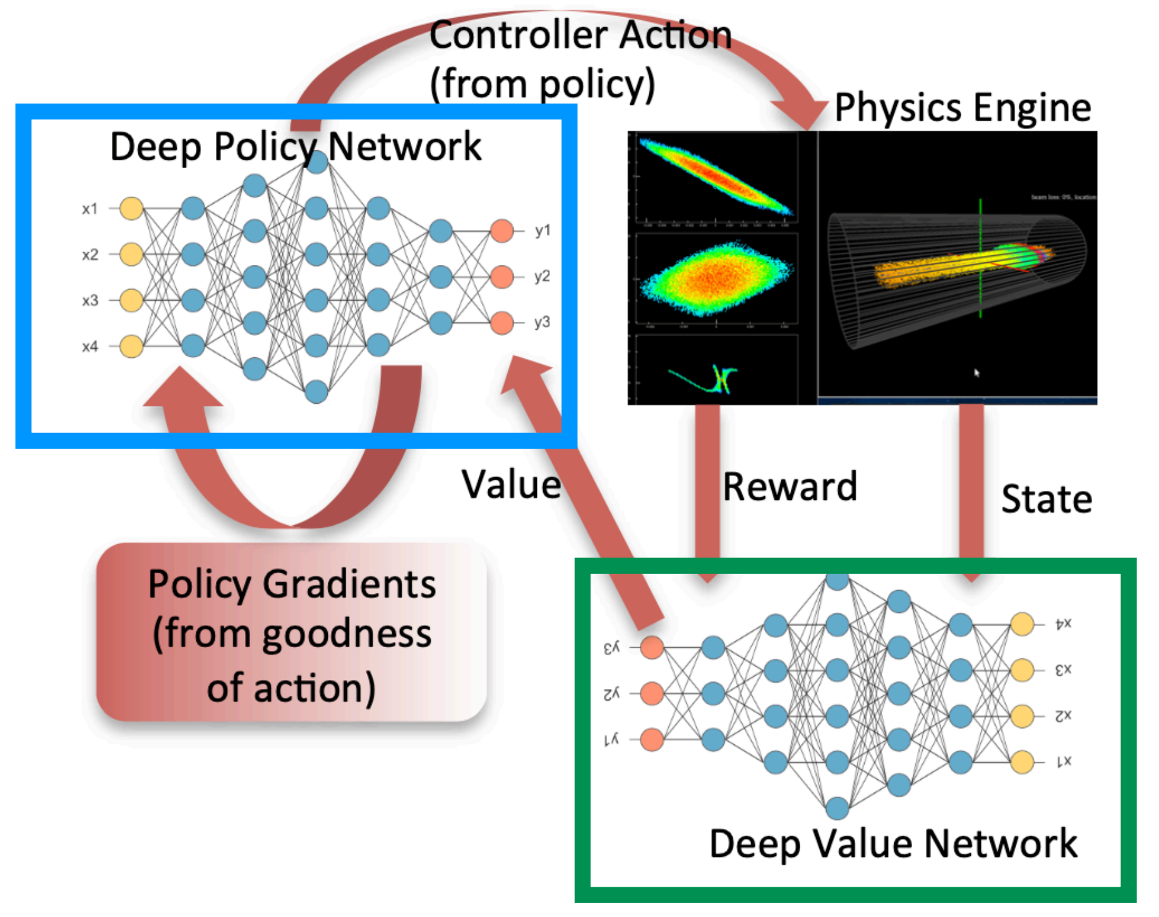


Actor-Critic + Asynchronous updates from multiple agents

$$a_t \sim \mathcal{N}(\mu_t, \sigma_t)$$

Actor/policy

$$\mu_t, \sigma_t \leftarrow f_\theta(s_t)$$



Critic/value

$$V^\theta(s_t) \leftarrow f_\phi(s_t)$$

$$\textbf{PLoss} = -\log \pi_\theta(a_t|s_t)A^{\pi_\theta}(s_t, a_t) + C_1 \cdot \textbf{actionPenalty}$$

$$\textbf{VLoss} = \frac{1}{2}\|A^{\pi_\theta}(s_t, a_t))\|^2$$

$$A^{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$$

$$\textbf{TotalLoss} = \textbf{PLoss} + C \cdot \textbf{VLoss}$$

A closer look at the Actor-Critic model held by each agent

References

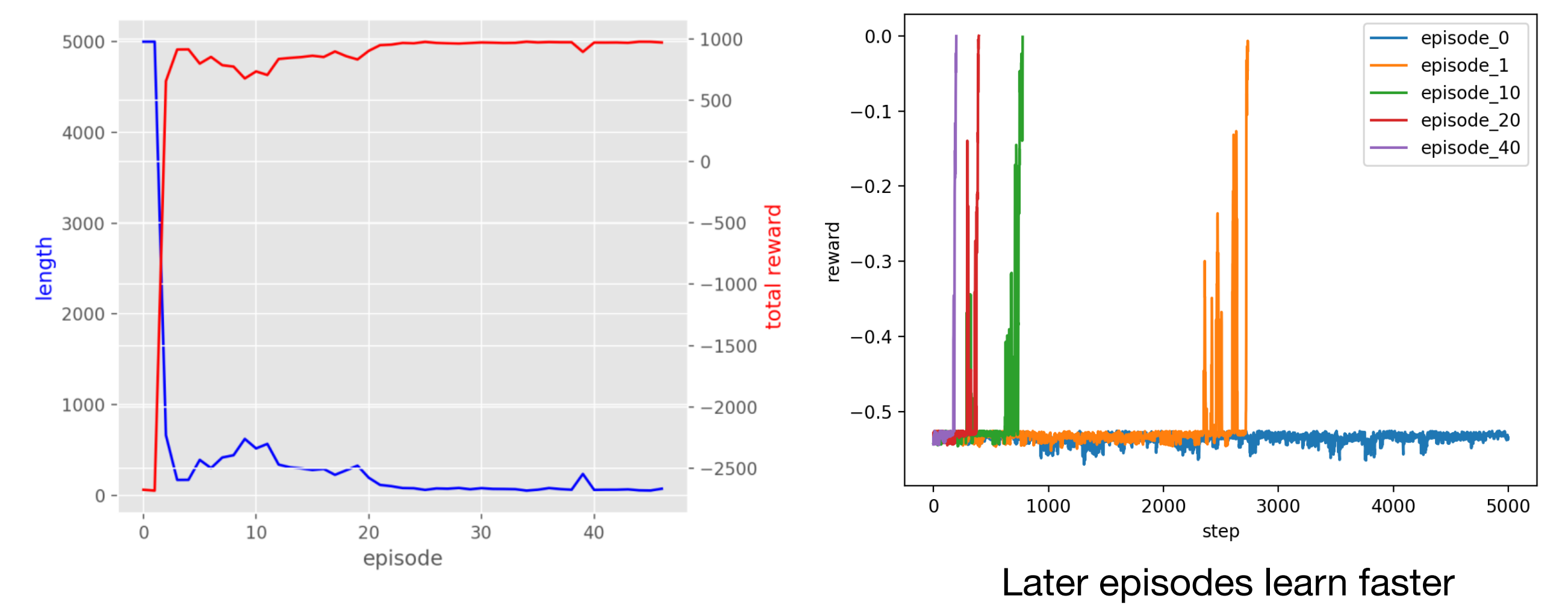
HPSim. https://github.com/apphys/hpsim

OpenAI Gym. https://gym.openai.com/.

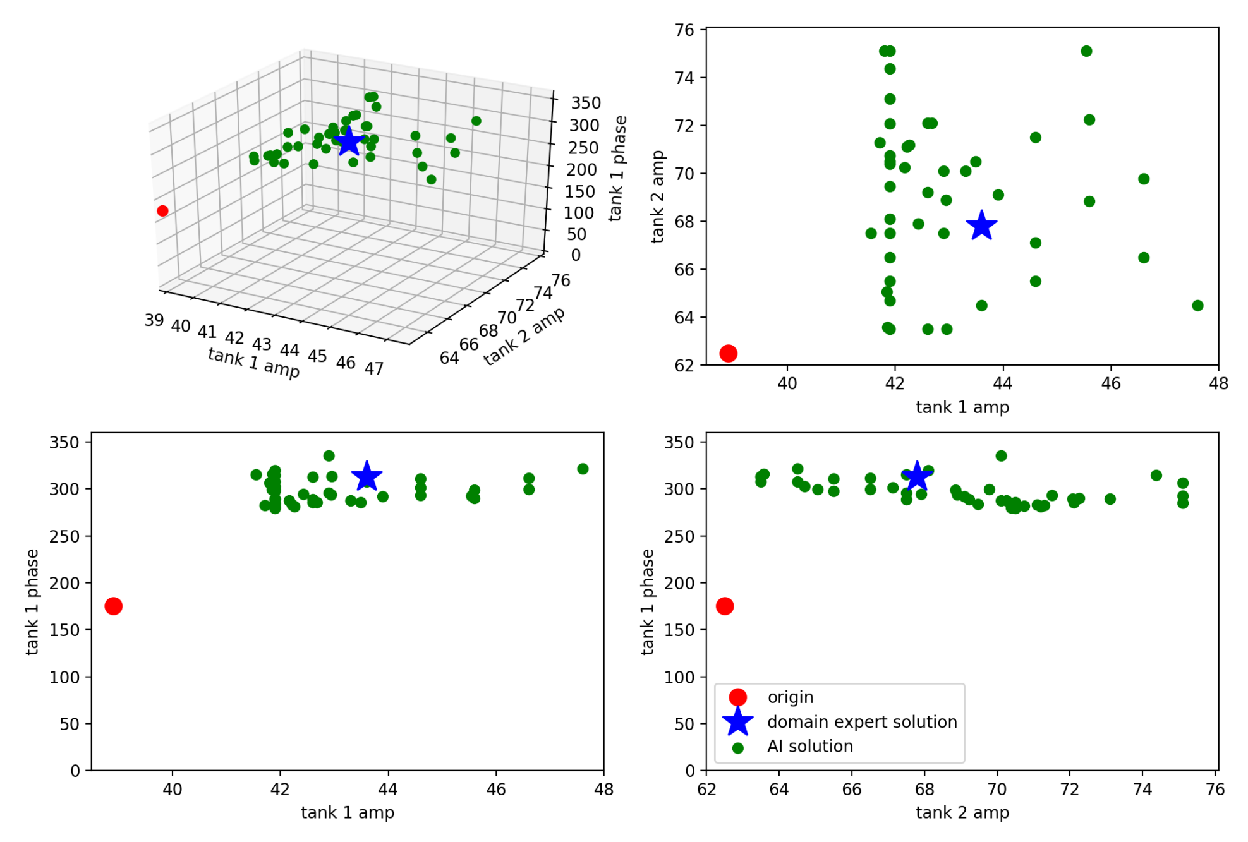
V. Mnih et al. Asynchronous methods for deep reinforcement learning. ICML, 48:1928–1937, 2016.

Results

Tuning 3 action variables

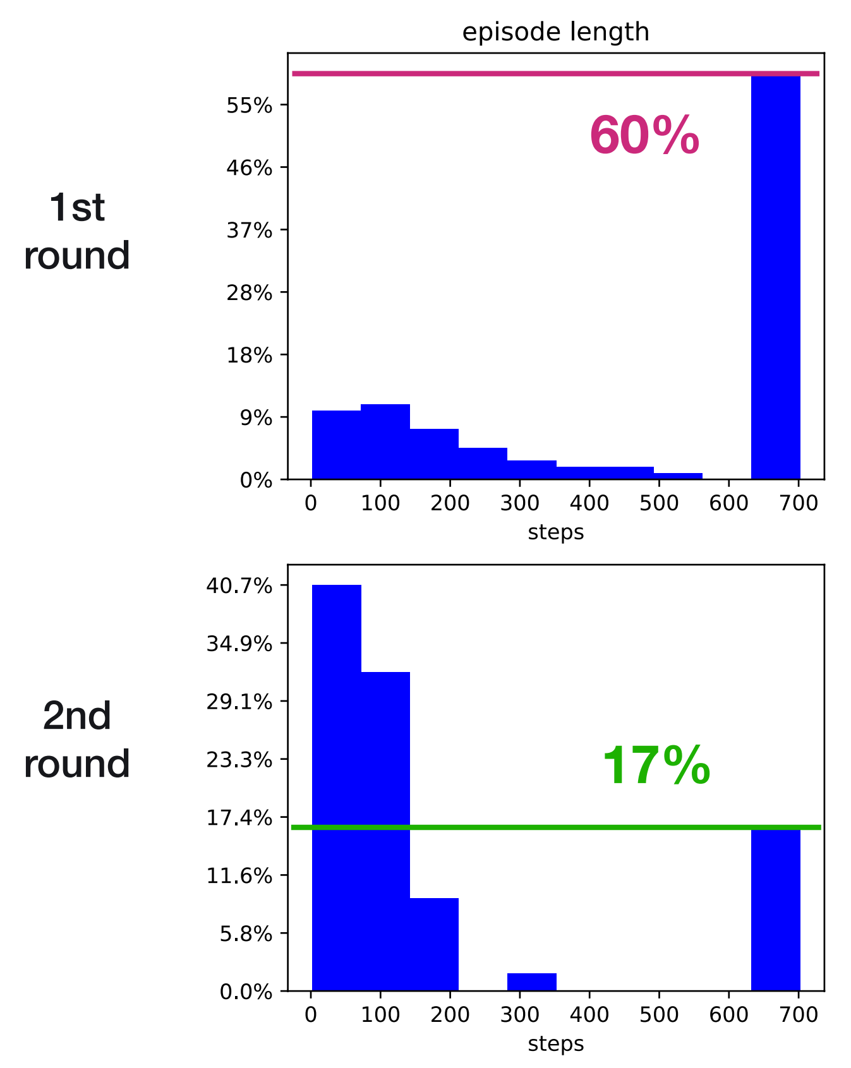


Later episodes learn faster

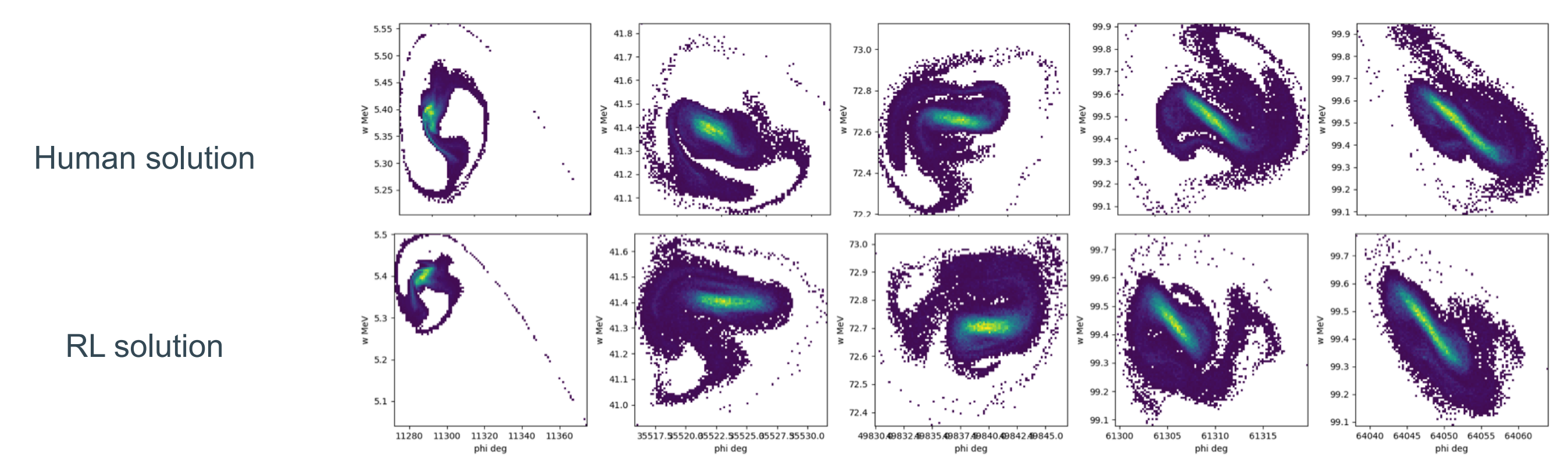


Due to the non-linearity of the physical process, there could be multiple optimal operating settings. The RL agent is able to discover many of them including the ones that are close to the domain expert's. Red start: starting point of the tuning process. Blue star: domain expert's solution. Green dots: solutions found by the RL agent.

Tuning 5 action variables



Steps to successfully tune up the machine. If the agent cannot succeed within 700 steps, we declare the tuning episode a failure. We observe a 60% failure rate, after training the agents for one iteration. The failure rate drop significantly to 17% after just another around of training. That means the agent now can tuning up the machine from any initial point with 83% of success rate.


Human solution
RL solution