# Strategyproof reinforcement learning for online resource allocation

Ioana Moisoiu, Sebastian Stein (Supervisor)

UNIVERSITY OF Southampton

## Problem description

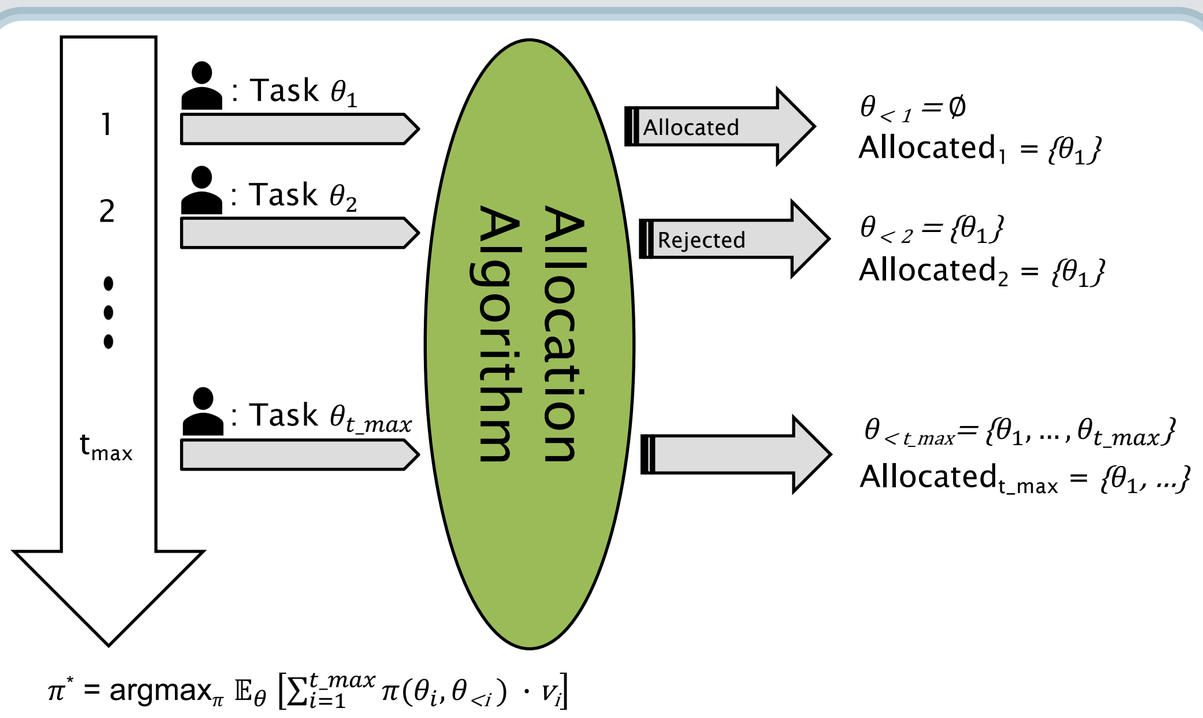
- This is an extension of previous work about online mechanism design using reinforcement learning for cloud resource allocation (Ochal et al., 2018). We present an online solution to the problem;
- The problem:
  - agents relay tasks to a system with a finite number of resources. The system can allocate or drop tasks depending on its state;
  - due to scarcity of resources, agents can strategically manipulate tasks in order to increase the chances of allocation for their tasks;
- The problem is of high importance in settings where tasks are urgent or expire quickly and resources are limited.

## Formal description of the System Model

- The setting is bounded by a time horizon $t_{max}$ and presupposes discrete time steps $T = \{1, 2, \dots, t_{max}\}$;
- The system is defined by a finite number of different resource types $R = \{1, 2, \dots, r_{max}\}$ and $r \in \mathbb{R}, \forall r \in R$. Each resource type $r$ is determined by a limited quantity of resource units $a_r \in \mathbb{N}^0$;
- Each task $i$ is defined by a type $\theta_i = (v_i, d_i, q_i)$, where:
  - $v_i$ is the value of the task;
  - $d_i$ represents the number of time steps in which the the task will be computed;
  - $q_i = [q_{i,1}, q_{i,2}, \dots, q_{i,rmax}]^T \in \mathbb{R}^{r_{max}}$ is a vector representation of the number of units required for the task from each resource type;
- For current time step $i$ and current task to be considered $\theta_i$ we denote by $\theta_{<i}$ all the tasks that arrived before time step $i$. By $Allocated_i$ we denote all the tasks that have been allocated to resources up until and including the current time step.

## Objective

- Design an online algorithm using reinforcement learning that is strategyproof. The algorithm must learn an optimal policy $\pi^*$ for allocating tasks with respect to an initial allocation policy $\pi$ which allocates or rejects a task at each time step. The optimal policy should maximize the expected utility of the allocated tasks.




$\pi^* = \text{argmax}_\pi \, \mathbb{E}_\theta \left[\sum_{i=1}^{t\_max} \pi(\theta_i, \theta_{<i}) \cdot v_i\right]$

## Solution

- Our solution uses an on-policy algorithm (SARSA) with a linear function approximation for the value function. As this is a dynamic setting, using SARSA has proven to generate good results faster, with less amount of experience then Q-learning would require for convergence to an optimal policy.

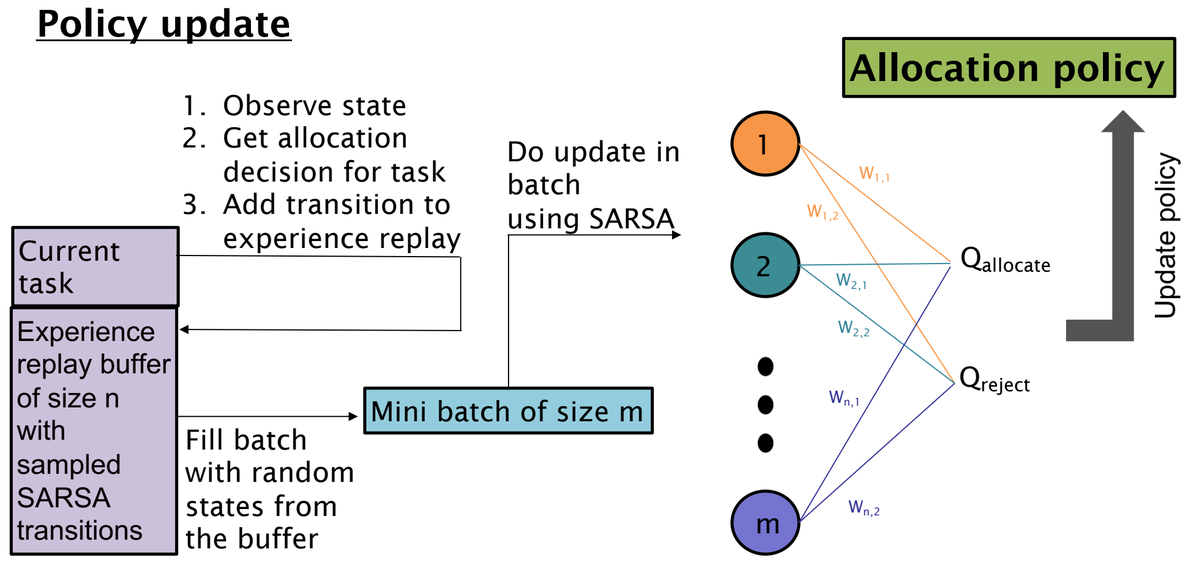
## The Current Reinforcement Learning Algorithm

- States: combine information about the task type and the system's state at the current time step, as well as a linear combination of features
- Actions: allocate or reject
- Reward: value of the allocated task
- Applying SARSA with linear approximation for the value function to which mini-batch gradient descent with experience replay can be applied when updating the value function. In addition, monotonic constraints are added when updating the value function in order to avoid manipulation of the learnt allocation algorithm by any possible misreporting of the tasks.

## Action Selection

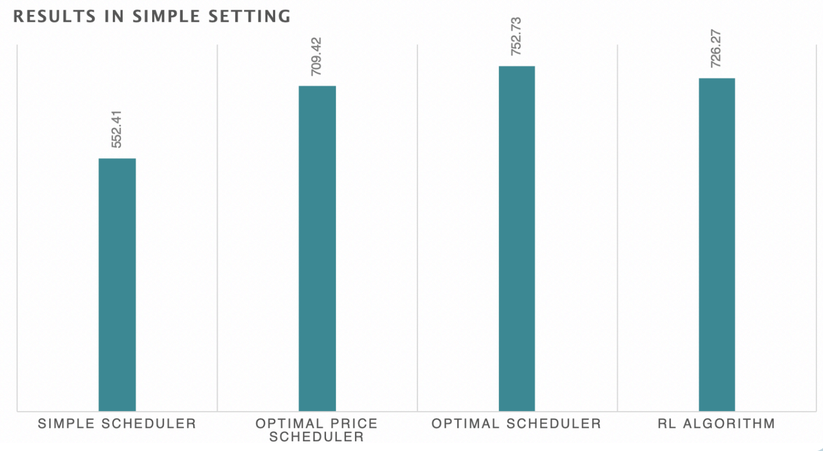
If for the task requirements all resources are busy then drop the task → else → If epsilon exploration is true or Q values are equal then choose action randomly → else → Select an action based on the current policy (action corresponding to the largest Q value)

## Policy update




## Partial Results

- Results for other benchmark algorithms are shown for comparison.
- Further tests will be conducted.




References:
- Ochal, M., Stein, S., Bi, F., Cook, M., Gerding, E., He, T. and La Porta, T. (2018). Online Mechanism Design using Reinforcement Learning for Cloud Resource Allocation.

Author contact details
Email: iam1g16@soton.ac.uk