Online Mechanism Design for Resource Allocation using Reinforcement Learning

Mateusz Ochal*, Matthew Cook*, Valerio Restocchi*, Hana Khamfroush‡, Tom La Porta‡ and Sebastian Stein*
*School of Electronics and Computer Science, University of Southampton, UK.
Email: {mo4g15,mc18g14,ss2}@ecs.soton.ac.uk
‡School of Electrical Engineering and Computer Science, Penn State University, PA, USA.
Email: {hxk5299,tlp}@cse.psu.edu

I. Introduction

We consider a setting where coalition members wish to use a pool of heterogeneous resources (e.g., a computational cloud or edge network) to complete their tasks. These tasks have varying resource requirements, different values and they arrive dynamically over time. Given this, we are interested in designing a resource allocation mechanism that decides how to dispatch these tasks to the resources, in order to maximise the total value derived. Existing work has demonstrated that reinforcement learning is a promising approach in these types of settings [1]. However, that work has neglected that task owners may behave strategically and misreport the characteristics of their tasks when this is in their interest [2]. In this paper, we use the framework of mechanism design to address this issue, and we show how to design a mechanism based on reinforcement learning that ensures several desirable properties, including dominant-strategy incentive compatibility and individual rationality [3].

II. System Model

We consider a setting with discrete time steps $T = \{1, 2, \dots, T_{\max}\}$. There is a set of tasks $I = \{1, 2, \dots\}$ that arrive dynamically over time, and each task i is characterised by an arrival time a_i, a deadline d_i, a required quantity of work q_i and a value v_i. Together, we describe the type of task i as $\theta_i = (a_i, d_i, q_i, v_i)$. We use θ^t to denote the tasks that have arrived by time t (i.e., those with $a_i \leq t$), and we use $\theta = \theta^{T_{\max}}$ to denote all tasks that arrive over the time horizon. To service tasks, there is a set of m resources: $R = \{1, 2, \dots, m\}$. Each resource r produces $p_r \in \mathbb{R}^+$ units of work per time step. Tasks can be allocated to resources — we assume that once started, the resource will work on the task until completed (i.e., tasks are non-interruptible), and a resource can work on at most one task per time step. Thus, when allocating task i to a resource r, the resource will be occupied for $\lceil q_i/p_r \rceil$ time steps. We denote the set of free resources (i.e., those that are not occupied) at time step t as R^t_{free}.

Given this, at every time step t, we need to decide which available tasks to allocate to which unoccupied resources. We denote this decision as $\pi_t(i, \theta^t) \in R^t_{\text{free}} \cup \{0\}$, where $\pi_t(i, \theta^t) = r$ means that task i will be allocated to resource r at time t and $\pi_t(i, \theta^t) = 0$ means that task i is not allocated at time t. Note the dependence on θ^t, i.e., allocation decisions have to be made based only the tasks that have arrived by time t. We assume that tasks are only allocated once, i.e., $\forall i : |\{t \in T \mid \pi_t(i, \theta^t) \neq 0\}| \leq 1$. Finally, we use π to denote the policy that determines π_t at every time step, and we use $x_i(\pi, \theta)$ to indicate whether task i was completed successfully by its deadline:

$$x_i(\pi, \theta) = \begin{cases} 1 & \text{if } \exists t \geq a_i, r > 0 : \\ & \quad \pi_t(i, \theta^t) = r \wedge t + \lceil q_i/p_r \rceil \leq d_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Our objective is to maximise the total value derived from tasks that complete within their deadline, which we denote as the *social welfare*. More formally, we define this as $W(\pi, \theta) = \sum_{i \in I} x_i(\pi, \theta) \cdot v_i$.

Clearly this is challenging, because allocation decisions have to be made online, before future arrivals are known. In practice, some probabilistic information about these may available, in which case we wish to find a policy that maximises the expected social welfare:

$$\max_{\pi} \mathbb{E}_{\theta}\{W(\pi, \theta)\} \quad (2)$$

Theorem 1. *Solving this problem is NP-hard.*

III. Reinforcement Learning Algorithm

In principle, the resource allocation problem can be formulated as an MDP and solved using standard reinforcement learning techniques. However, the state space is very large, primarily because it needs to represent the tasks that have not been allocated to resources yet, as well as the current availability of resources (i.e., when they will next become available). Furthermore, the action space is potentially very large too, as multiple tasks can be allocated to different resources at each time step.

Thus, we significantly simplify the state space by myopically considering each unallocated task in isolation and deciding which resource to allocate this task to, or to postpone it, i.e., leave it unallocated until the next time step. We assume that the ordering of tasks is decided by a prioritisation policy $\rho : 2^{\Theta} \times T \rightarrow \Theta$. This policy takes the set of available, unscheduled tasks that have not been considered at a given time step yet, and it returns the next task to consider.

Algorithm 1 shows a generic algorithm for this simplified resource allocation problem.

Algorithm 1 Generic algorithm for simplified resource allocation problem

$\theta_{\text{wait}} \leftarrow \emptyset$ ▷ Unscheduled tasks
$o_{1,\dots,m} \leftarrow 0$ ▷ Resource occupancy
$\pi_t(i) \leftarrow 0, \forall t, i$ ▷ Allocation decisions
for $t \in \{1, \dots, T\}$ **do** ▷ For each time step
  Update θ^t ▷ Receive new tasks
  $\theta_{\text{wait}} \leftarrow \theta_{\text{wait}} \cup \{\theta_i \in \theta^t \mid a_i = t\}$ ▷ Update tasks
  $\theta' \leftarrow \theta_{\text{wait}}$ ▷ Tasks to consider
  while $\theta' \neq \emptyset$ **do**
    $\theta_i \leftarrow \rho(\theta', t)$ ▷ Next to consider
    $j \leftarrow$ RL-CHOOSE$(\theta_i, t, o_{1,\dots,m})$ ▷ RL decision
    $v \leftarrow 0$ ▷ Holds reward
    if $j \neq 0$ **then** ▷ Scheduled?
      $\theta_{\text{wait}} \leftarrow \theta_{\text{wait}} \setminus \{\theta_i\}$ ▷ No longer available
      $\pi_{t+o_j}(i) \leftarrow j$ ▷ Update allocation
      $o_j \leftarrow o_j + \lceil q_i / p_j \rceil$ ▷ Update occupancy
      if $t + o_j \leq d_i$ **then** ▷ In time?
        $v \leftarrow v_i$ ▷ Receive value
      end if
    end if
    RL-REWARD(v) ▷ Update RL
    $\theta' \leftarrow \theta' \setminus \{\theta_i\}$ ▷ Remove considered task
  end while
  $\theta_{\text{wait}} \leftarrow \{\theta_i \in \theta_{\text{wait}} \mid d_i > t\}$ ▷ Remove expired
  for $j \in R$ **do**
    $o_j \leftarrow \max(0, o_j - 1)$ ▷ Update occupancy
  end for
end for

Here, different reinforcement learning algorithms could be used to implement functions RL-CHOOSE and RL-REWARD, which, respectively, pick an appropriate action to take in a given state, and signal the received reward to the reinforcement learning algorithm. Algorithm 2 shows a simple implementation of this, which uses a linear function approximator to pick the best action. This uses a weight vector $\mathbf{w}_j$ for each action j, which is used to estimate the value of taking an action in a particular state using a weighted sum of the state features.

Algorithm 2 Simple reinforcement learning algorithm

$\mathbf{w}_j$ ▷ Initial weights for $j \in \{0, 1, \dots, m\}$
$\mathbf{s}_{\text{last}}, a_{\text{last}}, v_{\text{last}}, \mathbf{s}$ ▷ Last state transition

procedure RL-CHOOSE$(\theta_i, t, o_{1,\dots,m})$
  $\mathbf{s} \leftarrow [1, d_i - t, q_i, v_i, t, o_1, \dots, o_m]$ ▷ State
  UPDATE-WEIGHTS$(\mathbf{s}_{\text{last}}, a_{\text{last}}, v_{\text{last}}, \mathbf{s})$
  $\mathbf{s}_{\text{last}} \leftarrow \mathbf{s}$
  for $j \in \{0, \dots, m\}$ **do**
    $q_j \leftarrow \mathbf{s}\mathbf{w}_j$
  end for
  return $\text{argmax}_{j \in \{0,\dots,m\}} q_j$
end procedure

procedure RL-REWARD(v)
  $v_{\text{last}} \leftarrow v$
end procedure

IV. Dealing with Strategic Agents

In practice, task owners should be seen as strategic agents that may misreport their types, if this is to their benefit.[1] To consider such cases, we assume that task types are reported as $\hat{\theta}_i = (\hat{a}_i, \hat{d}_i, \hat{q}_i, \hat{v}_i)$. We make the standard assumptions of limited misreports, i.e., $\hat{a}_i \geq a_i$, $\hat{d}_i \leq d_i$ and $\hat{q}_i \geq q_i$. We are now interested in modifying the algorithm presented in the previous section to ensure the desirable properties of *dominant strategy incentive compatibility* and *individual rationality* [3]. This can be achieved by combining the allocation rule π with a payment rule τ.

Due to space reasons, we omit the technical details and only outline the two key steps in achieving this:

- **Monotonicity of schedule actions:** We ensure that the action $j = 0$ (i.e., postpone the task) becomes less desirable as agents report a *better* type (i.e., higher value, later deadline, smaller size). This is done by adjusting the weight vector $\mathbf{w}_0$ as necessary.
- **Repeated evaluation:** We ensure that tasks are evaluated again at each time step whenever resource availability changes. This prevents strategic manipulation of the queue order.

We design a new resource allocation mechanism with these two properties called MONO-RL. In terms of payments, this charges agents their *critical values* (i.e., the minimum values they could report and still remain allocated). With this, we show:

Theorem 2. MONO-RL *is dominant strategy incentive compatible and individually rational.*

V. Conclusions

Our work has addressed the important area of resource allocation with strategic agents. In particular, we have presented the first algorithm based on reinforcement learning that adaptively learns the best scheduling actions to take and is robust to strategic manipulations.

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy-right notation hereon.

References

[1] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning." in *HotNets*, 2016, pp. 50–56.

[2] S. Stein, E. Gerding, V. Robu, and N. Jennings, "A model-based online mechanism with pre-commitment and its application to electric vehicle charging," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*, 2012, pp. 669–676.

[3] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.

[1] We assume each task owner is associated with exactly one task — thus, there is no collusion. We will consider other settings in future work.