Load-Balancing Strategies for Decentralised Queueing System

May 9, 2022

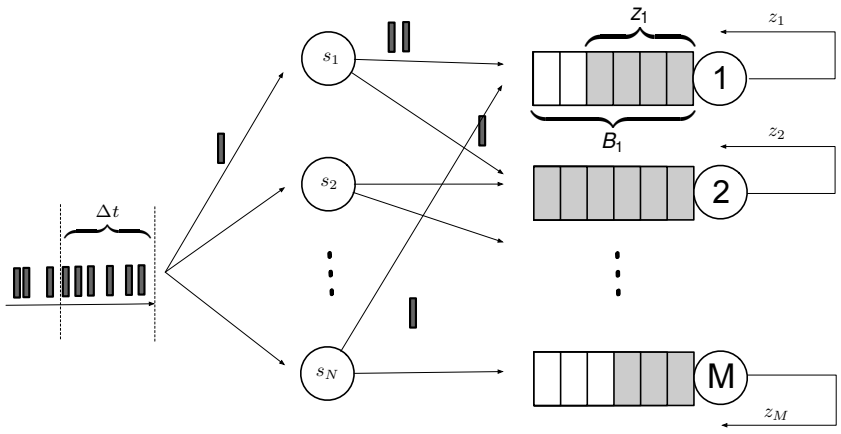

Figure 1: System Model

1 System Model

We have N schedulers and M finite queues with buffer capacity B each. Each scheduler has access to $d < M$ queues, where the value of d is fixed. $\mathcal{N} = \{1, \ldots, N\}$ is the set of agents and $\mathcal{M} = \{1, \ldots, M\}$ is the set of queues. There is an associated fixed underlying directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, and $\mathcal{E} = \mathcal{N} \times \mathcal{N}$ is the set of edges between the agents based on the queue access d that they have. An agent will have an edge $\mathcal{E}$ with another agent only if they have access to the same queue. So an agent can have maximum d edges (neighbors) to other agents. Any agent that has atleast 1 queue access common to another one will be neighbors.

Each agent, i, is associated with a local state, $z_i \in \mathcal{Z}$ and local action $a_i \in \mathcal{A}_i$. The state z_i will be the current queue filling of the d queues from $\mathcal{M}$ that the agent i has access to and the set of actions will be the set of these d

accessible queues. So, we have a finite set of action and state space. The global state of the system is given by, $z = (z_i, \ldots, z_N) \in \mathcal{Z} \coloneqq \mathcal{S}_1 \times \cdots \times \mathcal{S}_N$. Similarly, the global action is defined as $a = (a_1, \ldots, a_N) \in \mathcal{A} \coloneqq \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$.

At every epoch, t, given the current local state $z_i(t)$ and local action $a_i(t)$, the next state $z_i(t+1)$ of agent i can be calculated independently only using the neighbors' actions $a_{N_i}(t)$:

$$P(z_i(t+1)|z_i(t), a_i(t)) = \prod_{i=1}^{N} P(z_i(t+1)|a_{N_i}(t)) \tag{1}$$

where N_i is the neighborhood of agent i including itself. The state of the agent is the state of the queue it has access to and that can be affected by its own action and the action of other neighboring agents who have access to this queue.

Each agent is associated with a class of localized policies $\zeta_i^{\theta_i}$ parameterized by θ and is a distribution on the local action a_i conditioned on the states of β hop neighborhood $z_{N_i^\beta}(t)$), including agent i itself, where $\beta \geq 0$ and fixed. For us, β is the number of agents which have access to the same queues as you. For simplicity, we can assume queue allocation is fixed and in a cyclic manner, then e.g, for $d = 2$ you will always have two neighbors.

Given the state of your neighborhood $z_{N_i^\beta}(t)$, each agent takes an action $a_i(t)$ which is independently drawn from $\zeta_i^{\theta_i}(\cdot|z_{N_i^\beta}(t))$. *Note: We can not include action of other agents because they all will take action at the same time, so we won't have this information. And having the action from previous time doesn't make sense because we already have the current state.* $\theta = (\theta_i, \ldots, \theta_N)$ is the tuple of localized policies $\zeta_i^{\theta_i}$ and the joint policy is the product since each agent acts independently, $\zeta^\theta(a|z) = \prod_{i=1}^{N} \zeta_i^{\theta_i}(a_i|z_{N_i^\beta})$.

Each agent is associated with a local stage reward function $r_i(z_i, a_i)$ and a global stage reward $r(z, a) = \sum_{i=1}^{N} r_i(z_i, a_i)$. For us the reward is in terms of a penalty for job drops due to each action a_i. Objective is to find the localized cooperative policy tuple θ such that the global reward is maximized, starting from some initial state distribution μ_0:

$$\max_\theta J(\theta) \coloneqq \mathbb{E}_{z\sim\mu_0}\left[\sum_{t=0}^{\infty} \gamma^t r(z(t), a(t))|z(0) = z)\right] \tag{2}$$

$$\coloneqq \mathbb{E}_{z\sim\mu_0}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{N} r_i(z_i, a_i|z_i(0) = z_i)\right] \tag{3}$$

It is a partially observable decentralized setup since the agent does not observe the entire state but only its state and its neighbourhoods state.

2 Solutions

Possible scalable solutions in mind to compare:

- Regret matching to achieve correlated equilibrium for a cooperative setting.
- Dec-MCTS [1] for local observations
- Other concepts from game theory like any scalable POSG solutions?
- PPO with RNNs + parameter sharing?
- What else...?

References

[1] Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, and Robert Fitch. Dec-mcts: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, 38(2-3):316–337, 2019.