Communication Load Balancing via Efficient Inverse Reinforcement Learning

Abhisek Konar*, Di Wu*, Yi Tian Xu*, Seowoo Jang†, Steve Liu*, Gregory Dudek*

*Samsung Electronics, Canada, †Samsung Electronics, Korea (South)

{abhisek.k, di.wu1, seowoo.jang, steve.liu, greg.dudek}@samsung.com

Abstract—**Communication load balancing aims to balance the load between different available resources, and thus improve the quality of service for network systems. After formulating the load balancing (LB) as a Markov decision process problem, reinforcement learning (RL) has recently proven effective in addressing the LB problem. To leverage the benefits of classical RL for load balancing, however, we need an explicit reward definition. Engineering this reward function is challenging, because it involves the need for expert knowledge and there lacks a general consensus on the form of an *optimal* reward function. In this work, we tackle the communication load balancing problem from an inverse reinforcement learning (IRL) approach. To the best of our knowledge, this is the first time IRL has been successfully applied in the field of communication load balancing. Specifically, first, we infer a reward function from a set of demonstrations, and then learn a reinforcement learning load balancing policy with the inferred reward function. Compared to classical RL-based solution, the proposed solution can be more general and more suitable for real-world scenarios. Experimental evaluations implemented on different simulated traffic scenarios have shown our method to be effective and better than other baselines by a considerable margin.**

I. Introduction

The volume of wireless communication data has been snowballing in recent years. As reported in [1], the annual mobile data has increased seven times in the past six years and it is expected to reach 220.8 million terabytes per month by 2026. Alongside surge in mobile traffic, their distribution is also very uneven. As reported in [2], over 50% of the mobile traffic is being routed through a number as low as 15% of the existing cells, severely hampering the quality of experience of the users being served by these overloaded cells. Load balancing aims to balance the traffic distribution within the network, improving the quality of service (QoS) of the systems and the quality of experience (QoE) for the customers.

Load balancing algorithms can be classified into two broad categories: rule-based methods, and reinforcement learning-based methods. Rule-based methods aim to balance the load distribution using pre-determined rules but they usually lack the ability to adapt to rapidly evolving network conditons. Reinforcement learning (RL) is a class of learning algorithms, where a controller is optimized by interacting with an environment. RL has recently shown impressive performance on communication load balancing [3, 4, 5, 6, 7]. It is particularly suited for solving intricate tasks with well-defined rewards like Atari games [8]. It has also been applied for real-world tasks such as self-driving cars [9], smart grid [10, 11, 12**? ?**], traffic

control [13]. The outcome of an RL policy is contingent upon the design choices of the associated reward [14]. Reward-engineering, i.e., the design of the RL reward function, for real-world tasks can be quite challenging. Especially, for tasks like load balancing in communication networks, where improvement in the QoE of the customers is gaining significant traction. But QoE can be vague and different network performance indicators can contribute to varying degrees to form the desired reward function that can effectively capture the QoE for a demographic. Inverse reinforcement learning (IRL) [15] can act as a potential solution by circumventing the need for tedious reward designing instead inferring it from expert data.

IRL leverages a set of expert demonstrations to infer the underlying reward function that best explains the behavior of the expert. It has been used for real-world problems that lack a well-defined reward function like in robotics [16, 17] and autonomous driving [18]. Traditionally, IRL algorithms

Fig. 1: Overview of our IRL-based pipeline for load balancing.

are generally formulated under the assumption that "expert" demonstrations are optimal and, the goal of such algorithms is to recover the reward function that best explains the observed expert behavior. This assumption, however, limits the performance of the output policy to that of the expert demonstrations, which in reality are rarely actually optimal. This limitation is addressed in a recent work on reward extrapolation [19]. Reward extrapolation is able is exploit sub-optimal demonstrations to infer a reward function which is used to train policies that can outperform the demonstrator.

In this paper, we present, the first attempt of using IRL and reward extrapolation for communication load balancing. The contributions of this paper are two-fold: (1) an IRL-based learning framework (see Fig.1) to train a reward function that accurately captures the latent reward function from a set of ranked sub-optimal demonstrations. This reward function is

used in a downstream RL task to train a policy function that significantly outperforms the demonstrations; and (2) a trajectory sub-sampling technique, Temporally Consistent Sampling (TCS), that is suited for load balancing.

II. BACKGROUND

A. Terminology and notation

For consistency, we define a few terms that will be used throughout this paper as well as some mild simplifying assumptions. A communication network is composed of a number of base stations (eNB). An eNB is a physical site containing radio access devices. Each eNB in turn consists of N_S sectors. Each sector is made up of N_C cells $(c_1, c_2, ... c_{N_c})$, one for each frequency in the sector. A cell serves user equipment (UEs) of a particular carrier and is directional in nature. The sectors of an eNB are designed in a non-geographically overlapping fashion to maximize coverage. A schematic diagram of a base station is shown in Fig.2. In practice, although actual networks may violate some of these assumptions, they are needed only for explanatory purposes.

Fig. 2: The simulated communication network layout with seven eNBs along with a graphical representation of a base station within the network.

B. Load balancing mechanisms

Out of the many network load balancing mechanisms presently in practice, in this work, we focus on idle mode user equipment based load balancing and mobility load balancing. They differ by the status of the UEs that they redistribute, targeting idle and active UEs respectively.

Idle mode UE based Load Balancing: IULB deals with offloading idle UEs from an overloaded cell to its neighbors by adjusting cell re-selection priorities. Each cell i has an associated IULB weight (w_i). Once the load of a given cell breaches a specific threshold, idle UEs from that cell move to nearby cells with probability proportional to their current IULB weight. As IULB deals with idle UEs, improvements are perceived when the UEs becomes active.

Mobility Load Balancing: MLB focuses on migrating active UEs from an overloaded cell to their less-loaded counterparts through handovers. MLB gets triggered when the IP throughput of a given cell (source cell) falls below a predetermined threshold value. Then, the source cell determines the set of UEs to transfer and their corresponding target cells according to the signal quality from the UEs and the IP throughput of the

neighbouring cells. Unlike IULB, MLB has immediate effect on the network.

C. Metrics for load balancing

Many metrics, focusing on different aspects of the network, can be used to quantify the condition of a network. In this work, we use the IRL approach that, in theory, should bypass the need for reward engineering in favour of pairwise trajectory ranking provided by an expert. However, due to the lack of such infrastructure, we resort to using an engineered reward function. We focus on a set of metrics based on the throughput of the network following the practice used in a few recent works [4, 20, 21].

1) $\mathbf{T}_{min} = \min_{i \in \{1,...N_c\}} x_i$ denotes the minimum IP throughput among the cells of the sector under consideration. Here, x_i denotes the IP throughput of cell i. Higher $\mathbf{T}_{min}$ indicates better worst case cell performance.

2) $\mathbf{T}_{std} = \sqrt{\frac{1}{N_c} \sum_{i=1}^{N_c} (x_i - \frac{1}{N_c} \sum_{i=1}^{N_c} x_i)^2}$ is the standard deviation in IP throughput of the cells. Lower $\mathbf{T}_{std}$ implies fairer services across the cells.

3) $\mathbf{T}_{cc} = \sum_{i=1}^{N_c} \mathbb{1}(x_i < \mathbb{x})$ counts the number of cells in the sector that have a throughput lower than a given threshold value $\mathbb{x}$. Here $\mathbb{x}$ is chosen to be a small constant. Lower $\mathbf{T}_{cc}$ suggests less congested cells.

In addition to gauging performance, the above KPIs individually or as a combination can also serve other roles like creation of a reward function in an RL setting [21].

D. Load balancing algorithms

Classic load balancing algorithms are rule-based and use expert domain knowledge to make informed decisions. [22] performs intra-frequency load balancing in Long Term Evolution (LTE) networks by automatically adjusting the cell-specific offset based on the current cell loads. [23] focuses on the use of adaptive step-size to adjust the offset between neighboring cells. [24] exploits the geometric properties of the network infrastructure and uses Voronoi tessellations over a geographical area to compute UE association with nearby edge servers. Data-driven approaches have recently enjoyed considerable success. Various learning-based methods, including supervised and reinforcement learning, have been successfully applied to the problem of load balancing. [25] use spatio-temporal network data to predict future traffic patterns and subsequently use this information to perform proactive user association. [26]and [27] use Q-learning to address the load balancing in self-organizing networks, and heterogeneous networks respectively. [3] proposes a multi-agent actor-critic network model in a model-free off-policy setting to obtain an optimal policy for MLB. [28] uses RNN to understand past SINR measurements as a function of UE trajectory and number of HOs. [4] propose a data-efficient transfer learning-based RL approach that is robust to environmental fluctuations.

A thorough scrutiny, however, will reveal a lack in consensus for the choice of reward function. [26] considers the change in load while optimizing for MLB. [3] use the inverse

of the maximum load of a cell in a given neighborhood and, [27] relies on the change in overload among neighboring cells. The choice of the reward function, in RL, defines the task [29] and plays a pivotal role in determining the final performance of the controller policy. There are different opinions in the community on what an *ideal* reward function should be.

E. Inverse RL and reward extrapolation

Inverse RL aims to recover a reward function from expert demonstrations that best explains the expert's behavior [15]. While the use of IRL is yet to receive traction in network load balancing community, there has been recent work that use it in the field of cellular networks. [30] use IRL to optimally allocate power in multi-user cellular networks. Other real-world applications where IRL has seen significant success are autonomous driving and dexterous robotic manipulation. In autonomous driving, expert demonstrations are used to train a policy that drives [18] and parks [31] like a human driver. For robotic manipulation, [16] uses IRL to train a robot to perform dexterous tasks. [32] trains a model to learn grasping from failed demonstrations. Recent works like TRajectory EXtrapolation (T-REX) [19] aim towards reward extrapolation. The goal being able to leverage the goodness of expert data to come up with a reward function that can infer rewards from unseen states. It is a relatively new topic, with applications limited to environments like Atari and Mujoco [33].

III. PROPOSED METHOD

Our proposed method, TRajectory EXtrapolation(T-REX) [19] using Temporally Consistent Sampling (TCS) follows a similar training pipeline of T-REX along with the inclusion of a data-augmentation module, that uses TCS, to generate training data for better extrapolation results. We start by defining the problem in Section III-A, followed by the individual components of the IRL-based learning framework in Sections III-B and III-C.

A. Problem formulation

The problem is modelled as an Markov decision process (MDP) represented by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ and are defined as follows:

1) $\mathcal{S}$ is the set of states. Each state, $s \in \mathcal{R}^{3 \times N_C}$, consists of 3 components: the number of active UEs $s_{ue} \in \mathcal{R}$, the average throughput per cell $s_{ip} \in \mathcal{R}$, and the average percentage of the physical resources of each cell used $s_{prb} \in \mathcal{R}$ of every cells of a given sector. The number of cells in our system, N_C, is 4. Hence $s \in \mathcal{R}^{12}$.
2) $\mathcal{A}$ is set of actions. Each action, a consists of two parts, one to initiate IULB ($a_{IULB} \in \mathcal{R}^{N_C}$) and one that controls handover thresholds to trigger MLB ($a_{MLB} \in \mathcal{R}^{3 \times N_C}$). All the actions are discrete with a step size of 1.
3) $\mathcal{P}(s, a, s') = P(s' = s_{t+1} | s = s_t, a = a_t)$ are the state transition probabilities.
4) $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is the reward function.

For IRL, $\mathcal{R}$ is unavailable and the objective is to learn the reward function from expert observations. A policy π_ϕ : $\pi_\phi(a|s) \in [0, 1]$, parameterized by ϕ, is a function that maps a given state, $s \in \mathcal{S}$, to a distribution over actions, $a \in \mathcal{A}$. A trajectory τ of length n is represented by a sequence of states $\{s_1, s_2, ... s_n\}$.

Given a set of m ranked trajectories, $\{\tau_1, \tau_2, ... \tau_m\}$ where τ_j is better than τ_i (i.e., $\tau_i \prec \tau_j$ if $i < j$, the objective is to find a parameterized reward network $\hat{r}_\theta$ that is able to capture the relative ranking of the demonstrated trajectories. In the process, it has to extrapolate the underlying reward function the demonstrations are trying to maximize. Once, $\hat{r}_\theta$ is obtained, it is used to train a policy that has the potential to outperform the demonstrations.

B. Reward extrapolation using T-REX

The goal is to train a reward network, $\hat{r}_\theta$, that maintains $\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)$ when $\tau_i \prec \tau_j$, where $\hat{J}_\theta(\tau_i) = \sum_{s \in \tau_i} \hat{r}_\theta(s)$ denote the total reward obtained by trajectory τ_i using $\hat{r}_\theta$. For such a model, the general loss function can be given by Equation 1.

$$\mathcal{L}(\theta) = \mathbf{E}_{\tau_i, \tau_j \in \Pi} \left[\xi(P(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j)), \tau_i \prec \tau_j \right] \quad (1)$$

where Π is the set of ranked demonstrations, and ξ is a binary classification loss. Following the classic models of preference [34] (Equation 2), and using a cross-entropy loss for ξ, $\mathcal{L}$ can be rewritten as Equation 3.

$$P\left(\hat{J}_\theta(\tau_i) < \hat{J}_\theta(\tau_j) \right) \approx \frac{\exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)} \quad (2)$$

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \log \left(\frac{\exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)} \right) \quad (3)$$

Once a reward network $\hat{r}_\theta$ is trained, it is used to train a policy using Proximal Policy Optimization (PPO) [35]. Additional details about the training parameters for both reward and policy learning are provided in Section IV-C.

C. Temporally Consistent Sampling

The collection of expert demonstrations is expensive, and reward extrapolation relies on data augmentation making the sub-sampling technique employed for the data augmentation a vital component of the training pipeline. [19] proposes the sampling of contiguous blocks of states of equal length from randomly selected starting points in trajectories. The method assumes that the relative ranking of these sub-trajectories match that of the complete trajectories they were sampled from. While this assumption is reasonable in certain domains such as Atari and Mujoco, we argue that it can be detrimental in the load balancing domain when the performance is mainly determined by the throughput-based metrics as presented in Section II-C. It is commonly observed that the network

Fig. 3: Plot of all trajectories from the demonstration set from a load balancing scenario. The trajectories exhibit a periodic behavior over the minimum IP throughput. Similar trends are observed across different network KPIs.

Fig. 4: Scatter plot of the minimum IP throughput at different timesteps of two trajectories. While the value across different timesteps varies widely, for a given time, minimum Ip throughput of the state from the better trajectory is consistently (9/10) higher than its worse counterpart. Only 10 such states are shown for visual clarity.

throughput varies significantly through time due to regular daily high and low usage periods. For example, Fig. 3 shows the variation of the minimum IP throughput among the cells (T_{min}) in the span of one week. Clearly, sampling pairs of sub-trajectories from different time intervals would often result in an inconsistency between the assumption from [19] and the performance metric. Indeed, we find such inconsistency in 35% of our samples in our experiment. Its adverse effect on the learning capabilities of the reward network is discussed in detail in Section IV-D.

To mitigate the problem, we propose a new sampling procedure: Temporally Consistent Sampling(TCS). TCS introduces two changes to the previously proposed sub-sampling technique. Firstly, to account the temporal variation in the network throughput, we opt for sampling over random time indexes rather than contiguous blocks. Secondly, as the variation in the network KPIs across different trajectories at a given time index is relatively lower and fairly consistent with the original relative ranking (Figure 4), we mandate temporal consistency, i.e., given a pair of ranked trajectories, a sub-trajectory should be sampled from the same time indexes for both the trajectories maintaining temporal consistency. Empirically, TCS can reduce the aforementioned inconsistency from 35% to 15%, significantly increasing the effectiveness of the training samples. Our approach is summarized in Algorithm 1.

IV. EXPERIMENTS AND RESULTS

A. Simulation environment

The experiments are conducted on a system-level RAN simulator[36, 37] consisting of seven eNBs. Each base station consists of three sectors, which in turn comprises of four cells with different carrier frequencies. Six eNBs are arranged in a uniform hexagonal ring with one eNB at the center as shown in Figure 2. The UEs in the system are randomly distributed over the geographical area and are assumed to be points with fixed velocities and random directions drawn from a uniform distribution. We test the extrapolation performance of the reward network in two traffic scenarios and the load balancing performance of the policy network in four scenarios

---

Algorithm 1: Temporally consistent sampling

Input : Ranked demonstrations: D_E, Number of samples to generate: t_{sub}

begin

  for $i = 1$ *to* t_{sub} **do**

    Randomly select a pair of trajectories, t_x and t_y, from D_E

    Define the length of a sample sub-trajectory, l.

    Randomly select a set of l indexes, n_l.

    $t_x^{samp} \leftarrow t_x[n_l]$, $t_y^{samp} \leftarrow t_y[n_l]$

    if *rank(t_x) > rank(t_y)* **then**

      $t_{label} \leftarrow 0$

    else

      $t_{label} \leftarrow 1$

    Append $\{(t_x^{samp}, t_y^{samp}), t_{label}\}$ to T_s.

  Return T_s

Output: Training set, T_s

(ID 1-4). These scenarios are determined by different number of UEs, request packet size and interval distributions.

B. Collecting demonstrations

For each scenario, we collect a set of 100 trajectories using a random controller. A random controller randomly samples an action at each hour. In the absence of an external critic to rank the demonstrations, we resort to an ad-hoc ranking function, R_f, based on a weighted combination of a set of KPIs. We use the reward function from [21] as our ranking function. We would like to emphasise the fact that the use of ranked trajectories provides a significantly low resolution image of the reward landscape than using a reward function to learn a controller in an RL setting which needs access to the reward corresponding to each state it encounters. For a given pair of expert demonstrations τ_i and τ_j, $\tau_i \prec \tau_j$ when $\sum_{s \in \tau_i} R_f(s) < \sum_{s \in \tau_j} R_f(s)$. From the set of 100 trajectories, we select the worst 70% to build our training set and leave the rest to test the extrapolation capabilities of the trained reward network.

C. Training details

The reward network consists of 3 fully-connected layers with a leaky ReLU activation function [38] for the input and the first hidden layer. Other hyperparameters of the training the reward and the policy networks are listed in Table Ia. The training time for the reward network took on average roughly 45 hours on an NVIDIA A6000. The policy network is trained using PPO. Both the actor and value network consist of 3 fully-connected layers with 256 neurons in each layer and use the tanh activation function. All the networks are optimized using Adam [39]. Hyperparameters used for training the policy network are listed in Table Ib.

Learning rate	$1e-5$		Learning rate	0.0003
Weight decay	$1e-4$		Weight decay	$1e-4$
No. of epochs	1200		Total timesteps	$200k$
No. trajectory pairs	1000		Gamma	0.97
No. sub-trajectory pairs	50000		Clip range	0.15
			Batch size	64

(a) Reward learning.	(b) Policy learning.

TABLE I: Training hyperparameters for different parts of the learning pipeline.

ID	Method	Training	Extrapolation
1	T-REX (Original)	**0.98**	0.83
	T-REX + TCS(ours)	0.94	**0.94**
4	T-REX (Original)	**0.99**	0.53
	T-REX + TCS(ours)	0.98	**0.93**

TABLE II: Pearson correlation on the training and extrapolation set for different sub-trajectory sampling techniques. Maintaining temporal consistency between candidate sub-trajectories during sampling from a given pair of trajectories consistently outperforms its counterpart across different scenarios. For reference: a higher value is better.

D. Results: Reward extrapolation

To test the performance of the algorithm, we use the Pearson correlation coefficient [40]. It calculates the linear relationship between two datasets and outputs a value in the range of $[-1, 1]$ which is proportional to their correlation. Table II shows the correlation values obtained between the ranking reward function and the reward predicted by the trained reward network. From Table II, we see that while the original method (T-REX) outperforms in the training set, using TCS shows consistent improvement in the extrapolation or test set across different scenarios. This indicates that the model trained from samples generated from the original sampling technique learns a reward function that lacks generalization. A possible explanation for this stems from the empirical observations that the probability of mislabeling a sub-sample pair using TCS drops from 0.35 to 0.13, helping the model better capture the latent ranking reward and thus contributing to better extrapolation. A qualitative overview of the extrapolation performance is shown in Figure 5.

E. Results: Model performance

We compare the load balancing performance of the trained policy with four baselines, i.e., demonstrations, fixed rule-based method, adaptive rule-based method and the original

(a) Without TCS.	(b) With TCS.

Fig. 5: Scatter plots qualitatively showing the correlation between the predicted and ad-hoc reward for the demonstration trajectories in the training and test (extrapolation) set.

TREX method. Demonstrations are the set of trajectories generated from the random controller on which the reward network was trained. Fixed rule-based method is a rule-based method consisting of a set of fixed IULB and MLB parameters as used in [4]. Adaptive Rule-based method is a rule-based load balancing algorithm [41] that dynamically adjusts the IULB and MLB parameters according to the difference in the loads between neighboring cells. We do not include RL based methods in the comparison because these methods usually require access to rewards from individual states as compared to pairwise trajectory ranks utilized by our method. Pairwise ranking on a trajectory level is far more accessible as compared to the former, especially in the real world, and in this work we focus on methods that can be trained and deployed in such conditions.

We evaluate the performance of the methods on a single sector of the central base station on the three metrics introduced in Section II-C. Table III reports these metrics averaged over the entire sequence of a given scenario at each hour.

ID	Method	$\mathbf{T}_{min}$	$\mathbf{T}_{std}$	$\mathbf{T}_{cc}$
1	Ours	**2.69±0.01**	**1.46±0.02**	**0.00±0.00**
	Fixed rule	1.60 ± 0.01	2.55 ± 0.02	0.09 ± 0.01
	Demonstrations	1.89 ± 0.19	2.29 ± 0.29	0.05 ± 0.07
	Adaptive rule	2.15 ± 0.05	2.05 ± 0.05	0.03 ± 0.01
	TREX(Original)	2.51 ± 0.02	1.66 ± 0.02	**0.00±0.00**
2	Ours	**1.97±0.02**	**1.07±0.01**	0.02 ± 0.01
	Fixed rule	1.50 ± 0.01	1.51 ± 0.02	0.03 ± 0.01
	Demonstrations	1.45 ± 0.13	1.74 ± 0.28	0.25 ± 0.19
	Adaptive rule	1.44 ± 0.07	1.89 ± 0.10	0.32 ± 0.07
	TREX(Original)	1.68 ± 0.02	1.34 ± 0.01	**0.02±0.01**
3	Ours	**1.63±0.02**	**1.63±0.04**	**0.20±0.02**
	Fixed rule	1.47 ± 0.02	2.31 ± 0.04	0.31 ± 0.02
	Demonstrations	1.54 ± 0.08	2.07 ± 0.15	0.30 ± 0.10
	Adaptive rule	1.57 ± 0.04	2.06 ± 0.06	0.28 ± 0.03
	TREX(Original)	1.62 ± 0.02	1.74 ± 0.04	0.21 ± 0.02
4	Ours	**2.77±0.01**	**1.73±0.02**	0.01 ± 0.00
	Fixed rule	2.05 ± 0.01	2.58 ± 0.02	**0.00±0.00**
	Demonstrations	2.09 ± 0.15	2.61 ± 0.29	0.02 ± 0.03
	Adaptive rule	2.27 ± 0.07	2.41 ± 0.12	0.02 ± 0.01
	TREX(Original)	2.38 ± 0.01	2.14 ± 0.02	0.01 ± 0.00

TABLE III: Evaluation results on different traffic scenarios. For T_{min}, a higher value translates to a better performance and for T_{std} and T_{cc}, lower is better. Our method enjoys an an average improvement of 19.6%, 26.7% and 32.3%in T_{min}, T_{std} and T_{cc} respectively over the next best method.

Table III, shows that our proposed method consistently

outperforms the other methods across different network scenarios. For T_{min}, we achieve on average an improvement of 10.3% over the second best performing method across different scenarios. Figure 6a shows that our method is able to obtain significantly higher T_{min} values at times of moderately high traffic. At times when the network traffic is low, the performance of all the methods is comparable due to the lack of any room for further optimization to wring out more from the existing network infrastructure. Our method also attains a substantial reduction in T_{std}. It outperforms its nearest competitor by approximately 14.4% averaged across all the scenarios, and it shows consistent improvement across both periods of high and low traffic as seen from Figure 6b.

(a) Minimum IP throughput obtained by different methods over the period of one week. From the trends in the minimum IP throughput, the timeline of each week can be divided into low and high traffic intervals. The line plots show that our proposed method better capitalizes on the network resources during moderately low network traffic hours when compared to the other baselines improving the minimum IP throughput in the network.

(b) Standard deviation in IP throughput over time. The plot shows that our method consistently maintains a lower standard deviation across the entire week.

Fig. 6: Performance of competing methods in different network KPIs.

V. CONCLUSION AND FUTURE WORK

With the rapid increase and uneven distribution of communication traffic, communication load balancing has become a pressing problem in maintaining the quality of experience for customers. In this work, we showcase the first attempt to use inverse reinforcement learning for communication load balancing, bypassing the need for an explicitly defined reward signal. We can learn a reward function from a collection of system demonstrations and then utilize that to train a reinforcement learning-based load balancing control policy. Experimental results on different traffic scenarios have showcased the the proposed solution can help significantly improve the system performance. We believe that this work has showcased the effectiveness of inverse reinforcement learning and provides a new direction for future load balancing research. In the future, we plan to further improve the data efficiency of the proposed solution and investigate the applicability of the proposed solution to other networking optimization problems, e.g., energy saving, network slicing.

REFERENCES

[1] "Yahoo News global traffic," https://yhoo.it/3vT1DEC, accessed: 2022-04-20.
[2] H. Holma and A. Toskala, *LTE-Advanced: 3GPP Solution for IMT-Advanced*. John Wiley & Sons, Ltd, 2012.
[3] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Deep reinforcement learning based mobility load balancing under multiple behavior policies," in *2019 IEEE International Conference on Communications*, 2019, pp. 1–6.
[4] D. Wu, J. Kang, Y. T. Xu, H. Li, J. Li, X. Chen, D. Rivkin, M. Jenkin, T. Lee, I. Park, X. Liu, and G. Dudek, "Load balancing for communication networks via data-efficient deep reinforcement learning," in *2021 IEEE Global Communications Conference*, 2021, pp. 01–07.
[5] A. Feriani, D. Wu, Y. T. Xu, J. Li, S. Jang, E. Hossain, X. Liu, and G. Dudek, "Multiobjective load balancing for multiband downlink cellular networks: A meta-reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2614–2629, 2022.
[6] M. Ma, D. Wu, Y. T. Xu, J. Li, S. Jang, X. Liu, and G. Dudek, "Coordinated load balancing in mobile edge computing network: a multi-agent drl approach," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 619–624.
[7] J. Kang, X. Chen, D. Wu, Y. T. Xu, X. Liu, G. Dudek, T. Lee, and I. Park, "Hierarchical policy learning for hybrid communication load balancing," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.
[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[9] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.
[10] Y. Fu, D. Wu, and B. Boulet, "Reinforcement learning based dynamic model combination for time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6639–6647.
[11] H. Zhang, D. Wu, and B. Boulet, "Metaems: A meta reinforcement learning-based control framework for building energy management system," *arXiv preprint arXiv:2210.12590*, 2022.
[12] D. Wu, *Machine learning algorithms and applications for sustainable smart grid*. McGill University (Canada), 2018.
[13] X. Huang, D. Wu, M. Jenkin, and B. Boulet, "Modellight: Model-based meta-reinforcement learning for traffic signal control," *arXiv preprint arXiv:2111.08067*, 2021.
[14] Y. Fu, D. Wu, and B. Boulet, "A closer look at offline rl agents," in *Advances in Neural Information Processing Systems*.
[15] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 1.
[16] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*. PMLR, 2016, pp. 49–58.
[17] X. Zhang, L. Sun, Z. Kuang, and M. Tomizuka, "Learning variable impedance control via inverse reinforcement learning for force-related tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2225–2232, 2021.
[18] Q. Zou, H. Li, and R. Zhang, "Inverse reinforcement learning via neural network in driver behavior modeling," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1245–1250.
[19] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *International conference on machine learning*. PMLR, 2019, pp. 783–792.
[20] J. Kang, X. Chen, D. Wu, Y. T. Xu, X. Liu, G. Dudek, T. Lee, and I. Park, "Hierarchical policy learning for hybrid communication load balancing," in *IEEE International Conference on Communications*, 2021, pp. 1–6.
[21] J. Li, D. Wu, L. T. Xu, Yi Tian, S. Jang, X. Liu, and G. Dudek, "Traffic scenario clustering and load balancing with distilled reinforcement learning policies," in *2022 IEEE International Conference on Communications*, 2022.
[22] R. Kwan, R. Arnott, R. Paterson, R. Trivisonno, and M. Kubota, "On mobility load balancing for lte systems," in *2010 IEEE 72nd Vehicular Technology Conference - Fall*, 2010, pp. 1–5.

[23] Y. Yang, P. Li, X. Chen, and W. Wang, "A high-efficient algorithm of mobile load balancing in lte system," in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, 2012, pp. 1–5.

[24] V. Sohrabi, M. E. Esmaeili, M. Dolati, A. Khonsari, and A. Dadlanit, "Evblb: Efficient voronoi tessellation-based load balancing in edge computing networks," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[25] Y. Zhang, K. Sun, X. Gao, W. Huang, and H. Zhang, "Load balancing and user association based on historical data," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[26] S. S. Mwanje, L. C. Schmelz, and A. Mitschele-Thiel, "Cognitive cellular networks: A q-learning framework for self-organizing networks," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 85–98, 2016.

[27] T. Kudo and T. Ohtsuki, "Q-learning based cell selection for ue outage reduction in heterogeneous networks," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, 2014, pp. 1–5.

[28] M. Gupta, R. M. Dreifuerst, A. Yazdan, P.-H. Huang, S. Kasturia, and J. G. Andrews, "Load balancing and handover optimization in multi-band networks using deep reinforcement learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[29] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 663–670.

[30] R. Zhang, K. Xiong, X. Tian, Y. Lu, P. Fan, and K. B. Letaief, "Inverse reinforcement learning meets power allocation in multi-user cellular networks," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–2.

[31] P. Fang, Z. Yu, L. Xiong, Z. Fu, Z. Li, and D. Zeng, "A maximum entropy inverse reinforcement learning algorithm for automatic parking," in *2021 5th CAA International Conference on Vehicular Control and Intelligence (CVCI)*. IEEE, 2021, pp. 1–6.

[32] X. Xie, C. Li, C. Zhang, Y. Zhu, and S.-C. Zhu, "Learning virtual grasp with failed demonstrations via bayesian inverse reinforcement learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1812–1817.

[33] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[34] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[35] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] J. Kang, X. Chen, D. Wu, Y. T. Xu, X. Liu, G. Dudek, T. Lee, and I. Park, "Hierarchical policy learning for hybrid communication load balancing," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[37] A. Feriani, D. Wu, Y. T. Xu, J. Li, S. Jang, E. Hossain, X. Liu, and G. Dudek, "Multiobjective load balancing for multiband downlink cellular networks: A meta- reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2614–2629, 2022.

[38] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*. Madison, WI, USA: Omnipress, Jun. 2010, pp. 807–814.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] Wikipedia contributors, "Pearson correlation coefficient," 03 2022. [Online]. Available: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[41] Y. Yang, P. Li, X. Chen, and W. Wang, "A high-efficient algorithm of mobile load balancing in lte system," in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, 2012, pp. 1–5.