Meta Multi-Objective Reinforcement Learning for Communication Load Balancing

by

Amal Feriani

A Thesis submitted to The Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

December 2021

Abstract

With the rapid adoption of fifth-generation (5G) communication systems and the increasing data demand per device, communication traffic is soaring to reach unprecedented numbers in the upcoming years. Moreover, the traffic is often unevenly distributed across frequency bands and base stations, thereby resulting in a degradation of the network throughput and user experience. Thus, load balancing has become a key technique to adjust the traffic load by offloading users from overloaded cells to less crowded neighboring ones.

In this thesis, we study multi-objective reinforcement learning (MORL) and meta reinforcement learning (meta-RL) for load balancing to learn highly customized policies for different trade-offs between network performance metrics. We begin with a thorough review of existing load balancing literature to motivate the need for better algorithms that can further improve the network performance and the user's quality of service (QoS). Specifically, we emphasize the importance of a multi-objective approach to solve the load balancing problem since network providers aim to simultaneously optimize multiple conflicting objectives by adjusting the load balancing parameters. Using MORL, we formulate communication load balancing as a multi-objective control problem where the agent seeks to find optimal policies depending on possible trade-offs between the network performance indicators. Motivated by the dynamic nature of wireless networks, we propose a practical algorithm based on meta-RL concepts to compute a general load balancing policy capable of rapidly adjusting to new trade-offs. Indeed, the learned meta-policy can be fine-tuned for given preferences using fewer samples and gradient steps. We further enhance the generalization and adaptation of our proposed meta-RL solution using policy distillation techniques. To showcase the effectiveness of our framework, experiments are conducted based on real-world traffic scenarios. Our results show that our load balancing framework can: i) significantly outperform the existing rule-based load balancing methods ii) achieve better performance than single-objective solutions iii) compute better Pareto front approximations compared to several MORL baselines and iv) quickly adapt to new unseen objectives.

To conclude, we analyze the limitations of the proposed solutions and we discuss several future promising directions for multi-objective load balancing.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Ekram Hossain. His constant support and advice are what made this thesis possible.

I am thankful to Montreal Samsung AI Center (MSAIC) for providing me with a great opportunity to work at the intersection of wireless communication and artificial intelligence. I would like to thank the leaders of MSAIC, Professors Greg Dudek and Steve Liu, who gave me the possibility to work with their extremely talented team. It was a great pleasure to work with Dr. Di Wu, who has shaped many of the core ideas of this work and closely collaborated with me to design a meta-MORL solution for load balancing.

My special thanks go to my thesis committee members: Professor Ahmed Ashraf and Professor Amine Mezghani for their great feedback which contributed to the improvement of this work.

Finally, I dedicate this work to my family for their endless support and encouragement.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CMLB	Connected Mode Load Balancing
eNB	evolved NodeBs
GAE	Generalized Advantage Estimation
HO	Handover
IMLB	Idle Model Load Balancing
KPI	Key Performance Indicator
LB	Load Balancing
LTE	Long Term Evolution
MAML	Model Agnostic Meta Learning
MDP	Markov Decision Process
MOMDP	Multi-Objective Decision Process
MOO	Multi-Objective Optimization
MORL	Multi-objective Reinforcement Learning
OFDM	Orthogonal Frequency Division Multiplexing
PD	Policy Distillation
PG	Policy Gradients
PPO	Proximal Policy Optimization
PRB	Physical Resource Block
QoS	Quality of Service

SINR	Signal to Interference and Noise Ratio
SON	Self Organizing Networks
RSRP	Reference Signal Receive Power
RSRQ	Reference Signal Receive Quality
RL	Reinforcement Learning
UE	User Equipement
TRPO	Trust Region Policy Optimization
TTI	Transmission Time Intervals

Chapter 1

Introduction

1.1 Overview and Motivation

The future wireless generations are expected to accommodate a significant increase in the number of connected devices including mobile devices such as smartphones, tablets, and laptops and other types of devices such as sensors and smart home devices. An upsurge of high data rate applications has also been recorded: mobile data traffic is expected to increase by a factor of five to reach 237 exabytes per month by 2026, and 77% of this traffic is related to video data usage [4]. Furthermore, the traffic distribution is subject to spatio-temporal fluctuations due to heterogeneity and the high mobility of the connected devices [4, 5]. Indeed, more crowded areas, like office buildings, demand a higher amount of resources during the daytime.

To ensure a good service, network operators are looking for creative solutions to address these difficulties. For instance, network densification has recently been adopted to boost the system capacity and connectivity and address the spatial disparities in traffic distribution [6]. Either by deploying more evolved NodeBs (eNBs) or placing smaller base stations, the signal quality can be improved in high-traffic areas. However, without careful user association schemes, the load distribution will become unbalanced. Faced with these challenges, network operators risk not meeting users' QoS requirements and the expected network efficiency. To provide reliable and low-latency communications, there is a pressing need for more efficient management of network resources.

To address the mentioned challenges, self-organizing networks (SON), in Long Term Evolution (LTE) systems, enable the automation of several operational tasks related to the configuration, optimization, and healing of the network [7]. In this context, load balancing is considered as one of the key technologies enabling a better regulation of the network traffic load. Load balancing mechanisms adjust the load distribution by offloading user equipment (UE) units from crowded eNBs or frequency bands to less loaded ones (see Fig 1.1). Consequently, the network's overall performance can be improved through more efficient resource utilization and better coverage for edge users. Traditionally, the load balancing parameters are manually set by the operator, which is a time-consuming and costly task. Mobility robustness optimization and mobility load balancing enable the automatic adjustment of the mobility parameters to account for the dynamic channel conditions and different mobility patterns. In this thesis, we will focus on mobility load balancing for two types of UEs: *load-based handover* (HO) for connected UEs and *cell re-selection* for idle UEs. Incorrect adjustment of the HO and re-selection parameters (i.e event thresholds and/or offsets) engender a degradation in the user experience and network performance due to offloading failures (early or late triggering) and ping-pong effects. As an example, incorrect HO parameters may lead to a prolonged connection to a non-optimal eNB. Furthermore, the cell re-selection and HO parameters should be aligned to avoid potential congestion when idle UEs become connected. A more detailed explanation of the different HO and cell re-selection parameters is provided in Section 3.2.

In this work, we will focus on two categories of load balancing algorithms: rule-based and reinforcement learning (RL) based methods. Rule-based algorithms present a predefined set of rules for adjusting the load balancing parameters. These rules rely on adding (or

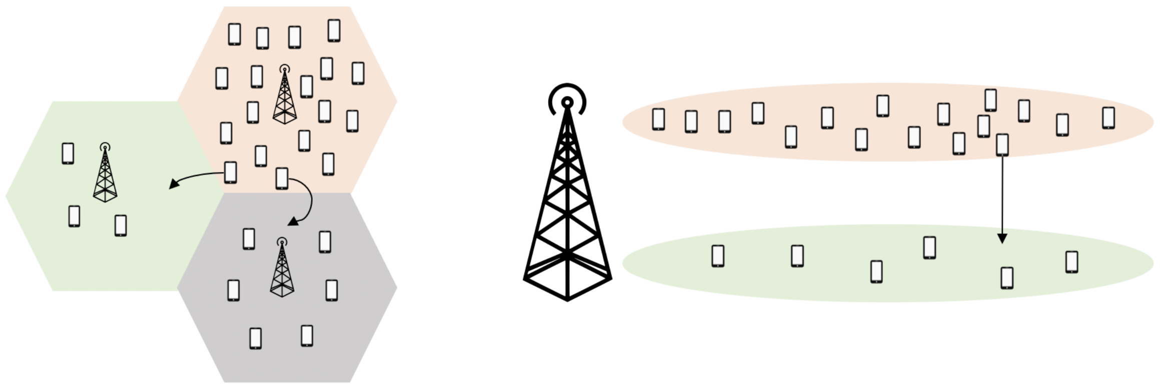


Figure 1.1: Illustration of load balancing between eNBs (left) and frequency bands (right)

decreasing) a fixed [8, 9] or an adaptive [10–13] step size to the mobility parameters based on traffic load measurements and signal conditions. These reactive approaches have proved that load balancing results in a more balanced traffic distribution and a better network performance. However, they are tailored to specific network settings and require expert knowledge to define the adjustment rules.

Recently, RL methods have shown great success in solving complex and high-dimensional control problems. In this context, RL-based solutions are proposed where an agent learns a load balancing strategy based on interactions with its network environment. Using RL techniques for load balancing is not new since several tabular Q-learning algorithms are designed for HO management (e.g., [14, 15]). Recently, deep RL techniques are adopted to learn load balancing policies in different network architectures such as ultra-dense 5G networks [16], device-to-device networks [17, 18], mobile millimeter-wave networks [19]. In this thesis, we focus on the second family of load balancing methods which is RL-based methods.

The aforementioned RL-based load balancing solutions are *single-objective* solutions since only one metric (e.g., throughput) or a fixed combination of network key performance indicators (KPIs) is optimized. The drawbacks of such approaches are: i) in real-world situations, these KPIs are often *conflicting* such that one fixed preference cannot cover all the possible trade-offs ii) the desired KPI weighting may vary across eNBs on which the solution will be deployed and iii) the objective preferences for the deployment network may not be known before training. This motivates the design of a multi-objective load balancing method. We propose to use MORL to learn a set of control policies depending on the selected objective preferences. Different algorithms exist to solve MORL problems (see Section 1.2 for a literature review), meta-RL [20–22] has been recently adopted to learn a general policy that can quickly adapt to new trade-offs between objectives [23]. To the best of our knowledge, this is the first attempt to use a meta multi-objective RL method for communication load balancing. We aim to learn a generalized load balancing policy from multiple objective preferences which can be used as model initialization to quickly adapt to changes in the KPI importance weights. Fast adaption to new preferences is a key benefit of the proposed method, since learning a new reliable control policy from scratch typically requires a large number of interactions [24].

1.2 Related Work

1.2.1 Load Balancing for Wireless Communication

This section presents an overview of previous work on load balancing by distinguishing two main categories: rule-based and RL-based methods.

Rule-Based methods

As mentioned above, rule-based algorithms hand-design a set of update rules for the HO/cell re-selection parameters. For HO management, most of the previous work considers the adjustment of the cell individual offset based on the load measurements and/or the signal strength. For instance, the authors in [8] propose to adjust the cell individual offset by adding or subtracting a fixed step size. Another work [9] presents a rule-based algorithm for downlink LTE networks taking into consideration non-adjacent neighboring cells. The drawback of these methods is they do not adapt to changes in the system due to their fixed step size. This is why other studies introduce adaptive rule-based algorithms. In these works [10, 11], the step size is dynamically adapted considering the load difference between cells and/or the signal strength condition. Another line of work [12] proposes an adaptive approach to determine the threshold for overloaded cells instead of using a fixed one. All the aforementioned works are intended for UEs in radio resource control (RRC) connected state where load balancing is mainly performed through HO. Load balancing for RRC idle UEs has attracted less attention. Cell re-selection is studied in [13] where an adaptive algorithm is proposed to adjust the re-selection parameters without harming the HO performance.

RL-based methods

Due to the increasing heterogeneity and high mobility of connected devices, RL-based methods gained more attention since designing hand-tailored rules cannot cover all possible circumstances encountered in real-world scenarios. Specifically, (deep) RL has been largely adopted to dynamically adjust the load balancing parameters. Through interactions with the environment, the RL agent can incorporate knowledge about the network in its decision process. Indeed, earlier work by Muñoz *et al.* [14, 25] proposes a fuzzy Q-learning approach that self-tunes the parameters of fuzzy logic controllers [26]. Dynamic programming [27] is applied to learn a policy that chooses the optimal step size for the adjustment of different HO parameters under different load conditions [15, 28, 29]. More recently, Attiah *et al.* apply deep Q-learning to automatically adjust the cell offsets and maximize the throughput and/or resource block utilization [30]. Similarly, the joint optimization of the HO offsets and the base station transmission power using a value-based RL method has also been proposed [31]. In a similar vein, several endeavors applied deep RL for load balancing in different propagation environments and network architectures [16–19, 32].

1.2.2 Multi-Objective Reinforcement Learning

MORL is an extension of the classic RL framework where the agent aims to maximize multiple conflicting objectives [33, 34]. Previous work on MORL problems can be classified into three approaches. The first is to combine the objectives into a single scalar value using prefixed weights and to train a policy that maximizes the weighted sum [35, 36]. However, this scalarization approach has three major shortcomings: i) domain-specific expertise is necessary to set the objective weights ii) one specific set of preferences that covers different real-world circumstances is hard to find and iii) the objective preferences are not known before learning. This is why, the other MORL approaches focus on learning optimal policies without fixing the objective weights.

The second category of MORL algorithms compute non-dominated or Pareto optimal solutions. One approach is to randomly initialize a set of policies and optimize using convex combinations of the single-objective gradients in the policy parameter space [37]. Another line of work decomposes the multi-objective problem into sub-problems that are collaboratively solved using a neighborhood-based parameter-transfer strategy [38]. A manifold-based policy search method is introduced to generate a continuous approximation of the Pareto front [39]. Although these methods provide comprehensive algorithms to estimate Pareto solutions, they have a high computational complexity. Finally, the third category of MORL methods relies on learning a meta-policy that can be adapted to different preferences. This approach has been applied to solve continuous control problems [23]. In this thesis, we resort to this approach to solve load balancing problems in real-world scenarios.

1.3 Publications and Achievements

The list of my academic achievements is summarized in Table 1.1. The content of this thesis includes material from the first work entitled “Multi-Objective Load Balancing for Multi-

Band Downlink Cellular Networks:A Meta-Reinforcement Learning Approach“. In addition, the introductory background on RL in chapter 2 is based on the second work “Single and Multi-Agent Deep Reinforcement Learning for AI-enabled Wireless Networks: A Tutorial“.

Table 1.1: Summary of publications and achievements

1.	**A. Feriani**, D. Wu, Y.T. Xu, J. Li, S. Jang, E. Hossain, X. Liu and G. Dudek, “Multi-Objective Load Balancing for Multi-Band Downlink Cellular Networks: A Meta-Reinforcement Learning Approach,” submitted to *IEEE Journal on Selected Areas in Communications.*
2.	**A. Feriani** and E. Hossain, “Single and Multi-Agent Deep Reinforcement Learning for AI-enabled Wireless Networks: A Tutorial,” *IEEE Communications Surveys & Tutorials.*
3.	**A. Feriani**, A. Refaey and E. Hossain, “Tracking Pandemics: a MEC-enabled IoT EcosysTem with Learning Capability,” *IEEE Internet of Things Magazine.*

1.4 Thesis Organization

The organization of this thesis is as follows:

- Chapter 2 presents an overview of the relevant background on RL, MORL and meta-RL;
- Chapter 3 proposes a multi-objective solution for the load balancing problem based on meta-RL. Moreover, we extend the proposed framework using policy distillation to improve its generalization and adaptability;
- Chapter 4 concludes the thesis while pointing out future research directions.

Chapter 2

Background

In this chapter, we introduce preliminaries about the different techniques applied in this thesis. We start by presenting the fundamentals of deep RL, followed by MORL and its associated algorithms. Finally, we provide a brief overview of meta-RL concepts.

2.1 Deep Reinforcement Learning

2.1.1 Markov Decision Process

RL involves an agent interacting with an environment to solve a sequential decision-making problem, often modeled as Markov Decision Process (MDP) [27]. An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho_0)$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and the action spaces respectively. Here, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the probability of transiting from a state s to a state s' after executing an action a; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function that defines the agent's immediate reward for executing an action a at a state s; $\gamma \in [0, 1]$ is a discount factor and $\rho_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution of the environment.

Given a state s, the agent takes an action a transiting the environment to a new state s' given by $\mathcal{P}(\cdot|s, a)$. The agent aims to find a policy π that maximizes its expected return $\hat{r}$

defined as the expected sum of discounted rewards over time:

$$\hat{r} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, \mathcal{R}(s,a) \,\middle|\, a \sim \pi(\cdot|s), s_0 \sim \rho_0 \right].$$

Under a given policy π, we can define the state-dependent value function as the expected accumulative rewards staring from any given state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, \mathcal{R}(s_t,a_t) \,\middle|\, a_t \sim \pi(\cdot|s_t), s_0 = s \right].$$

The state-action value function specifies the value of taking an action a at a given state s and following the policy π thereafter. It is expressed as follows:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, \mathcal{R}(s_t,a_t) \,\middle|\, a_t \sim \pi(\cdot|s_t), s_0 = s, a_0 = a \right].$$

In this thesis, the agent's goal is to learn a deterministic policy to automatically adjust the load balancing parameters such that the network performance metrics are maximized. In what follows, we overview a category of model-free RL algorithms called policy-based methods that will be used in subsequent chapters. We refer the reader to [40] for a detailed overview of different RL algorithms.

2.1.2 Policy-Based Algorithms

Policy-based methods aim to directly learn a parameterized policy π_θ, where θ are the policy parameters, by maximizing the agent's expected long-term reward J as in

$$J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, \mathcal{R}(s_t,a_t) \middle| s_0 \sim \rho_0, a_t \sim \pi_\theta, s_{t+1} \sim \mathcal{P}(s_t,a_t) \right], \tag{2.1}$$

In Policy Gradient (PG) methods [41], the optimal parameters θ^* are learned by performing gradient ascent on the objective J. Using the PG theorem [41], the policy gradients are expressed as in (2.2) and estimated using trajectories $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T}$ collected under the current policy:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{H} \nabla \log \pi_\theta(a_t|s_t)\, Q_\theta(s_t, a_t) \right], \tag{2.2}$$

where H refers to the episode horizon. For the rest of this chapter, we replace Q^{π_θ} by Q_θ for brevity. To estimate the Q_θ function in Eq. 2.2, several methods are proposed. The REINFORCE algorithm [42] uses the rewards-to-go defined as $\sum_{k=t}^{T} \mathcal{R}(s_k, a_k)$. The major caveats of the REINFORCE algorithm are i) that it is well-defined for episodic problems only and ii) it suffers from high gradient variance. To overcome these challenges, actor-critic methods learn an approximation of Q_θ. The learned Q_θ, often called critic, reduces the variance of the gradient estimates but introduces bias. Generalized Advantage Estimation (GAE) [43] is proposed to reduce this bias based on the idea of n-step returns [27]. In what follows, we will briefly summarize the algorithms used in the implementation of our solution.

PG algorithms suffer from sample inefficiency since only one gradient update is performed using the collected trajectories τ. Besides, they are sensitive to the choice of the learning rate since it affects the training performance and can alter the state visitation distribution. This motivates the Trust Region Policy Optimization (TRPO) method [44] where the following constrained optimization problem is solved at each iteration:

$$\theta_{k+1} = \arg\max_\theta L(\theta) = \mathbb{E}_{s \sim \rho_{\theta_k}, a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\theta_k}(s, a) \right] \tag{2.3}$$

$$\text{s.t.} \quad \mathbb{E}_{s \sim \rho_{\theta_k}} \left[D_{\mathrm{KL}}(\pi_{\theta_k}(.|s) || \pi_\theta(.|s)) \right] \le \delta, \tag{2.4}$$

where $\rho_\theta = \rho_{\pi_\theta}$ refer to the state-visitation distribution induced by π_θ, δ is the trust region radius which controls how much the policy is allowed to change per iteration and D_{KL} is the Kullback–Leibler divergence. The original formulation of the TRPO algorithm uses the natural gradients [45] as descent direction and a second-order Taylor expansion of the constraint, thereby yielding to the following policy update rule:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\,\delta}{\nabla_\theta L^T(\theta)\, F^{-1}\, \nabla_\theta L(\theta)}}\; F^{-1}\, \nabla_\theta L(\theta), \tag{2.5}$$

where I refers to the Fisher information matrix. To avoid the expensive inversion of the Fisher matrix, a practical implementation based on the conjugate gradient method [46] is proposed in the original paper. In a similar vein, Proximal Policy Optimization (PPO) [47] algorithm solves the same optimization problem as TRPO but proposes a simpler formulation by introducing a new surrogate loss function:

$$L_{\text{PPO}}(\theta) = \min_{\theta} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\theta_k}(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\delta, 1+\delta \right) A_{\theta_k}(s,a) \right],$$

where “clip” is a function used to keep the value of the ratio $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$ between $1-\delta$ and $1+\delta$ to penalize the new policy if it gets far from the old policy.

2.2 Multi-Objective Reinforcement Learning

The aforementioned single-agent RL framework aims to solve sequential decision-making problems where the agent's actions are evaluated using a *single*, *scalar* reward function. In several wireless communication problems, it is possible to construct a scalar reward function, e.g., beamforming optimization problem where the agent aims to maximize the sum-rate of the users. However, several other problems are inherently formulated using *multiple*, often *conflicting* objectives, e.g., the task offloading problem in a mobile edge computing system, the agent should minimize the latency while maximizing the energy efficiency. In this context, MORL is a more suitable tool to solve control problems where multiple conflicting objectives are optimized.

2.2.1 Multi-Objective Markov Decision Process

An MORL problem is formulated as a *Multi-Objective Markov Decision Process* (MOMDP) which is an extension of the MDP framework. Formally, an MOMDP is presented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \boldsymbol{\mathcal{R}}, \gamma, \rho_0)$ where $\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma$ and ρ_0 are as defined in Section 2.1.1. The reward function $\boldsymbol{\mathcal{R}} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^m$ returns a *vector* of m rewards, corresponding to each objective, where

m is the total number of objectives. Hence, for a given policy π, the expected discounted return is defined as $\mathbf{J}^\pi = [J_1^\pi, \dots, J_m^\pi]$ where

$$J_i^\pi = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \, \mathcal{R}_i(s_t, a_t) \middle| s_0 \sim \rho_0, a_t \sim \pi, s_{t+1} \sim \mathcal{P}(s_t, a_t)\right], \tag{2.6}$$

Maximizing the expected discounted return $\mathbf{J}^\pi$ involves finding policies optimal with respect to (w.r.t) all the objectives by solving the following optimization problem:

$$\max_\pi \mathbf{J}^\pi = \max_\pi [J_1^\pi, \dots, J_m^\pi]$$

To compute the optimal policies for the problem above, we resort to multi-objective optimization (MOO) that we will describe in the next section.

2.2.2 Multi-Objective Optimization

MOO tackles the simultaneous optimization of multiple, possibly conflicting objectives. Formally, an MOO problem is formulated as

$$\max_\pi \mathbf{F}(\pi) = \max[F_1(\pi), \dots, F_m(\pi)],$$

where $F_i(\cdot)$ represents the ith objective function. In the MORL problem defined in Eq. 2.6, the ith objective function corresponds to the expected discounted return for ith objective, i.e., $F_i(\pi) = J_i^\pi$. When the objectives F_i are conflicting, a unique optimal solution cannot be computed since it can either maximize one of the objectives or achieve a trade-off between them. Therefore, MORL algorithms seek to compute a set of control policies that are optimal for different trade-offs between the objectives. To evaluate the optimality of the agent policies, the concept of dominance is often adopted to identify the set of non-dominated solutions:

Definition 1. *A policy π strictly-dominates another policy π' if it has higher values in all the objectives and a strictly higher value in at least one objective [48]. That is to say:*

$$\pi > \pi' \iff \forall i, F_i(\pi) \geq F_i(\pi') \land \exists i,\ F_i(\pi) > F_i(\pi'). \tag{2.7}$$

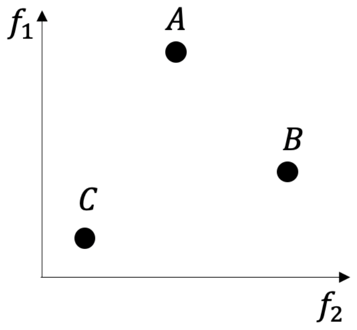
A policy π is said to be Pareto optimal or non-dominated if it is not strictly-dominated by any other policy. Furthermore, a policy π weakly-dominates another policy π' (i.e., $\pi \geq \pi'$) if it verifies:

$$\forall i,\ F_i(\pi) \geq F_i(\pi').$$

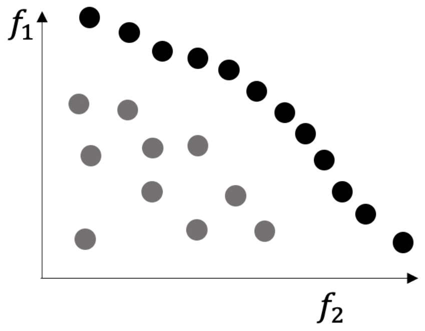
Definition 2. *The Pareto front $\mathcal{F}$ is the image of the set of the Pareto optimal policies in the objective space [34]:*

$$\mathcal{F} = \{\boldsymbol{F}(\pi) \mid \exists!\ \pi',\ \pi' > \pi\}\,. \tag{2.8}$$

In Fig. 2.1, we illustrate an example of the Pareto dominance and Pareto front concepts for a two objective setting. The policy C in Fig. 2.1(a) is dominated by the policies A and B. However, A and B do not dominate each other. To construct the Pareto front, all the dominated solutions (marked in gray in 2.1(b)) are discarded and only the *non-dominated* policies are included (marked in black in 2.1(b)). For real-world problems, it is not plausible to compute the exact Pareto front. Therefore, MOO aims to obtain the set of solutions that best approximates the optimal Pareto front. In the next section, we will detail different MORL algorithms to approximate the Pareto front.



(a) Pareto dominance.



(b) Pareto front

Figure 2.1: Illustration of Pareto dominance and Pareto front concepts [1]

2.2.3 Algorithms

There are three distinct approaches to solve MORL problems: *single-policy* (e.g., [35, 36]), *multiple-policy* (e.g., [37–39]) and *meta-policy* (e.g., [23]) methods which we discussed hereafter to put our contribution in a proper perspective.

Single-policy approaches: This family converts the MOMDP to an MDP by applying a *scalarization function* [49] and solve the resulted MDP using traditional RL methods.

Definition 3. *A scalarization function f converts a vector of multiple objectives $\boldsymbol{F}(\pi)$ to a scalar value given a weight vector or a preference $\boldsymbol{\omega}$:*

$$F_\omega(\pi) = f(\boldsymbol{F}(\pi), \boldsymbol{\omega}). \tag{2.9}$$

A common choice of the scalarization function f is the linear scalarization function where the weighted-sum of the objectives' values and a non-negative weight vector $\boldsymbol{\omega}$ is calculated as follows

$$F_\omega(\pi) = \boldsymbol{\omega} \cdot \boldsymbol{F}(\pi). \tag{2.10}$$

Each weight ω_i corresponds to the relative importance of each objective F_i. The drawbacks of the scalarization approach are four-fold [50]: i) it is not always feasible to know the exact preference vector before the learning phase, ii) the selection of an exact preference vector requires expert and accurate knowledge of the problem, iii) a single weight vector may not cover all possible trade-offs between the objectives, especially in problems characterized with high variability such as wireless communication problems and iv) the linear scalarization approach cannot find policies in the non-convex region of the Pareto front because it only computes a convex combination of the objectives. This motivates other methods to solve MOMDPs where no specific weight vector is known in advance.

Multi-policy approaches: This category of methods aim to learn multiple non-dominated policies at once to approximate the Pareto front. Since the computation of the exact Pareto front is often intractable, there are metrics to measure the quality of the approximated Pareto front. The most well-known metric is called the *hypervolume indicator* [51] which quantifies the uniformity of the solution distribution.

Definition 4. *Given a reference point $\boldsymbol{q} \in \mathbb{R}^m$, the hypervolume indicator $\mathcal{H}(\mathcal{F})$ measures the region weakly-dominated by $\mathcal{F}$ and bounded by $\boldsymbol{q}$, i.e.:*

$$\mathcal{H}(\mathcal{F}) = \Lambda(\{\boldsymbol{z} \in \mathbb{R}^m \mid \exists \boldsymbol{p} \in \mathcal{F} : \boldsymbol{p} \geq \boldsymbol{z} \geq \boldsymbol{q}\}\,, \tag{2.11}$$

where $\Lambda(.)$ is the Lebesgue measure.

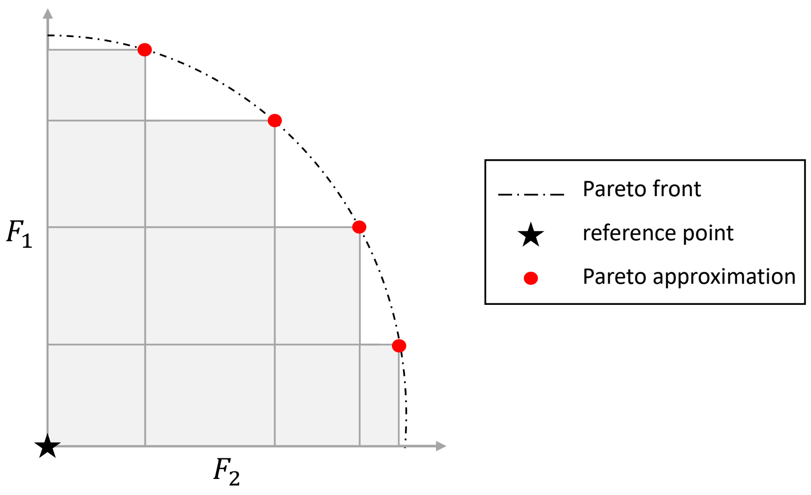


Figure 2.2: Illustration of the hypervolume indicator in 2-objective space : the area (shaded) dominated by the Pareto front approximation and dominating the reference point. Inspired from [2]

Meta-policy approaches: This is the third and last category of MORL solutions. It consists in learning a general policy or a meta-policy that can quickly adapt to new trade-offs between the objectives [23]. By randomly sampling tasks from a given distribution over the objective preferences, the agent learns a meta-policy to maximize the performance on these training tasks. This approach does not explicitly approximate the Pareto set. Alternatively, the meta-policy is finetuned for different preferences for a number of iterations to compute the Pareto front. The work in this thesis falls into this category where we propose the first meta-based MORL solution to solve the multi-objective load balancing problem in wireless communication systems.

2.3 Meta-Reinforcement Learning

Data efficiency is one of the core challenges of real-world deployment of RL algorithms [52]. Indeed, an RL algorithm, especially an on-policy method, requires a dramatic number of interactions to converge to an optimal solution. In contrast, humans can learn to perform new tasks with a limited amount of samples since they rely on their past experiences to successfully learn new skills. Is it possible to design an RL algorithm that can leverage knowledge from previous experiences to rapidly generalize to new/unseen tasks with few training examples? This is the motivation behind the meta-learning or the “learning to learn” paradigm [20, 21].

In traditional RL, the agent aims to solve one MDP. However, in meta-RL, the agent learns a meta-policy that solves multiple tasks from a given distribution. The key objective is to generalize over a distribution of tasks and not a distribution of data instances sampled from a specific task. Formally, the learning procedure involves two steps and two task sets: the meta-training during which the agent is trained on a set of meta-training tasks $\mathcal{T}_{train}$ and meta-testing or finetuning where the agent is evaluated on a set of test tasks $\mathcal{T}_{test}$. The train and test task sets are assumed to be drawn from the same distribution $p(\mathcal{T})$. Each task $\mathcal{T}_i$ is an MDP given by $(\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i, \gamma)$. Note that the tasks can have different dynamics and reward functions. We assume that each task $\mathcal{T}_i$ has training and validation datasets $\mathcal{D}_i = \{\mathcal{D}_i^{train}, \mathcal{D}_i^{val}\}$. In the context of RL, these datasets are the collected trajectories in the environment governed by the task MDP.

Let θ be the parameters of the meta-policy. The goal for each task is to learn task-specific parameters $\theta_i = \mathrm{Alg}(\theta, \mathcal{D}_i^{train})$ starting from θ using $\mathcal{D}_i^{train}$ such that the task loss $\mathcal{L}_i$ on the validation set $\mathcal{D}_i^{val}$ is minimized. In this work, we will focus on one class of meta-RL algorithms which is gradient-based methods. The training phase of these methods can be formulated as a bi-level optimization problem [53] as follows:

$$\theta^*_{meta} = \overbrace{\arg\min_{\theta} L(\theta)}^{\text{outer-level}}, \tag{2.12}$$

$$\text{in which } L(\theta) = \frac{1}{|\mathcal{T}_{train}|} \sum_{i=1}^{|\mathcal{T}_{train}|} \mathcal{L}_i \left(\underbrace{\mathrm{Alg}(\theta, \mathcal{D}_i^{train})}_{\text{inner-level}}, \mathcal{D}_i^{val} \right).$$

Model Agnostic Meta Learning (MAML) [22] is a well-known gradient-based meta-RL algorithm where the inner-level is solved using one (or multiple) gradient descent step(s) as follows:

$$\theta_i = \mathrm{Alg}(\theta, \mathcal{D}_i^{train}) = \theta - \beta \, \nabla_\theta \, \mathcal{L}_i(\theta, \mathcal{D}_i^{train}). \tag{2.13}$$

where β is a learning rate.

In the meta-testing phase, the trained meta-policy π_θ can be finetuned to tasks $\mathcal{T}_i \in \mathcal{T}_{test}$ by solving the inner-level problem using (2.13). The key objective is to learn a good model initialization that can be quickly adapted to new tasks with few gradient steps and performs well on the testing tasks.

Chapter 3

Communication Load Balancing via Meta Multi-Objective Reinforcement Learning

Load balancing has emerged as a critical technique to deal with spatio-temporal load fluctuations by ensuring a more efficient and even resource allocation. This chapter proposes a new load balancing solution based on MORL and meta-RL techniques. The main contributions are as follows:

- We propose a novel RL-based load balancing framework that simultaneously maximizes conflicting network KPIs without using prefixed preferences. To this end, we first formulate the load balancing problem as an MOMDP to handle different trade-offs between the KPIs;

- To solve the underlying MOMDP problem, we opted for a meta-RL based algorithm. This choice is motivated by the data efficiency of this approach enabling the quick adaptation to new trade-offs using few environment interactions and gradient steps;

- We modify and extend the vanilla meta-training loop and propose a new technique based on policy distillation to guide the meta-policy learning process and improve its generalization for different traffic scenarios;

- We demonstrate the effectiveness of our framework on realistic traffic scenarios. Our solution i) outperforms existing rule-based and single-objective load balancing methods, ii) computes better Pareto front approximations compared to several mutli-policy MORL algorithms, and iii) quickly adapts to new trade-offs.

The rest of the chapter is organized as follows: the system model is presented in Section 3.1. Section 3.2 briefly introduces the considered load balancing control mechanisms. In Section 3.3, the problem formulation of the MOMDP is described. The proposed meta multi-objective framework for load balancing is detailed in Section 3.4. Finally, we provide a detailed description of the experimental setup in Section 3.5, followed by an exhaustive numerical results in Section 3.6.

3.1 System Model

3.1.1 Network Topology

We consider a multi-band downlink Orthogonal Frequency Division Multiplexing (OFDM) network modeled as a regular grid following a hexagonal layout. It consists of M macro-sites, equipped with M macro-eNBs with inter-site distance D. Each macro-site is three-fold sectorized and each sector operates on N_c frequency bands.

Each user is served by one cell $C_{i,j,b}$ where i, j and b are the eNB, the sector and the frequency band respectively. Both the frequency bands and bandwidths are homogeneous for the whole network, meaning that $C_{i,j,b}$ has the same carrier frequency and bandwidth for any sector $j \in [3]$ and any eNB $i \in [M]$. Each geographic macro-site operates $3 \times N_c$ cells. We denote by C the total number of cells in the network. Figure 3.1 depicts an example of the considered network layout.

In this chapter, we restrict our study to the downlink while considering co-channel interference based on varying traffic demand per area. It is assumed that the time is slotted in Transmission Time Intervals (TTIs) of a fixed duration where at most one user can be served over each sub-carrier.

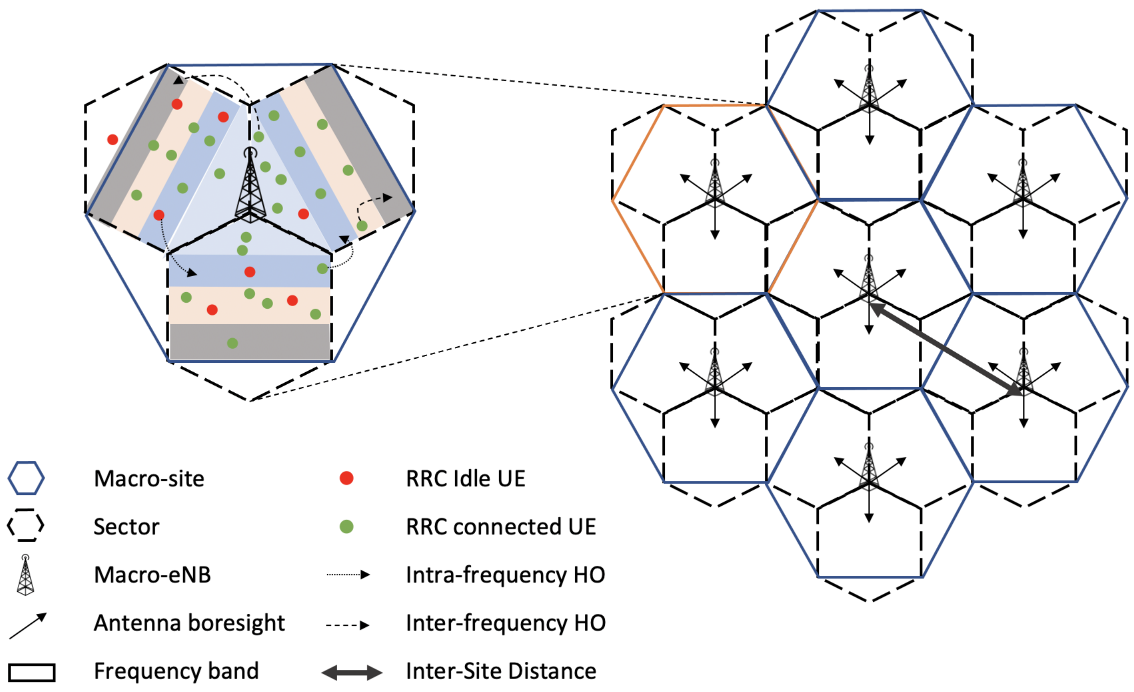


Figure 3.1: Hexagonal macro-cell with inter-site distance D layout, $M = 7$ macro-sites and $N_c = 4$ frequencies, per sector. UEs are uniformly distributed.

3.1.2 Propagation Model

The macro-sites deploy directional antennas with a two-dimensional radiation pattern. As in [3], the antenna gain is expressed as

$$G(\psi, \varphi) = -\min\left\{G_H(\psi) + G_V(\varphi), G_m\right\}, \tag{3.1}$$

where ψ and φ are the azimuth and elevation angles between the transmitter and receive antennas, respectively, computed using geometric locations. G_H and G_V refer to the horizontal and vertical antenna patterns defined in [3] and G_m is the maximum attenuation. We assume the UEs have antennas with omni-directional radiation pattern with 0 dBi antenna gain. Thus, the total received power, in dBm, for a user u served by a cell $c = C_{i,j,b}$ is defined as

$$P_{c,u}^{rx} = P_c^{tx} + G_{c,u} - L_d + \mathcal{X}_\sigma - L_p, \tag{3.2}$$

where P_c^{tx} is the transmitted power in dBm, $G_{c,u}$ is the antenna gain, L_d is the path loss in dB, $\mathcal{X}_\sigma$ is the shadow fading parameter modeled as a zero-mean Gaussian random variable with a standard deviation σ in dB and L_p is the penetration loss measured in dB. The path loss is given by

$$L_d = L_{d_0} + 10 \cdot \nu \cdot \log_{10}\left(\frac{d}{d_0}\right), \tag{3.3}$$

where L_{d_0} is the free-space reference path loss at a distance d_0, ν is the path-loss exponent, d is the distance between the UE and its serving cell c in Km.

3.1.3 Traffic Model and Scheduling

To estimate the cell load in terms of utilized resources, we assume a file transfer protocol traffic model [3] where users request a sequence of file transfers separated by a reading time (see Fig. 3.2). The latter consists of the time between two successive file transfers. During the reading time, the user is considered inactive. The reading time is modeled as a Poisson process with mean λ and the file size is fixed to L bits.

In OFDM systems, orthogonal resource allocation is adopted within a cell such that users in the same cell are allocated non-overlapping resources. Thus, there is no intra-cell interference. We assume that each cell c has N_c^{tot} physical resource blocks (PRBs) to be scheduled per

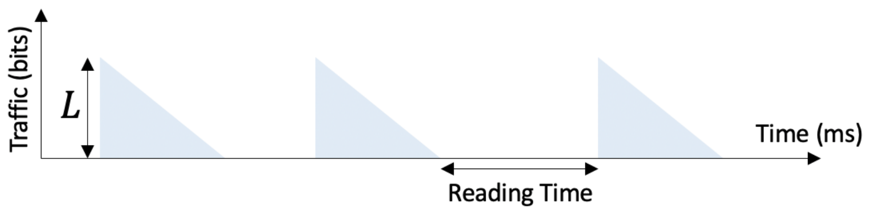


Figure 3.2: Traffic generation model per user according to [3]

TTI. Proportional Fair scheduling algorithm is implemented to provide an optimal trade-off between throughput and fairness [54]. The key idea of the proportional fair scheduler is to allocate resources to the UEs according to their proportional fairness value. Given the described traffic model and scheduling algorithm, we can characterize the signal to interference and noise ratio (SINR) and estimate the load of each cell.

3.1.4 SINR and Load Estimation

As mentioned above, orthogonal resource allocation is assumed within a given cell, thereby intra-cell interference does not occur. To compute the inter-cell interference, we introduce the vector $\boldsymbol{X} = [X_1, \ldots, X_C]^T$ such that each element $X_{c'} \in \{0, 1\}$ indicates if the cell c' is causing interference on a specific resource unit (i.e., PRB) used by the cell c during data transmission to a user u. Note that we only consider interfering cells operating on the same carrier frequency as the cell c. Thus, the SINR at the user u, served by a cell c, is expressed as follows:

$$\gamma_{c,u} = \frac{P^{rx}_{c,u}}{\sum_{c' \neq c} P^{rx}_{c',u} \cdot X_{c'} + N_0}, \tag{3.4}$$

where N_0 is the additive white Gaussian noise power in dBm. Therefore, the required PRBs to meet the demand of the user u is

$$N_u = \frac{R_u}{B \ \log_2 (1 + \gamma_{c,u})}, \tag{3.5}$$

where B is the bandwidth of one PRB and R_u is the required data rate by the user u. Consequently, the total cell load, measured in terms of PRBs, is given by

$$\rho_c = \frac{\sum_{u \in \mathcal{U}_c} N_u}{N_c^{tot}}, \tag{3.6}$$

where $\mathcal{U}_c$ is the set of scheduled users in cell c. Using this definition of the cell load, it is straightforward to observe that the higher the resource block utilization ratio is, the more crowded the cell is. When a cell c completely utilizes its available resources (ρ_c close to 1), its users will either experience low throughput or their connections can be dropped. Thus, load balancing can improve the network resource utilization by offloading UEs from the cells experiencing high load to other under-loaded cells.

3.2 Load Balancing Mechanisms

In LTE, UEs can be in one of the two RRC states : RRC idle or RRC connected. RRC idle UEs perform neighboring cell measurements and cell re-selections while monitoring a paging channel to detect incoming calls. When a UE switches to an RRC connected state, it becomes able to send and receive data from the network, measure channel quality, and report neighboring cell measurements, measurement reports, and feedback information. Amongst the measurements reported by RRC connected UEs, Reference Signal Receive Power (RSRP) and Reference Signal Receive Quality (RSRQ) are key quantities used in LTE for HO triggering events [55]. We further consider two types of RRC connected UEs: active UEs are UEs undergoing data reception or transfer. Otherwise, the UEs are considered inactive. As mentioned before, during inter-file arrival time, the UEs are considered inactive. It is possible to switch from an RRC connected state to an RRC idle state. If the inter-file arrival time is larger than a given inactivity time, the state of the UE changes from inactive to idle.

Two types of HO/cell re-selection can be triggered: *intra-frequency* where the UE moves from one sector to another without changing the serving frequency band, and *inter-frequency* which consists in changing the UE's serving frequency band and possibly the serving sector depending on the user mobility (see Fig. 3.1). In this work, we will focus on inter-frequency load balancing instead of intra-frequency since the former is preferred when i) a strong interference exists, the intra-frequency load balancing will not help because the UE will suffer from the same level of interference and ii) the serving frequency suffers from a coverage hole. For the rest of the thesis, we denote by c and t the indices of the serving and target cells of a given UE, respectively, and M refers to the RSRP/RSRQ measurements.

3.2.1 Idle Model Load Balancing (IMLB)

When a UE is in idle mode, it needs to select which cell to camp on. The objective of IMLB or cell re-selection is to guarantee that the UE camps on the best cell in terms of radio conditions measured by the RSRP or RSRQ (see 3GPP TS 38.304 for more details). In multi-band LTE systems, frequency bands have different priorities and inter-frequency cell re-selection consists in redirecting the UEs to the highest priority frequency band available. Indeed, it is possible to indicate a specific priority for an idle UE through the RRC Connection Release message.

Formally, IMLB is triggered when the load of the current serving cell exceeds a given threshold T_c^{IMLB} and the target neighboring cell t is selected such that:

$$M_t > O_{c,t},$$

where $O_{c,t}$ is a re-selection offset between the two cells. Once the target cell is selected, its re-selection priority is set to the maximum value. Thus, when the idle UE connects to the network, it will be associated with the selected cell. Consequently, adjusting the re-selection offsets helps redistribute the idle UEs and avoid future congestion when they are activated.

3.2.2 Connected Mode Load Balancing (CULB)

In the RRC connected mode, the UEs are constantly measuring the channel quality between their serving and neighboring cells. Based on these measurement reports, congested cells can redirect UEs to other less-loaded cells through HO to achieve better service quality. When the reported RSRP and/or RSRQ measurements satisfy certain event conditions, HO is triggered. In this work, we will consider inter-frequency HO triggered by A_2 and A_5 events. A_2 event signals if the channel quality of the serving cell c drops below a certain threshold and A_5 event is satisfied when the channel quality of the serving cell c drops below a threshold and the channel quality of a neighboring cell t becomes better than a threshold [55]. The A_2 event when triggered identify the overloaded cells and A_5 event is used for the selection of the HO target cell. In LTE, these conditions are defined as follows

$$A_2: \; M_c < T_{A_2}, \tag{3.7}$$

$$A_5: \; M_c < T^1_{A_5} \text{ and } M_t > T^2_{A_5}, \tag{3.8}$$

where T_{A_i} are event-specific thresholds that define the HO parameters. Decreasing the T_{A_2} and $T^1_{A_5}$ thresholds will result in offloading more UEs from cell c to cell t. This is why tuning these parameters is crucial to avoid early or late handover and engender link failures.

3.3 Problem Formulation

In this section, we start by defining the network performance metrics that represent the objectives in our multi-objective load balancing formulation. To further motivate the necessity of a multi-objective solution for the load balancing problem, we provide a use case to demonstrate the conflict between the considered network KPIs. Afterward, we detail the MOMDP design including the states, actions, and reward functions.

3.3.1 Network Performance Metrics

Minimum throughput (T_{min}) measures the minimum throughput across cells. By maximizing the T_{min} objective, we seek to improve the throughput of the cell with the lowest throughput. Unlike other RL-based load balancing work, this max-min formulation ensures a fair load balancing policy [56]. Maximizing the sum throughput may result in an undesirable offloading decision where the cell with the worst throughput is sacrificed (see the motivating example below).

$$T_{\text{min}} = \min_{c \in C} \frac{T_c}{\Delta t}, \tag{3.9}$$

where T_c is the estimated value of the total throughput of the cell c measured in megabits per second during a fixed interval of time Δt;

Standard deviation of the throughput (T_{std}) measures the performance gap and the resource unbalance between the cells. To achieve an even distribution of the resources, T_{std} should be minimized.

$$T_{\text{std}} = \sqrt{\frac{1}{C} \sum_{c \in C} \left(\frac{T_c}{\Delta t} - \bar{T} \right)^2}, \tag{3.10}$$

where $\bar{T}$ is the average throughput over all cells.

Although both KPIs are constructed to improve the user experience, they can be conflicting. To showcase this conflict, consider the example of one sector with fours cells of different coverage ranges as illustrated in Fig. 3.3. When most of the UEs reside far from the eNB and can only connect to the two cells with the largest coverage area (cells 3 and 4), these two cells will have low throughput, while the other cells (1 and 2) with smaller coverage can have high throughput (Fig. 3.3-a). On one hand, maximizing the T_{min} objective only will result in offloading users from cell 4 to cell 3 (Fig. 3.3-b). On the other hand, a naive solution to minimize the T_{std} objective will relieve one of the low throughput cells, e.g., cell 3, by moving

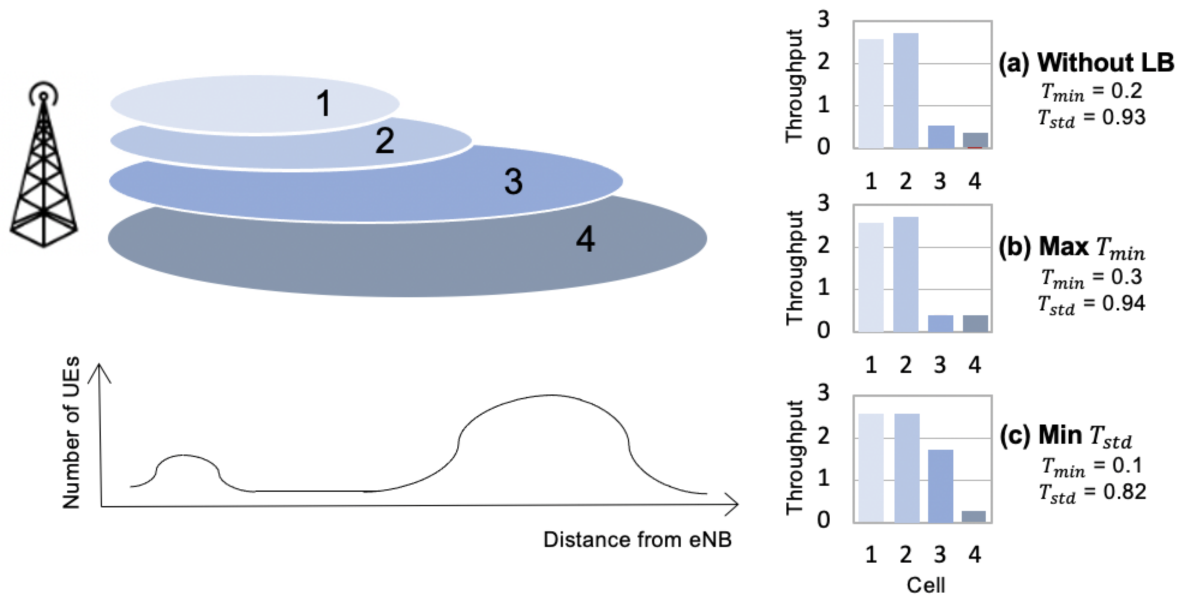


Figure 3.3: An example of the trade-off between the T_{min} and T_{std} objectives.

most of its UEs to cell 4 (Fig. 3.3-c). This will increase the throughput of cell 3, but it will also make cell 4 more congested, thereby decreasing the T_{min} metric. Finding a solution that can simultaneously improve T_{min} and reduce T_{std} is challenging, which highlights the importance of a multi-objective solution.

Consequently, we propose the following *multi-objective* load balancing problem:

$$\text{maximize} \left[T_{\text{min}}, \frac{1}{T_{\text{std}}}\right] \tag{3.11}$$

$$\text{s.t.} \sum_{u \in \mathcal{U}_c} N_u \leq N_c^{tot}, \ \forall \, c \in [C].$$

3.3.2 MOMDP Formulation

We now propose the following MOMDP formulation for the load problem (3.11):

- **State**: The state contains i) the number of active UEs per cell, ii) traffic load for each cell ρ_c, and iii) the throughput per cell T_c. Thus, the state space is continuous and its dimension is $3\,C$;

- **Action**: We simultaneously control IMLB and CMLB parameters. Thus, the action is a vector of re-selection/HO parameters $O_{c,t}, T^1_{A_5}, T^2_{A_5}$ and T_{A2} as introduced in Section 3.2;
- **Reward**: We focus on the two objectives defined in Eqs. 3.9 and 3.10. Since these metrics have different scales, we constructed scaled reward functions for each performance indicator. R_i is the reward for the $i-$th objective to optimize

$$\begin{aligned} R_1 &= \frac{1}{4.9} T_{\min} \\ R_2 &= \frac{1}{2.4 \cdot (1 + T_{\mathrm{std}})}. \end{aligned}$$

  To select the scaling coefficients in the reward functions R_1 and R_2, we perform a grid search over multiple possible scaling factors. We pick the coefficients that result in the best performance in terms of reward.

3.4 Proposed Solution

After formalizing the load balancing problem as MOMDP, we can readily apply the MORL algorithms detailed in Section 2.2.3. We decided to adopt an approach based on meta-RL since our goal is not only to find Pareto-optimal policies but to quickly adapt to new trade-offs between the objectives. Load balancing is a challenging real-world problem especially due to the constant changes in the wireless environment, user mobility, and traffic patterns. Consequently, the trade-offs between the objectives need to vary to accommodate these changes, hence the importance of the rapid adaptation in the design of our multi-objective load balancing framework.

3.4.1 Meta-RL for Load Balancing

Our first solution is a gradient-based meta-RL framework inspired from the MAML algorithm As explained in chapter 2, the training process of MAML interleaves two optimization steps as depicted in Fig. 3.4: i) a task adaptation phase where a number of policies are learned starting from the meta-policy parameters, ii) a meta-adaptation phase that adjusts the meta-parameters using trajectories sampled from the adapted policies. These two steps are repeated for a fixed number of meta-iterations N_{meta} [22].

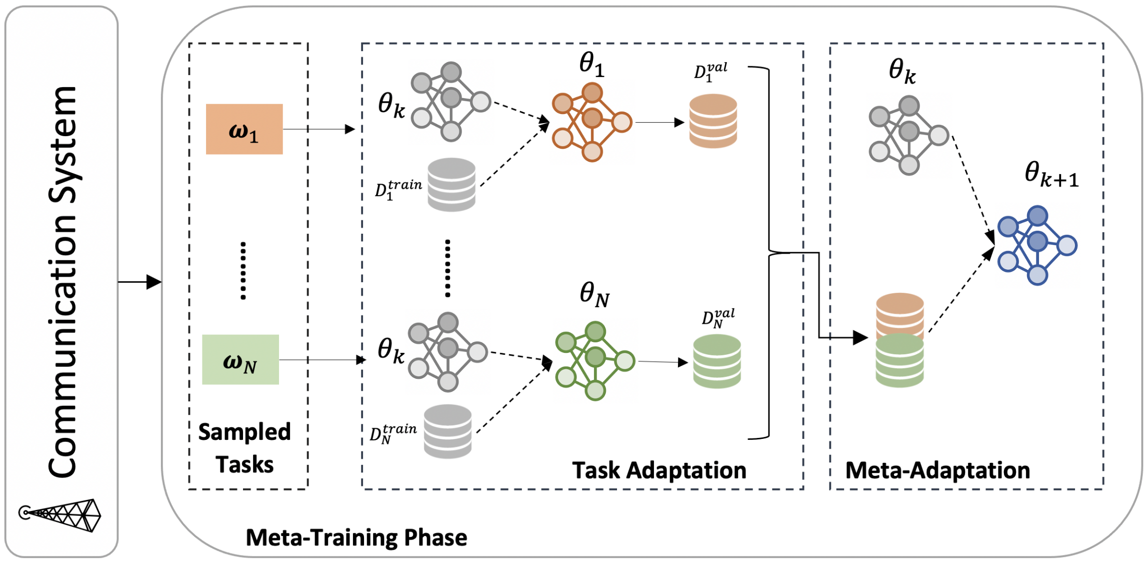


Figure 3.4: Multi-objective load balancing solution based on meta-RL. θ_k denotes the meta-parameters at a given meta-iteration k.

Once the training is finished, the meta-policy can be used as initialization and finetuned for a number of gradient steps to approximate the Pareto front. We will call this step as the finetuning phase. In addition, the meta-policy is an efficient starting point to quickly learn optimal solutions for new tasks or preferences. In the literature, different nomenclatures are attributed to the two steps of the meta-training. In this work, we opted for task adaptation to refer to the inner-level problem and meta-adaptation for the outer-level optimization problem. Algorithm 1 summarizes the steps of our solution.

Algorithm 1 Meta-RL for Multi-Objective Load Balancing

```
1: Input: $p(\boldsymbol{\omega})$: the preferences distribution, $N_{\text{meta}}$: number of meta-iterations, $N$: number
   of tasks per meta-iteration, $K$: number of trajectories sampled per task.
2: Initialize meta-policy $\pi_\theta$ randomly or using $\theta_{\text{PD}}$
3: for $t = 0, ..., N_{\text{meta}}$ do
4:     Task Adaptation
5:     Sample $N$ preference vectors $\boldsymbol{\omega}_i \sim p(\boldsymbol{\omega})$;
6:     for each weight vector $\boldsymbol{\omega}_i$ do
7:         Sample $K$ trajectories $\mathcal{D}_i^{train}$ using the meta-policy $\pi_\theta$
8:         Estimate $\nabla_\theta \mathcal{L}_i(\theta, \boldsymbol{\omega_i})$ using $\mathcal{D}_i^{train}$
9:         Compute the adapted parameters $\theta_i$ using (3.13)
10:        Collect trajectories $\mathcal{D}_i^{val}$ using the adapted policies $\pi_{\theta_i}$ in $\mathcal{T}_i$
11:    end for
12:    Meta Adaptation
13:    Update meta-policy with $\mathcal{D}_i^{val}$ and $\boldsymbol{\omega}_i$ using (3.14)
14: end for
15: Finetuning: Fine-tune the meta-policy for a number of iterations using (3.13) to ap-
    proximate the Pareto front.
```

Task Adaptation

In this phase, N preference vectors are randomly sampled from a specific distribution $p(\boldsymbol{\omega})$ such that each weight element ω_j is positive and $\sum_{j=0}^{m} \omega_j = 1$. Note that each preference $\boldsymbol{\omega}_i$ corresponds to a task $\mathcal{T}_i$ where the reward function is given by $\sum_i \omega_i \mathcal{R}_i$. In the rest of the thesis, we will use the terms preference and task interchangeably.

For each task, a policy is updated for a limited number of gradient steps using the meta-parameters θ as initialization.

The objective is to minimize the following inner loss $\mathcal{L}_i$

$$\mathcal{L}_i(\theta, \boldsymbol{\omega_i}) = -\mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[\sum_{t=0}^{H} \boldsymbol{\omega}_i^\top \left(\hat{\mathbf{r}}_i(s_t, a_t) - \mathbf{V}_i(s_t)\right)\right], \tag{3.12}$$

where $\hat{\boldsymbol{r}}$ is the return and $\mathbf{V}_i$ is the estimated value function for the task $\mathcal{T}_i$. To estimate the gradients of the inner loss in 3.12, trajectory data $\mathcal{D}_i^{train}$ is collected for each task $\mathcal{T}_i$ by following the meta-policy π_θ in the corresponding MDP. The task-specific policies θ_i are updated using one or more gradient steps as follows

$$\theta_i \mapsto \theta_i - \beta\, \nabla_\theta\, \mathcal{L}_i(\theta; \boldsymbol{\omega}_i), \tag{3.13}$$

where β is the step size for the task adaptation phase.

Meta-Adaptation

The meta-learner aggregates trajectories $\mathcal{D}_i^{val}$ sampled using the obtained policies from the previous step and adjusts the meta-policy parameters θ by differentiating through the adaptation phase to minimize the task-specific errors as in:

$$\theta \mapsto \theta - \eta\, \nabla_\theta \sum_{i=1}^{N} \mathcal{L}_i(\theta_i; \boldsymbol{\omega}_i), \tag{3.14}$$

where η is the step size for the meta-adaptation phase. Note that the gradients in (3.14) are estimated using the datasets $\mathcal{D}_i^{val}$ (see Fig. 3.4). The meta-policy, learned using multiple trade-offs, will have an inductive bias that will enable the fast adaptation to new tasks similar to the ones used during training.

3.4.2 Meta-RL and Policy Distillation for Load Balancing

Learning a general meta-policy can be challenging due to multiple reasons. First, the task adaptation step explained above requires collecting multiple trajectories for each task. Generally, the number of these trajectories is limited to ensure rapid adaptation with few samples.

Further, θ and θ_i have the same parameter space which could be in the order of millions in deep neural networks. In addition, learning one initial condition for a large family of tasks, such as in our case, is not trivial.

To account for these challenges, we propose to combine policy distillation and meta-RL. Policy distillation combines the knowledge from different tasks into a single policy that can be used as task-specific prior to warm-start the meta-training. This will help to i) get better task-specific policies with few samples since we assume that some of the preferences encountered during the task adaptation phase can be similar to the tasks used during policy distillation, ii) explore the parameter space more efficiently in both optimization steps.

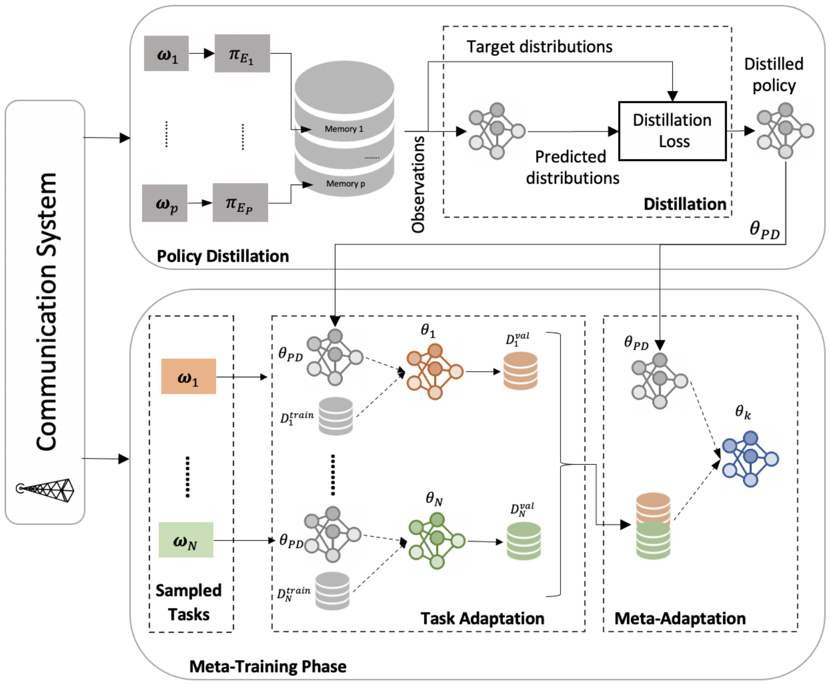


Figure 3.5: Multi-objective load balancing solution based on meta-RL and policy distillation. θ_k denotes the meta-parameters at a given meta-iteration k.

As depicted in Fig. 3.5, our second solution consists of two stages. The policy distillation stage starts by selecting $p \neq N$ preferences $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_p\}$ and training p task-specific policies for each weight vector to maximum performance. We will refer to these task-specific policies as teachers. Next, the trained teachers E_i are used to collect trajectories which will be saved in separate memory buffers. The distilled policy $\pi_{\theta_{\text{PD}}}$ is learned to match the teachers' state-dependent action probability distributions $\pi_{\theta_{E_i}}$ by minimizing the Kullback-Leibler (KL) divergence as follows:

$$\mathcal{L}_{\text{KL}}(\theta_{\text{PD}}, s) = \sum_{a \in \mathcal{A}} \pi_{\theta_{E_i}}(a|s) \log \left(\frac{\pi_{\theta_{E_i}}(a|s)}{\pi_{\theta_{\text{PD}}(a|s)}} \right).$$

The second stage of this solution is the meta-policy training starting from the distilled parameters θ_{PD}. The same learning procedure as in the meta-RL solution is followed.

3.5 Experimental Setup

3.5.1 Simulator

Our experiments are conducted using a proprietary System Level Simulator (SLS), which emulates traffic for 4G/5G communication networks based on real-world data. The simulation layout is as illustrated in Fig. 3.1. Each sector operates on four frequency bands, leading to 12 cells in each macro-site. We follow the same setting as in [57] and focus on balancing the load between different frequency bands for the first sector.

Initially, UEs are uniformly distributed across the cells. These devices are either static or undergo a random motion with a fixed velocity which is set to 3 m/s. As explained in Section 3.1, we assume a file transfer protocol traffic model, as in 3GPPP TR.36814, where each UE downloads a file with exponentially distributed reading time. The size of the file and the mean of the reading time distribution are time-dependent and aligned with the traffic patterns in the real world. Table 3.1 lists the simulation parameters.

Table 3.1: Simulation parameters

Parameters	Values
Inter-site distance D	500 m
Path-loss	$128.1 + 37.6\log_{10}(d)$
# of frequency bands	4
Bandwidth	$20, 10, 5, 10$ MHz
Number of RBs N^{tot}	$100, 50, 25, 50$
Penetration loss	20 dB
Std dev. of shadowing σ	8 dB
Transmit power p^t	$46, 46, 46, 49$ dBm
Traffic model	File Transfer Protocol
Scheduler	Proportional Fair
File size	$[0.5, 2]$Mbytes
Inter-file arrival mean λ	$[10, 320]$ ms
UE velocity	3 m/s
UE mobility model	Static/random motion
Thermal noise	-174 dBm/Hz
Δt	60 min

3.5.2 Baselines

We consider two families of baselines summarized in Table 3.2. The first one is the single-policy load balancing algorithms that include rule-based and single-objective RL-based methods. The second one is multi-policy algorithms allowing the comparison of the Pareto front approximation. A detailed description of the baseline algorithms is provided in Appendix A.1.

Table 3.2: Selected baselines

Baselines	Description
Single-policy	
No LB	no load balancing
Rule-based LB (RuleLB) [57]	uses fixed thresholds for all traffic scenarios
Adaptive LB (Adaptive) [11]	adapts the load balancing parameters based on the cells' load measurements
RL-based LB (RL)	trains an RL-based policy from scratch starting from a random initialization
Multi-policy	
Radial Algorithm (RA) [37]	samples a set of weights and runs RL to optimize a policy for each weight
Random Selection (RS)	uses a random selection strategy that uniformly sample weights in each iteration
Pareto-Following Algorithm (PFA) [37]	samples a set of weights and policies and gradually finetune the weight associated with each policy to cover the whole Pareto front

3.5.3 Model Details and Hyperparameter Settings

Our MAML implementation is based on that of Deleu [58]. The meta-policy network has three hidden layers of 256 units each. Vanilla policy gradient algorithm (REINFORCE) [59] is implemented to compute gradient updates in the task adaptation (i.e., Eq. 3.13), and TRPO [44] is used for the meta-adaptation (i.e., Eq. 3.14). Similar to the work by Fin *et al.* [22], the value function, used in both task and meta-adaptation phases, is a linear feature model fitted separately for each task. The learning rate β is set to 0.1 during the meta-training and 0.003 for the finetuning phase. The episode length is $H = 24$ time steps. Only $N = 5$ tasks are used in each meta-iteration and, for each task, $K = 10$ trajectories are sampled. The preferences are sampled from a Gaussian distribution restricted to be positive and L_1-normalized. The meta-policy is trained for $N_{meta} = 500$ iterations.

For the policy distillation stage, the teacher and student models have the same architecture as the meta-policy. In our experiments, we trained $p = 3$ expert policies using PPO [47]. The preferences used for policy distillation are set such that the weight on the $T_{\min}$ decreases (i.e $\omega_1 = 0.8, 0.5, 0.2$). We found that training the distilled policy for too long on the source tasks harms the knowledge transfer to the meta-policy. This can be explained by the fact that the distilled policy is over-fitted on source tasks. Hence, we train the distilled policy for 10 epochs on only 10 episodes for each teacher. More information about the hyper-parameters is provided in Appendix A.2.

3.5.4 Traffic Scenarios

The proposed framework is evaluated on different traffic scenarios corresponding to different UE settings (e.g., number and distribution per cell) and wireless environment parameters (e.g., reading time, file size). These conditions are used by SLS to mimic the entire communication chain between the transmitter (i.e., eNB) and the receiver (i.e., UE).

In this thesis, we consider two scenarios characterized by low and high traffic conditions as depicted in Fig. 3.6. The fourth cell in the high scenario has higher active UEs and the highest load ratio compared to the other cells. Hence, this cell can benefit from load balancing to evenly distribute the UEs across the other cells.

Furthermore, we examine the daily traffic pattern in terms of the number of active UEs, cell throughput, and percentage of used PRBs by running the rule-based baseline. From

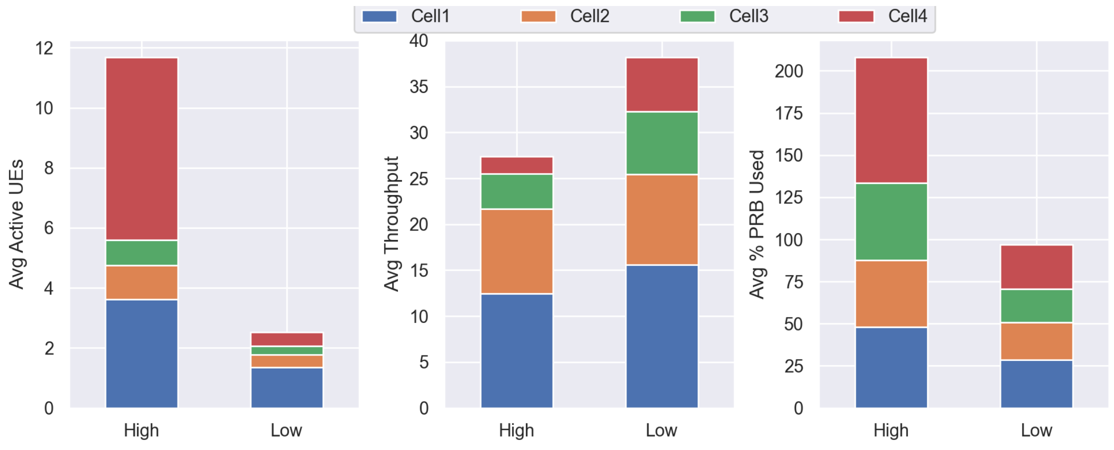
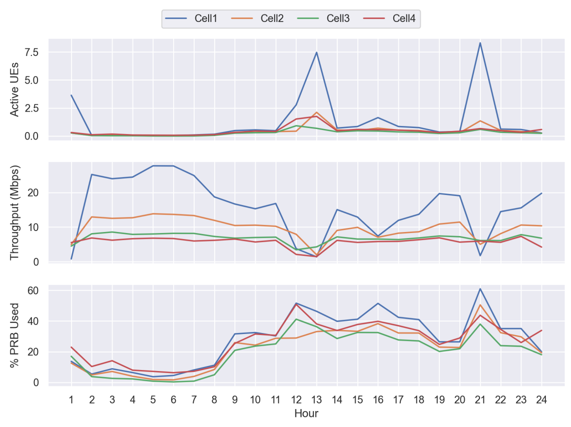


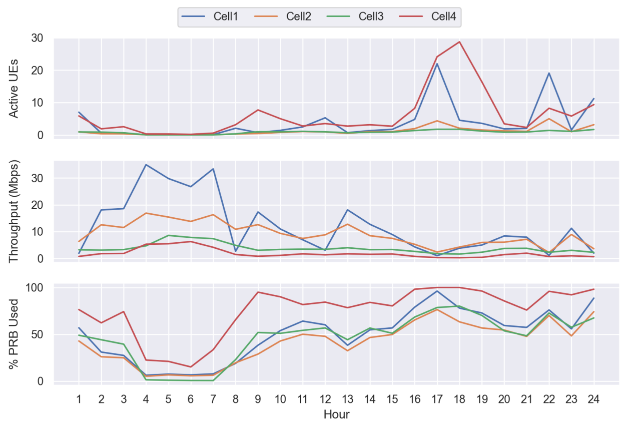
Figure 3.6: Average traffic over one day for one sector for different traffic scenarios: (Left) Average active UEs per cell. (Center) Average throughput per cell. (Right) Average cell load ratio.

Fig. 3.7, we can observe the temporal variations in the traffic and recognize a surge in network utilization, generally starting from 9 am. This is not surprising since most of the users are active during the daytime. Note that each traffic scenario is characterized by different peak hours. Load balancing is mostly needed during such peaks to prevent link failures and service disconnection.

Now, we examine the status of each cell. A congested cell is characterized by a high number of active UEs, low throughput, and a high PRB utilization percentage. From Fig. 3.7(a), for the low traffic scenario, the cell loads are comparable except for peak hours where cell 1 undergoes a sharp increase in the number of UEs and a considerable drop in throughput. This results in a decrease in the $T_{\min}$ objective. Whereas, for the high traffic scenario (Fig. 3.7(b)), we observe that cell 4 has the lowest throughput and the highest PRB utilization compared to the other cells. In addition, we notice that the high traffic scenario is characterized by a large percentage of used PRBs compared to the low traffic case. Indeed, the percentage of used PRBs, in the high traffic scenario, is above 50% for most of the high



(a) Low traffic



(b) High traffic

Figure 3.7: Illustration of the daily traffic pattern for each traffic scenario

utilization hours and reaches 100% for cell 4. However, all the cells in the low traffic scenario use less than 60% of their available resources.

3.6 Numerical Results and Discussion

In this section, we study the performance of our proposed solutions and compare them to several baselines. We will denote our vanilla meta-RL solution by MeMo-LB and our augmented meta-RL solution with policy distillation by MeMoPD-LB.

3.6.1 Meta-policy Performance

In this section, we examine the performance of the meta-policy without further finetuning and compare it with the rule-based load balancing approaches. We only compare with the rule-based methods since they compute the same parameters regardless of the selected preferences. Note that since the meta-policy is not task dependent, its performance cannot be directly compared with RL-based methods since they are trained with a specific preference.

The results in Table 3.3 show that load balancing has a major impact on improving the network performance compared to the case where no load balancing is applied. Furthermore, Table 3.3 demonstrates that our proposed solutions outperform the classical rule-based methods on both KPIs. The meta-policy of MeMo-LB achieves a relative improvement of 10% (31%) and 19% (13%) on the $T_{\min}$ and T_{std} objectives over the rule-based baseline for the low (high) traffic scenario. This means that MeMo-LB can simultaneously improve the throughput of the worst cell while maintaining a balanced load distribution. For the high traffic scenario, a greater improvement is achieved on $T_{\min}$ compared to the low traffic scenario, which proves that the impact of load balancing depends on the traffic distribution.

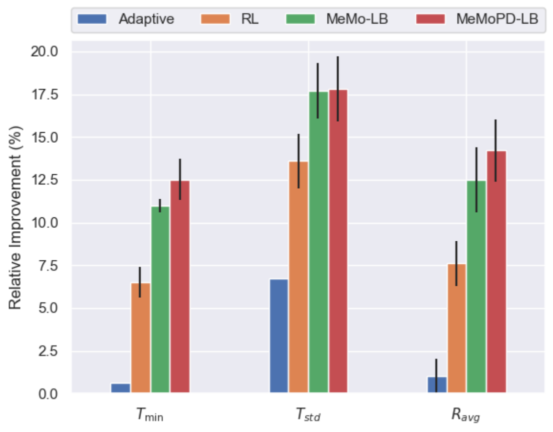
Augmenting our MeMo-LB with policy distillation (MeMoPD-LB) improves the performance on both traffic scenarios. A further improvement of 1.7% and 4.1% on the $T_{\min}$ and T_{std} KPIs is obtained for the high traffic scenario and 1.5% increase on $T_{\min}$ for low traffic scenario. Hence, the knowledge transfer through policy distillation improves the generalization of the meta-policy. This observation is consistent with other work on the impact of knowledge distillation on multi-task learning [60].

		No LB	RuleLB	Adapative	MeMo-LB	MeMoPD-LB
Low	$T_{\min}$	5.10	5.40	5.43	5.92	**6.01**
	T_{std}	4.19	4.34	4.05	**3.53**	3.56
High	$T_{\min}$	1.73	1.80	1.95	2.35	**2.39**
	T_{std}	5.89	4.48	4.33	3.90	**3.74**

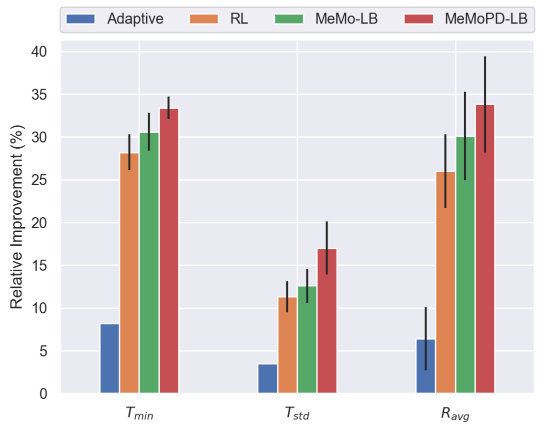
Table 3.3: Evaluation of our methods and baseline algorithms on both traffic scenarios.

3.6.2 Comparison with Single-Policy Methods

In this section, both our methods are compared to single-objective RL methods. In addition to the network KPIs, we consider the average scalarized rewards R_{avg} as an evaluation metric. Given a preference vector $\boldsymbol{\omega}$, the scalarized reward is given by $R = \omega_1 \cdot R_1 + \omega_2 \cdot R_2$, where R_1 and R_2 are defined in Section 3.4. The evaluation process starts by sampling N test



(a) Low Traffic



(b) High Traffic

Figure 3.8: Average relative improvement w.r.t. RuleLB baseline for both performance metrics and the scalarized reward across different preferences. Error bars show one standard deviation. Note that the adaptive method yields identical performance metrics across runs.

preferences uniformly in $[0, 1]$ with a step of 0.1. Then, the meta-policy is finetuned using Eq. 2.13 to compute a policy for each test preference. The number of finetuning steps is limited to 300 gradient updates. To ensure a fair comparison, the RL baseline is also trained for the same number of gradient steps but starting from a random initialization instead of starting from the meta-policy.

As shown in Fig. 3.8, MeMo-LB and MeMoPD-LB significantly outperform the adaptive and the RL baselines on both traffic scenarios. Our results also demonstrate the superiority of the learning-based load balancing compared to the rule-based ones. By interacting with the system, the agent can incorporate knowledge about the network to achieve a better performance. Compared with the RL baseline, using the meta-learning approach enables us to improve the network performance with a great margin. This confirms that the learned meta-policy is a better initialization than the random one [22]. Further, our results highlight the advantage of a multi-objective solution over a single-objective one.

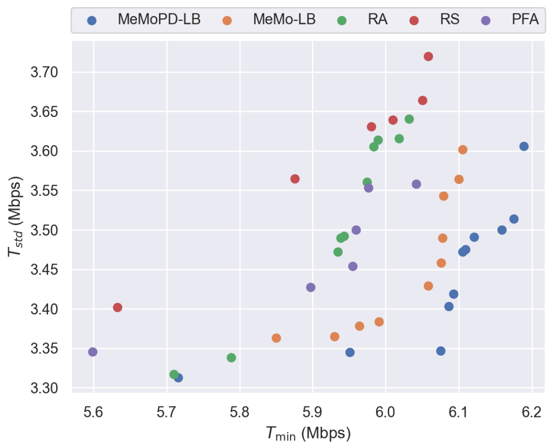
3.6.3 Comparison with Multi-Policy Methods

In this section, we investigate Pareto front approximations computed by our methods and different multiple-policy MORL baselines. To do so, we sample 300 preferences evenly distributed in $[0, 1]$ and finetuned the meta-policy up to 300 gradient steps. Our objective is to assess how well the policies approximate the true Pareto front, and how rapidly these policies are computed. We start by examining the hypervolume indicator to measure the

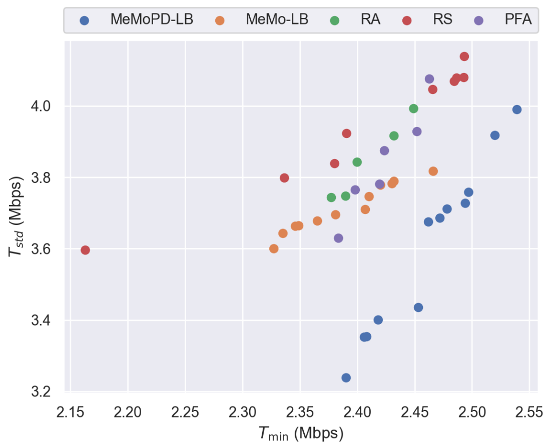
Table 3.4: The hypervolume indicator for our solutions and baseline algorithms on both traffic scenario. The best number is in bold.

Method	MeMoPD-LB	MeMo-LB	RA	RS	PFA
Low	**0.92**	0.82	0.76	0.67	0.75
High	**2.09**	1.67	1.53	1.65	1.63

quality of the computed Pareto fronts. As a reference point, we chose the performance of the No LB baseline because it represents the lowest performance possible. The results in Table 3.4 demonstrate that our methods outperform all baselines on both traffic scenarios. For the high traffic scenario, MeMo-LB achieves a Pareto front that is comparable to other MORL baselines. However, MeMoPD-LB surpasses the baselines and MeMo-LB with an importance margin. This proves again that knowledge transfer between tasks has a significant impact on the improvement of the Pareto solutions.



(a) Low traffic



(b) High traffic

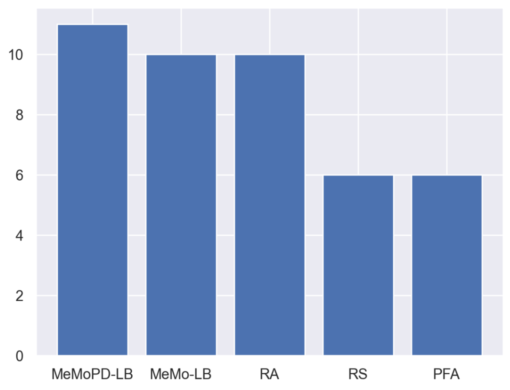
Figure 3.9: Pareto front approximation comparison for both traffic scenarios.

Figures 3.9(a) and 3.9(b) illustrate the Pareto fronts for both traffic scenarios. Recall that the $T_{\min}$ objective is maximized while the T_{std} objective is minimized. Thus, the best solutions are situated in the bottom right corner of the graphs. The RS baseline has the lowest performance compared to the other methods and MeMoPD-LB results in Pareto optimal solutions that strictly dominate all other solutions.

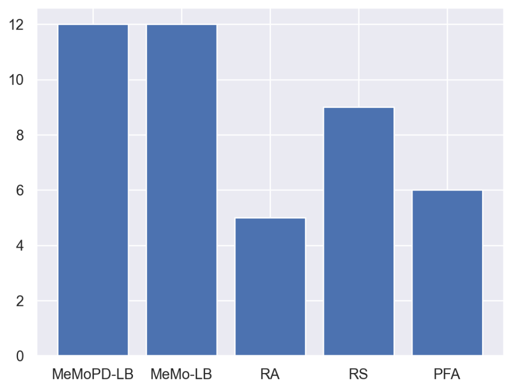
For the low traffic scenario (Fig. 3.9(a)), the RA and the PFA algorithms are able to find solutions that cover different regions of the Pareto front. Although RA and MeMo-LB find the same number of non-dominated solutions (Fig. 3.10(a)), the Pareto solutions computed by MeMo-LB dominate the one found by RA, thereby yielding in a better Pareto front. Also, we see that MeMoPD-LB finds more non-dominated solutions for this traffic scenario, which will produce a denser Pareto front.

Regarding the high traffic scenario, we observe that the performance of MeMo-LB is comparable to the PFA baseline and both these methods compute Pareto solutions that cover only a part of the Pareto front. Hence, MeMo-LB encounters difficulties in more complex tasks which emphasize the importance of our second proposed solution MeMoPD-

LB. Indeed, MeMoPD-LB computes a better quality Pareto front than MeMo-LB with the same number of solutions (see Fig. 3.10(b)). Also, the Pareto solutions of MeMoPD-LB are better distributed across the Pareto front than the other methods.



(a) Low traffic



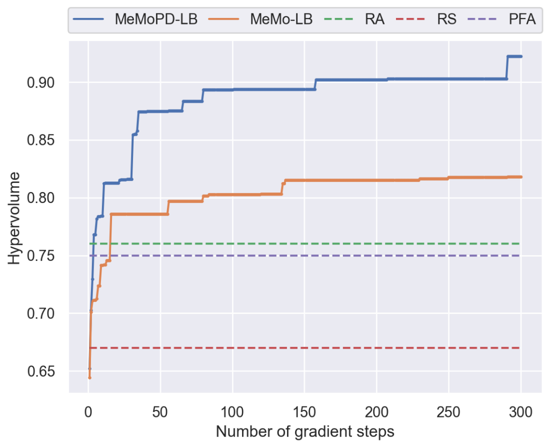
(b) High Traffic

Figure 3.10: The number of non-dominated solutions found by different methods.

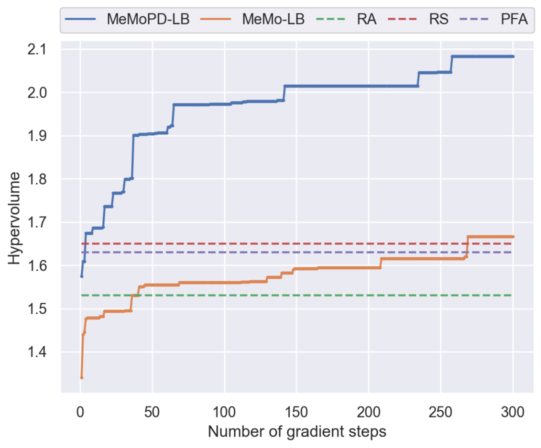
3.6.4 On the Impact of Policy Distillation

In the previous sections, we showcased the advantage of using a distilled policy as a task-specific starting point for the meta-training. In this part, we put more emphasis on this contribution and further highlight its superiority compared to the original meta-training process.

We start by studying the data efficiency of our approach. We plot the progress of the hypervolume indicator as a function of the number of finetuning steps. As observed in Figs. 3.11(a) and 3.11(b), our proposed methods achieve the same performance as the best MORL baseline after a few finetuning steps. It is seen that with policy distillation, the quality of the Pareto front keeps improving with more gradient steps, which is not the case for the MeMo-LB framework. With policy distillation, MeMoPD-LB can achieve better per-



(a) Low traffic



(b) High traffic

Figure 3.11: The improvements of the hypervolume indicator (y-axis) w.r.t the number of finetuning gradient steps (x-axis). The dotted lines denote the final hypervolume of the Pareto front estimated by the MORL baselines.

forming solutions with fewer samples and gradient steps than MeMo-LB. Consequently, our MeMoPD-LB solution is more data efficient than MeMo-LB thanks to knowledge transfer.

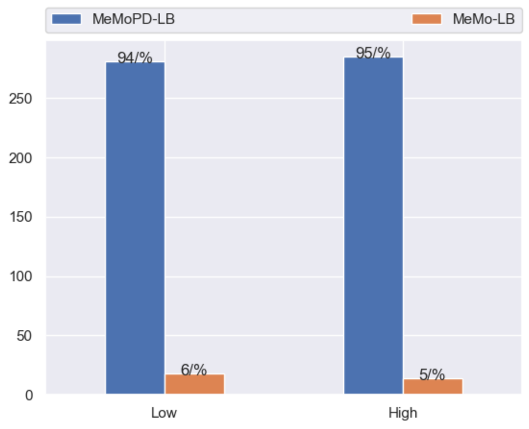


Figure 3.12: The number of policies achieving higher scalarized rewards given a weight vector

Additionally, we investigate the adaptation capabilities of both methods by comparing the performance of 300 individual policies at the end of the finetuning phase. We count the number of policies that outperform the other method in terms of scalarized rewards. As shown in Fig. 3.12, MeMoPD-LB outperforms MeMo-LB by finding better policies with the same number of samples and gradient steps for more than 90% of the preferences.

To summarize, the policy distillation extension has the following advantages compared to the original meta-training process:

- Higher quality Pareto front approximations, hence better improvement on the different objectives;
- Better sample efficiency since better performance is achieved with fewer data and gradient steps;
- Faster adaptation for individual trade-offs.

Chapter 4

Conclusion

In this thesis, we have investigated the application of MORL to learn multi-objective load balancing control policies. We advocated a meta-learning approach where a general parametrized policy is learned that can be adapted to new preferences with fewer samples and gradient steps. The two meta-learning solutions introduced here demonstrated that the meta-learning framework can be applied to complex, high-dimensional, and real-world control problems even with a limited number of samples and tasks. Our experimental results confirmed that the multi-objective approach is more effective to improve the wireless network performance than single-policy approaches. Furthermore, we showcased that policy distillation can help improve the generalization of the meta-policy by providing a task-specific starting point for the meta-training.

Below, we point out several important future directions that can be further investigated to enable the real-world deployment of our proposed solution.

Data Collection: In all our experiments, we used a system-level simulator to mimic real-world traffic scenarios. Although this simulator provides us with data as close to real network data as possible, this comes with a high computational cost. The more complex the traffic scenario is, the slower the trajectory collection becomes. Therefore, it will be valuable to design benchmark environments that are relevant, challenging, and yet computationally feasible. A recent attempt [61] has proposed a differentiable simulator for cellular radio access networks that significantly speeds up the network simulation compared to existing ones.

Overcoming under-fitting: Since our proposed solution is based on meta-learning, it inherits the challenges faced by the base meta-learning algorithm [62]. Namely, the MAML method relies on policy-gradient algorithms known for their sample inefficiency. Consequently, such methods require more samples and environment interactions to achieve their best performance. In our experiments, we limited the number of tasks and sampled trajectories for each task, thereby during the adaptation phase, it may not be possible to achieve good performance on each task. Using distilled policy as an adaptation starting point has significantly improved the general performance of the meta-policy. Other knowledge transfer or continual learning techniques can be explored to speed up the two steps of the meta-training loop and avoid under-fitting.

Robustness and Safety: Since our aim is to build solutions for a real-world problem, a natural continuation of this work is to investigate and mitigate the robustness and safety challenges related to meta-learning and RL in general. Amongst these challenges is the robustness against adversarial attacks. As an example, the authors in [63] developed specific attacks to manipulate the training data used by the meta-learners and showcased a significant drop in performance due to such attacks. Furthermore, meta-learning algorithms are known to generalize well to tasks that are similar to the ones observed in the training. However, real-world applications experience unexpected changes and perturbations in the environment and system dynamics. For instance, wireless networks are characterized by high variability, especially due to the mobility of the end-users. Hence, developing learning algorithms robust to uncertainties, noise as well as distributional shift is still an open research problem and a promising future direction.

Multi-agent load balancing: We proposed solutions for multi-objective load balancing for a single sector. Another promising direction is to extend this work to a multi-agent control problem. For instance, eNBs can cooperate together via communication or other coordination mechanisms to avoid potential problems related to load balancing such as interference.

Appendix A

Experimental Setup Details

A.1 Baselines

This section is dedicated to provide more details for the baseline methods listed in Table 3.2.

A.1.1 Single Objective Baselines

The first baseline is the rule-based algorithm which is currently deployed in the real network. This is why the reported relative improvements are computed w.r.t this baseline. We also consider the adaptive rule-based algorithm which is an improved version of the rule-based baseline. For the RL baseline, we follow the common procedure as in previous work [22, 64].
Rule-Based Load Balancing: It relies on fixed control parameters. These values are set by the operator based on prior knowledge of the network traffic. For the whole episode, the prefixed values do not change.
Adaptive Rule-Based Load Balancing [11]: It adjusts the load balancing parameters based on the load differences between the neighboring cells. We measure the cell load in terms of resource block utilization ratio. For pairwise parameters such as $O_{c,t}$ (see Section 3.2), the

load difference is computed between each cell pair (c, t). For individual cell parameters like T_{A_2}, $T^1_{A_5}$ and $T^2_{A_5}$ (see Section 3.2), the load difference is measured as the difference between the load of cell c and the mean load of its neighboring cells. For simplicity, we refer to the load difference as ρ_{diff} and the re-selection or HO parameters by X. If the load difference ρ_{diff} surpasses a given threshold ρ_{th}, the load balancing parameters are updated by adding or subtracting an adaptive step-size Δ as follows:

$$X^{m+1} = \begin{cases} X^m - \Delta; & \rho_{\text{diff}} > \rho_{\text{th}} \\ X^m + \Delta; & \rho_{\text{diff}} < -\rho_{\text{th}} \\ X^m; & |\rho_{\text{diff}}| \leq \rho_{\text{th}}. \end{cases} \tag{A.1}$$

The adaptive step-size Δ is modeled as a power function of the load difference such that when the difference between cells increases, the step size becomes higher:

$$\Delta = \frac{(\rho_{\text{diff}} - \rho_{\text{th}})^n}{1 - \rho_{\text{th}}} (X_{\text{max}} - X_{\text{ps}}) + X_{\text{ps}}, \tag{A.2}$$

where X_{max} is the maximum value for the parameter X and X_{ps} is the minimal step size. In our experiments, we used $\rho_{\text{th}} = 0.25$, $n = 1$, and $X_{\text{ps}} = 1\,\text{dB}$. The Algorithm 2 summarizes the adaptive rule-based algorithm considered in this work.

Algorithm 2 Adaptive Rule-Based Load Balancing

```
1: Input: $\rho_{\text{th}}$: load difference threshold, $n$: power for the step-size update, $H$: episode length
2: Initialize load balancing parameters: $O_{c,n}$, $T_{A_2}$, $T^1_{A_5}$ and $T^2_{A_5}$
3: for $t = 0, ..., H$ do
4:     Compute the step size $\Delta$ as in Eq. A.2
5:     Update the load balancing parameters using Eq. A.1
6: end for
7: Output: Updated load balancing parameters
```

A.1.2 Multi-Objective Baselines

All the considered MORL baselines follow an evolutionary learning process. It starts with a warm-up stage where N policies are randomly initialized and N evenly distributed non-negative weights. For a specific number of iterations M_w, each policy-weight pair (or task) is optimized via the multi-objective policy gradient (MOPG) algorithm [2] described in Algorithm 3. The obtained policies after the warm-up stage correspond to the first generation of the evolutionary algorithm. Afterward, the evolutionary stage starts. Each generation N policy-weight pairs are selected using a specific task selection procedure. The selected tasks are trained by MOPG for a specific number of iterations to generate new (or offspring) policies that will be added to the policy population. The Pareto set is constructed by storing all non-dominated intermediate policies. The evolutionary stage terminates when the total number of environment steps is achieved. Our implementation is based on the one provided in [2]. The different MORL baselines only differ in the task selection procedure:

- **RA**: in each generation, the policies are replaced by their offspring policies. The sampled weights do not change;
- **RS**: in each generation, new policies and weight vectors are randomly sampled;
- **PFA**: similar to RA, the policy in each generation is replaced by its offspring. However, the weights are changed with a fixed step size to cover all the Pareto front.

Algorithm 3 Evolutionary Multi-Objective RL

```
1: Input: $N$: Number of policies/weights, $M_w$: Number of warm-up iterations, $M_e$: Number of iteration of the evolutionary stage, $N_{max}$: total number of env steps
2: Initialize policy population, the Pareto set $\mathcal{F}$
3: Warm-up Stage
4: Randomly generate $N$ policy-weight pairs $\{(\pi_i, \boldsymbol{\omega}_i)\}_{i=0}^{N}$
5: For each pair, run MOPG for $M_w$ iterations and obtain the updated policy
6: Update the policy population and the Pareto set $\mathcal{F}$
7: Evolutionary Stage
8: while env steps $< N_{max}$ do
9:     Select $N$ policy-weight pairs using a selection function
10:    Run MOPG for $M_e$ iterations and obtain the offspring policies
11:    Update the policy population and the Pareto set $\mathcal{F}$
12: end while
13: Output: Approximated Pareto set $\mathcal{F}$
```

A.2 Training Details

For all learning-based baselines (single and multi-objective methods), the policy is a three-layer neural network. Each hidden layer has 256 units and activated with tanh function. For our meta-RL based methods, we used shared parameters including the

- N: the number of sampled preferences in each meta-iteration;
- K: the number of trajectories sampled for each task;
- N_{meta}: the total number of meta-iterations;
- grad steps: the number of gradient steps in the adaptation phase;
- β: the task adaptation learning rate;
- TRPO: all other hyperparameters for the TRPO algorithm

The optimization of the meta-policy is based on the TRPO algorithm, for which we report all training parameters in Table A.2. The shared hyperparamters of the MAML algorithm are summarized in Table A.1.

For the policy distillation, each teacher policy is trained using PPO. Our implementation is based on the open-source codebase [65] and the training parameters are summarized in Table A.4.

Regarding the multi-objective baselines, they are trained for the same number of total environment steps as our meta-policy. Knowing that each episode has 24 steps, the total environment steps can be computed using the information in Table A.1. Each meta-iteration involves $2 \cdot$ grad steps $\cdot\ (N \cdot K \cdot 24)$ environment steps. The 2 factor comes from the fact that we collected trajectories for both training and validation steps. We used the same PPO hyperparameters as in A.4 for all the MORL baselines except for the number of environments that we restricted to 4 to have the equivalent number of environment steps as for the meta-training. For all the experiments, we set $N = 6$, $M_w = 200$, $M_e = 20$ and $N_{max} = 200000$.

Parameter	Value
N	5
K	10
N_{meta}	500
β	0.1
grad steps	1

Table A.1: MAML hyperparameters

Parameter	Value
KL-divergence limit δ	0.01
Conjugate Gradient iterations	10
Damping coefficient	1e-5
Line search max steps	15
Backtracking coefficient	0.8
Discount factor γ	0.97
GAE λ	0.95

Table A.2: TRPO parameters

Table A.3: Hyperparamters for the training of our proposed solutions

Parameter	Value
Time-steps per actor batch	24
Number of environments	10
Learning rate	0.0003
Discount factor γ	0.97
GAE λ	0.95
Clip range	0.15
Number of epoch	10
Entropy coefficient	0.0
Value loss coefficient	0.5

Table A.4: PPO parameters for policy distillation

Bibliography

[1] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2014.

[2] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 607–10 616.

[3] 3rd Generation Partnership Project Technical Specification Group Radio Access Network, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN) - further advancements for (E-UTRAN): Physical layer aspects," no. 36.814 v9.0.0, 2010.

[4] R. Müller *et al.*, "Ericsson mobility report," *Ericsson, Jun*, 2020.

[5] Y. Zhong, X. Ge, H. H. Yang, T. Han, and Q. Li, "Traffic matching in 5G ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 100–105, 2018.

[6] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the dominant theme for wireless evolution into 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 82–89, 2014.

[7] 3rd Generation Partnership Project Technical Specification Group Radio Access Network, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); self-

configuring and self-optimizing network (SON) use cases and solutions," no. 36.902 v9.3.1, 2011.

[8] R. Kwan, R. Arnott, R. Paterson, R. Trivisonno, and M. Kubota, "On mobility load balancing for LTE systems," in *VTC Fall.* IEEE, 2010, pp. 1–5.

[9] N. Zia and A. Mitschele-Thiel, "Self-organized neighborhood mobility load balancing for LTE networks," in *Wireless Days.* IEEE, 2013, pp. 1–6.

[10] Z. Huang, J. Liu, Q. Shen, J. Wu, and X. Gan, "A threshold-based multi-traffic load balance mechanism in LTE-A networks," in *WCNC.* IEEE, 2015, pp. 1273–1278.

[11] Y. Yang, P. Li, X. Chen, and W. Wang, "A high-efficient algorithm of mobile load balancing in LTE system," in *VTC Fall.* IEEE, 2012, pp. 1–5.

[12] M. M. Hasan, S. Kwon, and J. Na, "Adaptive mobility load balancing algorithm for LTE small-cell networks," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 4, pp. 2205–2217, 2018.

[13] T. Yamamoto and S. Konishi, "Mobility load balancing scheme based on cell reselection in LTE systems," *IEICE Technical Report; IEICE Tech. Rep.*, vol. 112, no. 132, pp. 37–42, 2012.

[14] P. Muñoz, R. Barco, J. M. Ruiz-Avilés, I. De La Bandera, and A. Aguilar, "Fuzzy rule-based reinforcement learning for load balancing techniques in enterprise LTE femtocells," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 1962–1973, 2013.

[15] S. S. Mwanje and A. Mitschele-Thiel, "A Q-learning strategy for LTE mobility load balancing," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013, pp. 2154–2158.

[16] Y. Xu, W. Xu, Z. Wang, J. Lin, and S. Cui, "Load balancing for ultra-dense networks: A deep reinforcement learning based approach," *CoRR*, vol. abs/1906.00767, 2019.

[17] L. Deng, Y. He, Y. Zhang, M. Chen, Z. Li, J. Y. B. Lee, Y. J. Zhang, and L. Song, "Device-to-device load balancing for cellular networks," *IEEE Trans. Commun.*, vol. 67, no. 4, pp. 3040–3054, 2019.

[18] P. H. Barros, I. Cardoso-Pereira, L. Foschini, A. Corradi, and H. S. Ramos, "Load balancing in D2D networks using reinforcement learning," in *ISCC*. IEEE, 2019, pp. 1–6.

[19] S. Khosravi, H. S. Ghadikolaei, and M. Petrova, "Learning-based load balancing handover in mobile millimeter wave networks," in *GLOBECOM*. IEEE, 2020, pp. 1–7.

[20] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*. PMLR, 2016, pp. 1842–1850.

[21] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017, pp. 4077–4087.

[22] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1126–1135.

[23] X. Chen, A. Ghadirzadeh, M. Björkman, and P. Jensfelt, "Meta-learning for multi-objective reinforcement learning," in *IROS*. IEEE, 2019, pp. 977–983.

[24] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, 2009.

[25] P. Muñoz, R. Barco, I. de la Bandera, M. Toril, and S. Luna-Ramirez, "Optimization of a fuzzy logic controller for handover-based load balancing," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011, pp. 1–5.

[26] S. ZahirAzami, G. Yekrangian, and M. Spencer, "Load balancing and call admission control in UMTS-RNC, using fuzzy logic," in *International Conference on Communication Technology Proceedings, 2003. ICCT 2003.*, vol. 2, 2003, pp. 790–793 vol.2.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning - An introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998.

[28] S. S. Mwanje and A. Mitschele-Thiel, "Distributed cooperative Q-learning for mobility-sensitive handover optimization in LTE SON," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, vol. Workshops, 2014, pp. 1–6.

[29] 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.

[30] K. M. Attiah, K. Banawan, A. Gaber, A. Elezabi, K. G. Seddik, Y. Gadallah, and K. Abdullah, "Load balancing in cellular networks: A reinforcement learning approach," in *CCNC*. IEEE, 2020, pp. 1–6.

[31] G. Alsuhli, H. A. Ismail, K. Alansary, M. Rumman, M. Mohamed, and K. G. Seddik, "Deep reinforcement learning-based CIO and energy control for LTE mobility load balancing," in *CCNC*. IEEE, 2021, pp. 1–6.

[32] H. Zhang, L. Song, and Y. J. Zhang, "Load balancing for 5G ultra-dense networks using device-to-device communications," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 6, pp. 4039–4050, 2018.

[33] C. Liu, X. Xu, and D. Hu, “Multiobjective reinforcement learning: A comprehensive overview,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 45, no. 3, pp. 385–398, 2015.

[34] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, “A survey of multi-objective sequential decision-making,” *J. Artif. Intell. Res.*, vol. 48, pp. 67–113, 2013.

[35] K. Miettinen and M. M. Mäkelä, “On scalarizing functions in multiobjective optimization,” *OR Spectr.*, vol. 24, no. 2, pp. 193–213, 2002.

[36] I. Y. Kim and O. De Weck, “Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation,” *Structural and multidisciplinary optimization*, vol. 31, no. 2, pp. 105–116, 2006.

[37] S. Parisi, M. Pirotta, N. Smacchia, L. Bascetta, and M. Restelli, “Policy gradient approaches for multi-objective sequential decision making: A comparison,” in *ADPRL*. IEEE, 2014, pp. 1–8.

[38] K. Li, T. Zhang, and R. Wang, “Deep reinforcement learning for multi-objective optimization,” *CoRR*, vol. abs/1906.02386, 2019.

[39] M. Pirotta, S. Parisi, and M. Restelli, “Multi-objective reinforcement learning with continuous pareto frontier approximation,” in *AAAI*. AAAI Press, 2015, pp. 2928–2934.

[40] A. Feriani and E. Hossain, “Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial,” *IEEE Communications Surveys Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[41] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[42] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[43] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[44] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 1889–1897.

[45] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[46] J. Nocedal and S. Wright, *Numerical optimization.* Springer Science & Business Media, 2006.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[48] V. Pareto, *Manuel d'économie politique.* Giard & Brière, 1909, vol. 38.

[49] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *Australasian joint conference on artificial intelligence.* Springer, 2008, pp. 372–378.

[50] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[51] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on evolutionary computation*, vol. 7, no. 2, pp. 117–132, 2003.

[52] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[53] H. von Stackelberg, S. Von, and A. Peacock, *The Theory of the Market Economy.* Oxford University Press, 1952.

[54] P. Kela, J. Puttonen, N. Kolehmainen, T. Ristaniemi, T. Henttonen, and M. Moisio, "Dynamic packet scheduling performance in UTRA long term evolution downlink," in *2008 3rd International Symposium on Wireless Pervasive Computing.* IEEE, 2008, pp. 308–313.

[55] M. Song, S. Moon, and S. Han, "Self-optimization of handover parameters for dynamic small-cell networks," *Wirel. Commun. Mob. Comput.*, vol. 15, no. 11, pp. 1497–1517, 2015.

[56] D. P. Bertsekas and R. G. Gallager, *Data Networks, Second Edition.* Prentice Hall, 1992.

[57] 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.* IEEE, 2021, pp. 1–6.

[58] T. Deleu, "Model-Agnostic Meta-Learning for Reinforcement Learning in PyTorch," https://github.com/tristandeleu/pytorch-maml-rl, 2018.

[59] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.

[60] E. Parisotto, L. J. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," in *ICLR*, 2016.

[61] D. Rivkin, "System level differentiable simulation of radio access networks," in *Workshop on Differentiable Computer Vision, Graphics, and Physics applied to Machine Learning at NeurIPS*, December 2020.

[62] Y. Duan, "Meta learning for control," Ph.D. dissertation, University of California, Berkeley, USA, 2017. [Online]. Available: http://www.escholarship.org/uc/item/20z5k8dq

[63] H. Xu, Y. Li, X. Liu, H. Liu, and J. Tang, "Yet meta learning can adapt fast, it can also break easily," in *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, C. Demeniconi and I. Davidson, Eds. SIAM, 2021, pp. 540–548. [Online]. Available: https://doi.org/10.1137/1.9781611976700.61

[64] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, "Metalight: Value-based meta-reinforcement learning for traffic signal control," in *AAAI*. AAAI Press, 2020, pp. 1153–1160.

[65] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.