Optimized Data-driven MAC Schedulers for Low-Latency Downlink in LTE Networks

Fabio Pianese Peter J. Danielsen
Wireless Research Program
Bell Labs, Alcatel-Lucent
{first.last}@alcatel-lucent.com

***Abstract*—We consider a novel way by which DL dynamic MAC scheduling can be augmented based on cross-layer inputs to provide desired performance enhancements. We describe ODDS, an automated design process for the data-driven refinement of downlink resource scheduling algorithms. ODDS extracts insights from the behavior of a simulated LTE cellular system under randomized traffic patterns and propagation conditions in a given network scenario. An offline iterated reinforcement learning campaign seeks to best fulfill a target set of goals, e.g. "fair throughput with low latency", which can be encoded in the form of an arbitrary utility function specified by the designer. The knowledge base obtained from the learning campaign consists in a set of rules that a parametric scheduler leverages at run-time. We present our learning framework and evaluation results. ODDS-generated schedulers are shown to achieve improved performance compared to well-known reference scheduling strategies.**

I. Introduction

Channel-aware radio resource scheduling in cellular environments is a subject of long-standing interest in wireless research. Theoretical analysis has provided a number of tractable results, such as those yielded by the Network Utility Maximization (NUM) model [14]. The application of NUM to α-fair utility functions is at the foundation of the proportional fair (PF) resource scheduling discipline. In ideal deployment scenarios, the PF scheduler presents desirable asymptotic properties, such as optimality, stability, and insensitivity [5], at the price of a computational complexity that is linear in the number of User Equipments (UEs) to be scheduled.

Designing schedulers with interesting properties often entails some form of incremental improvement of existing algorithms, based on both theoretical and practical concerns. A new scheduler definition must strike a better trade-off between efficiency, computational complexity, and other application-related concerns such as fairness, proportionality among users, and QoS [8]. With the availability of vast amounts of computing power, it is no longer unthinkable to supplement the process of designing efficient algorithms with insights that can be automatically extracted by a machine learning process.

In the broader networking context, algorithms that perform simulation-driven exploration of problem spaces have been successfully applied to the automated design of complex network protocols, e.g. TCP congestion control [12], and to the run-time modeling of the transport-layer capacity of wireless links [13]. Examples in the cellular scheduling domain are emerging where the results of system modeling and simulation are used at run-time to reconfigure scheduler behavior to match a target fairness goal [9]. An interesting question is then whether similar data-driven learning techniques can be downright applied to the design of MAC scheduling algorithms of practical interest, and what the actual behavior of such algorithms would be in simulation and real-world conditions.

This paper presents a machine learning process based on reinforcement learning that can produce Optimized Data-Driven Scheduling (ODDS) algorithms for LTE MAC downlink. By relying on an approximate model of system behavior and extensive simulation campaigns, we extend the awareness of ODD schedulers to other salient properties of network traffic that cannot be captured easily by an analytic framework of tractable complexity. Approximation is required for it is obviously impractical to capture by enumeration the full extent of possible system configurations and traffic conditions. However, by a proper choice of training scenarios and state variables we can develop algorithms showing a consistently good behavior that exceeds the performance of the baseline PF scheduler in a range of simulated settings. We will mainly focus on:

- Adapting a MAC scheduler metric to support learning;
- Exploring a wide range of random resource allocation scenarios to which a scheduler-in-training is exposed; and
- Validating scheduler performance via simulations.

In the next Section, we will provide some background on LTE MAC layer scheduling. Section III describes the structure of our parametric scheduler and the salient aspects of the learning process that generates optimized scheduler definitions. Section IV follows up with the implementation of the ODDS framework as a distributed system and describes its operation. We will focus at first on building ODD scheduler definitions that strive to reduce the average transmission latency experienced by the whole UE population, presenting the behavior of algorithms that were trained under various scenarios. In Section V we evaluate the trade-offs among throughput, latency, and fairness achieved by examples of ODD schedulers and compare their behavior to well-known reference schedulers. Section VI concludes the paper.

II. Overview: DL MAC Scheduling in LTE

LTE uses Orthogonal Frequency Division Multiplexing (OFDM) to transmit data. The OFDM flavor used in LTE divides the assigned spectrum into a grid of resource blocks given by groups of contiguous subcarriers that are 180kHz wide in frequency domain and time slots that are 0.5ms long.

ISBN 978-3-901882-80-7 © 2016 IFIP

Resource blocks are the base unit of MAC scheduling and are assigned in multiples to system users. The goal of a scheduling algorithm in LTE is to optimally allocate resource blocks (RBs) to each user at every Transmission Time Interval (TTI) of 1ms. Optimization is performed on a set of metrics such as throughput, delay, fairness or spectral efficiency.

Many textbook scheduling algorithms have been considered and evaluated with LTE [1]. Maximum throughput (MT) assigns each RB to the user with the best radio link conditions. This scheduling discipline leads to disappointing outcomes in the case where users experience different channel conditions, since this results in strongly unfair allocation to the benefit of the users with better channel quality. Another algorithm is round robin (RR), in which users take turns in sharing resources. The drawback of RR scheduling is that users who have poor channel conditions will be assigned a fair share of resources which could have benefited more some user with better channel quality. The proportional fair (PF) scheduling algorithm improves the outcome by taking both resource allocation history and channel conditions into account. As discussed in [5], the PF scheduler explores the full set of assignments between a subset of users and available frequencies and maximizes a function of user rates. This heuristic yields an optimal assignment in the case of full buffers and i.i.d. rate distributions. At each scheduling instant and for each RB, the PF scheduler chooses the user who maximizes the ratio of expected transmission rate r_u and the set of assigned frequencies S_u over the average throughput R_u of the user on a previous time window:

$$\text{scheduled user} := \text{argmax}_{u \in users;\, \{S_u\}} \frac{r_u(S_u)}{R_u}$$

Several instances of priority-based schedulers have been built around the ones mentioned above. These often leverage standard facilities provided by LTE that assign quality levels to individual bearers. For instance, Weighted PF (WPF) inserts a term at the numerator of the PF metric that encodes a fixed level of priority of a bearer. Priority to GBR flows can be imparted by using a weight term that increases over time as a function of the difference between the guaranteed and achieved rate. In this paper, we consider only best effort bearers and undifferentiated traffic, which constitute the prevalent type of bearers used for everyday Internet connectivity.

A. Scheduling in Realistic Scenarios

Scheduling quality in the real world is subject to a number of additional influences that are not easily covered in the model outlined above. As an example, we consider two factors that may lead to sub-optimal results: non-ideal traffic patterns and errors in channel quality estimations.

Non-ideal traffic patterns may defeat the optimality of PF scheduling as an underlying condition of PF optimality is that the buffers are constantly full. In reality, data traffic patterns for many popular applications are bursty and may consist of messages whose length is smaller than the size of a transmission opportunity. One example is in the case of automated status notifications, which today represent a sizable fraction of the overall smartphone traffic volume. Whenever a small amount of data is available in a user's transmit buffer, assigning it the best possible resource block may negatively affect the performance of another user that has more buffered data but slightly worse channel quality estimates. At the other end of the scale, when a buffer's content is approaching its maximum size, packet drops between protocol layers might be avoided by prioritizing traffic based on buffer size [3].

Another possible source of inefficiencies is the presence of bursts of erroneous channel estimates, due to fast fading, interference from neighboring cells, and other propagation effects. LTE adopts HARQ mechanisms in order to mitigate the effect of losses, which are sometimes amplified by the aggressive use of adaptive modulation schemes. While uncorrelated among separate OFDM subcarriers, fading losses depend on the speed at which the UE moves and (via the use of adaptive modulation) on its distance from the cell center.

B. Related Work

In the recent literature, enhanced scheduling techniques that leverage cross-layer information have been a subject of great interest [2]. On one hand, theoretical frameworks have been proposed that turn optimization goals (subject to certain properties) into resource allocation algorithms. Song and Li [10] proposed a framework to enable cross-layer optimization based on simple analytic utility functions, which was then applied to guarantee different levels of QoS to multiple classes of applications [11]. On the other hand, many schemes have been investigated that introduce new pieces of information into the scheduling metrics with the goal of influencing the scheduling performance towards desired outcomes. Examples are the use of information about the buffer state, e.g., head-of-line packet size [3] and queuing delay [8], as a part of the scheduling metric. These schemes provide tractable ways to include one or more additional terms into the optimization goal.

It is still an open question whether it is possible to exploit new combinations of sources of cross-layer information to enhance scheduler performance without compromising its analytic properties. In the next section, we propose a practical attempt to answer this question based on an automated approach to scheduling algorithm design.

III. Scheduling as a Machine Learning Problem

Scheduling is a process that matches a finite set of available resources with user demand. In the LTE wireless downlink case, the resources being assigned to multiple users are sets of independent OFDM subcarriers, whose quality estimates may contain errors, e.g., due to fading. The demand, on the other hand, is known: the scheduler can observe the state of the upper protocol layers, where the buffers with data pending transmission are kept. This problem can be represented as a finite-state POMDP [7], where scheduler decisions are a function of the observation at present time of the internal state and of the estimated state of the external process. Finding optimal policies via analysis is an extremely difficult task, given the large dimensionality of the state space. On the other hand, the scientific literature offers many examples of

heuristics that have been used to attack similarly daunting problems in the past, with largely successful results [4].

In this work, we apply data-driven reinforcement learning techniques to determine how to optimize a scheduler to suit scenarios with realistic properties. Given the practical hurdles of collecting actual measurements from a huge number of randomized realizations of LTE cell configurations, we turn to simulation as a tool to approximate reality. Learning is performed on a simulated model of an LTE wireless system. The use of simulations as a source of data points allows to perform a randomized *monte carlo* exploration of the problem: our system observes the trajectories followed by the scheduler's state space over a large number of realizations via the massive use of computational resources, which today are readily available. This extensive training process performed over millions of simulations results in a small base of actions that inform on-line scheduling decisions toward a chosen goal.

A. The Learning Process

An abstract model of the learning process is represented in Figure 1. It is composed by an off-line (light orange) and an on-line (dark blue) part. The scheduler is instrumented to track during its execution a set of *input* variables $\mathcal{I}_u$ for each UE device u. The scheduler behavior and output metric $\mathcal{M}$ is parametrized by a set of *action* variables $\mathcal{A}_u$ that are called upon during the ranking of users by the scheduler metric, leading to an output schedule $S = \mathcal{M}(\mathcal{I}, \mathcal{A})$. An external knowledge base $\mathcal{B} : \mathbb{R}^{|\mathcal{I}|} \Rightarrow \mathbb{R}^{|\mathcal{A}|}$ is accessed on-line to read rules that represent the mapping between intervals of $\mathcal{I}$ and values of $\mathcal{A}$, where $\mathcal{A} = \mathcal{B}(\mathcal{I})$. *Training* is the iterative process by which rules are created and improved in $\mathcal{B}$ so that they collectively maximize the expected value of a target utility function $U(S)$. Training is performed off-line, starting from an initial state in which $\mathcal{B}$ contains a single rule and default action, $\mathcal{A}^*$, that is associated to any value of UE state.

The training process simulates a multitude of procedurally generated scenarios which are meant to sparsely probe the entire space of operating conditions that are liable to be found in the target deployment scenarios. Over multiple *rounds* (Section IV-B) the content of the rule base $\mathcal{B}$ evolves, the number of rules increases, and the values of the actions are modified to lead to increasingly better scheduling outcomes.

1) A Parametric Scheduler Metric: A scheduler metric that supports learning needs to expose a number of actionable parameters that will ultimately affect the value of the utility function. Its outcome must therefore be expressed as a function of $\mathcal{A}$, but may include any number of other variables of arbitrary origin, such as from $\mathcal{I}$, the external environment, and/or the cross-layer protocol state available to the scheduler.

2) Representing the UE's Scheduling State: The choice of an appropriate set of input variables $\mathcal{I}$ to characterize the state of each connected device in the system is crucial to a successful learning process. Since our base heuristic is modeled on a PF metric, we will opt for state variables that capture other cross-layer variables that are available at the scheduler and not represented in the original metric. These variables are designed to track the differences between a theoretical 'full-buffer, perfect channel estimates' model and a practical system where buffers are shaped by traffic activity and transmissions may fail, thus triggering HARQ.

3) Utility Functions for Adaptive Schedulers: In a reinforcement learning system, a utility function U provides the source of feedback by which the progress of the training process is evaluated. Scheduler behavior is characterized by repeatedly computing $U(S)$ on a large number of outcomes of scheduling decisions and tracking its average increase (and preventing its decrease) at each step of the training process. The amount by which the average utility increases across subsequent training steps can be taken as a measure of the progress of the training, thus providing an absolute measure of progress and eventually a stopping criterion.

B. Advantages and Limitations

Replacing the core function of a traditional scheduler, the metric used for ranking, with an automatically-designed heuristic presents several advantages to the system architect: 1) it allows improvements in scheduler quality by increasing the amount of processing devoted to the learning process and the accuracy of the simulation and traffic model; 2) it eases the development of specialized schedulers for targeted operating environments, since the training conditions and goals can be specified by the system designer; and 3) it provides a way to explore complex trade-offs between disparate variables, ranging from the more conventional, such as throughput and latency, to less tractable ones including computational budget and energy footprint terms. The data-driven process enables a designer to experiment with multiple utility functions and scheduler structures to find the best result for her own goals.

The main drawback is the loss of control by the human designer on the finer aspects of the behavior of the scheduler. The limitations imposed by the training process, which hides away most of the logic that underlies the resulting algorithm into unintelligible combinations of non-human-serviceable numbers, render the designer's selection of matching utility function and scheduler metric, together with the definition of an appropriate training environment, a delicate choice that is crucial to the quality of the outcomes.

IV. ODDS - Optimized Data-Driven Schedulers

ODDS is a framework that allows the development of LTE downlink MAC dynamic scheduling algorithms based on a data-driven process. It includes three parts: a learning engine, a simulator, and a validation and visualization suite.

The off-line learning process in ODDS is based on the seminal work in Remy [12], whose code was graciously made available by the authors. We introduced small modifications, in order to account for the different characteristics of the learning process in our case. To faithfully represent the behavior of LTE user equipment at the cell level we leverage the NS3 system simulator, whose LENA module [6] implements the relevant parts of the LTE standard. We modified the LENA implementation of the downlink MAC scheduler to accommodate the tracking of cross-layer state variables, introducing a parametric scheduler metric implementation based on the principles

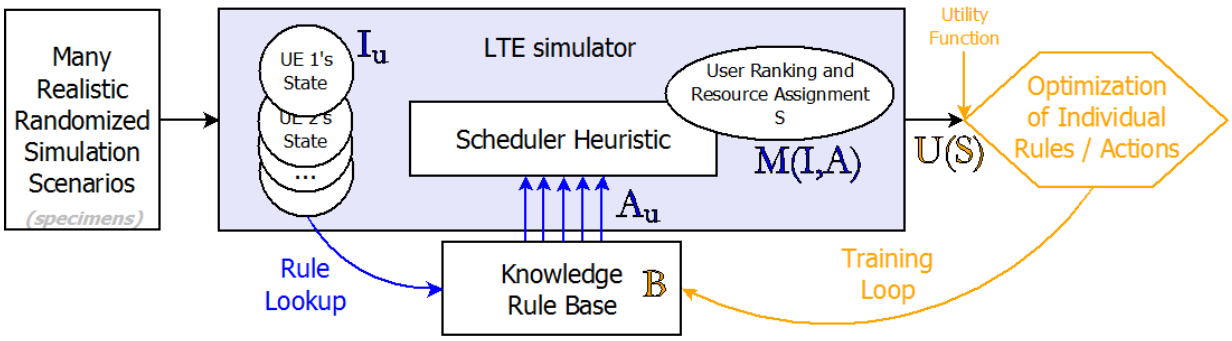


Figure 1: Automated simulation-based training process of the ODDS data-driven scheduling algorithm's rule base

enumerated in Section III. Finally, we designed a system that generates randomized network scenarios and organizes and presents the results. Evaluating an ODD scheduler involves running millions of simulations and collecting performance statistics: automating the entire process made it easy to closely track the behavior of the learning system and provided constant feedback about our design decisions.

A. Designing Schedulers, the ODDS Way

As explained in Section III, using ODDS to develop a scheduling algorithm requires different types of inputs by the designer. The first set of inputs are decisions about how the scheduler should operate and what its optimization objective will be: this amounts to a choice of scheduler metric with a set of available scheduler actions, a set of state variables that the learning process will observe, and a utility metric to guide the advancement of the learning process. These choices obviously need to be informed by a certain amount of domain knowledge about the LTE system, including its protocol stack layout and physical layer mechanisms. The key of a successful design lies in the way the individual mechanisms interact when exposed to the information coming from the simulated system.

1) Scheduler Metric: The ODDS training framework uses as its metric a simple parametric function based on the well-known PF metric. It inherits a) the use of channel estimates, made available from the PHY layer, b) the accounting of past bearer throughput averaged over a certain time window, and c) the existing assignment of modulation and coding schemes (MCS) that translate the measured channel quality into an estimate of available bandwidth. For each RB f, a ranking is established among UEs based on the metric:

$$\mathcal{M}(\mathcal{A})_{u,f} = \alpha_u + \beta_u \frac{\sigma_{u,f}}{R_u}$$

(including each UE u's estimated achievable rate $\sigma_{u,f}$). Values of α_u and β_u introduce a linear bias to the original PF metric, which corresponds to the initial action $\mathcal{A}^* = (\alpha = 0, \beta = 1)$. As usual, RBs are assigned until the current TTI's demand is satisfied or the available RBs are exhausted. The use of alternative ranking functions as metrics for the training of data-driven schedulers is an interesting possibility that we will explore in future work.

2) Observed state variables: The three input variables $\mathcal{I} = \{a, b, c\}$ we consider in ODDS are the following:

a An exponentially weighted moving average (EWMA) of the size in bytes of the data in the PDCP transmission buffer waiting to be sent

b A time-discounted EWMA-based estimate of the frequency of HARQ events that accounts for the bursty nature of fast-fading in time domain

c The time elapsed (in ms) since a UE was last successfully scheduled (excluding HARQ)

The instantaneous scheduling state of a UE at all times can then be represented as a point in 3d-space. Valid ranges for $\mathcal{I}$ parameters generally depend on the details of the scheduler implementation (buffer sizes) and on the simulated scenarios (maximum latency buildup given the traffic load). We expect that, in order for the contents of a rule base to be effective across different execution contexts, the model employed during training must be roughly consistent with the target deployment environment. This ensures that the support over which state parameters vary and their approximate distribution in the state space will be similar.

3) Utility Function: Our present choice for U is the function $log(\text{throughput}) - k \cdot \log(\text{delay})$ that enables a trade-off between PF utility, represented by the first term, and average packet delay, which is weighted by a constant k. In this paper, we will concentrate on schedulers that aim to optimize the latency vs. throughput trade-off. The utility score of a scheduling outcome S is given by the summation of the individual results in terms of average throughput and packet delay over all the participating UEs:

$$U(S) = \sum_{\text{UE}_i \in S} \log(\text{throughput}_i) - k \cdot \log(\text{delay}_i)$$

The evaluation (Section V) included in this paper shows the results we obtained for $k = 0.5$. In our future work, we plan to investigate the effects of other values of the k parameter and utility metrics involving other factors, such as application-relevant deadline violations, that can be used to influence the learning process.

B. Training a Latency-aware ODD Scheduler

The goal of the ODDS offline training process is generating a rule database $\mathcal{B}(\mathcal{I}) = \mathcal{A}$ that provides a mapping between

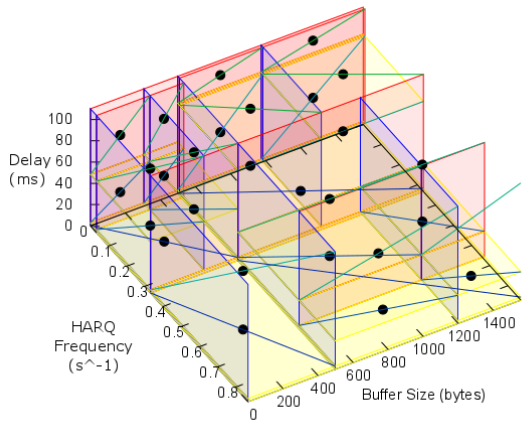


Figure 2: A rule base $\mathcal{B}$, or WhiskerTree, after several rounds of training. Partitions (rules) are highlighted by black dots.

Parameter	Description
PHY	Carrier Frequency: 2GHz DL Bandwidth=5MHz
RLC ARQ	Disabled (UM), RLC buffer is drop-tail
LTE	LTE Release 8 (as in LENA, NS3 3.20)
Cell	single cell, radius=2km Friis model Rayleigh fading
UEs	#UE$\in [10, 70]$ speed$\in [3, 60]$ Km/h (incr. 3 Km/h)
Traffic	best effort UDP; ON-OFF (100 ms); parametric

Table I: Specification: main ODDS simulation settings

input and action variables. By construction the process aims to optimize the utility of the scheduler behavior that $\mathcal{B}$ determines. This happens in two ways:

- Action parameters $\mathcal{A}$ are optimized during the offline learning process for specific ranges of state variables;
- At the same time, the layout of the state space $\mathcal{B}(\mathcal{I})$ is developed iteratively so that rules are created that provide the largest potential for action optimization.

The magnitude of action parameters, together with the value of the other terms in the ODDS metric, ultimately determines the ranking of UEs in the online scheduling of each RB.

In our implementation, the rule database partitions the 3D space of input variables $\mathcal{I}$ and associates to each region two action parameters for $\mathcal{A}$. The reinforcement learning engine we adopted [12] refers to the rule database as a *WhiskerTree* and each partition as a *Whisker*. The *breeding* process begins with a one-rule WhiskerTree and proceeds by identical rounds, in which new rules are alternatively created and optimized. The result can be visualized as in Figure 2.

1) Exploring the State Space: Specimen Generation: We develop a parametric *specification* in order to drive the exploration of the state space. A specification contains the ranges of parameters needed to generate *specimens*, i.e. the detailed environments under which simulations will be performed.

A *specimen* encompasses a channel model (fixed, details in Table I), a mobility model, and a traffic model. The mobility model contains a single cell and generates a UE population of variable size, distributed uniformly randomly over a 2 km radius circle around the eNB. Users all move following a random-waypoint model at a configurable speed

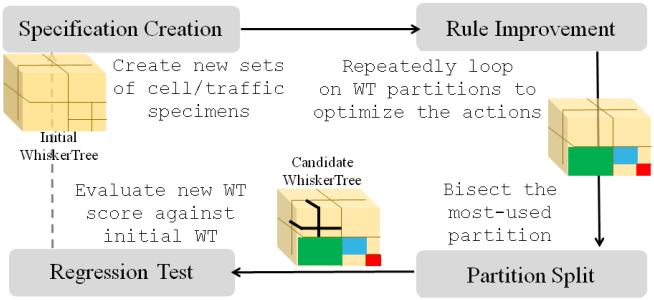


Figure 3: A round of the training process adapted from Remy

which is matched by an appropriate Rayleigh fading pattern. Simulations are generally of short duration (few seconds) to contain the computational requirements of our framework.

The traffic model's goal is to create all sort of contention among the participating UEs on a small timescale, exposing the scheduler to a random variety of packet buffering and delay conditions. A Cell Load parameter sets the total rate of traffic generation in a specimen. Traffic is generated in a non-elastic fashion: the use of RLC in unacknowledged mode (UM) introduces packet drops between PDCP and RLC layers when channel conditions do not support the demanded rate. Other parameters made available by the traffic model control: the time-domain behavior of UEs, which is set to be periodic, by defining the duty cycle of the downlink traffic generation; the allocation of traffic across users, which can be made unequal at will; and finally, the packet size.

Specimens are used as a frame of reference, ensuring that simulations to evaluate the behavior of a scheduler algorithm will take place under the exact same conditions. This is the case in the validation step, where the macroscopic behavior and performance of schedulers needs to be evaluated, but also during several phases of the training process.

2) Training Rounds and Rule Base Improvements: Training develops a WhiskerTree with the goal of maximizing its utility function score. This process is based on a series of identical *rounds*, such as the one represented in Figure 3. Each round yields a new WhiskerTree, which may or not be different compared to the outcome of the previous round. A *family* is the set of WhiskerTrees all generated by the same training process in subsequent rounds.

Each round starts by improving all the rules in $\mathcal{B}$ for a fixed number of *generations*. During a generation, each rule is examined once and a range of alternative $\mathcal{A}'$ values are considered. Evaluating the modified WhiskerTree produces a score and contributes to identify the partitions of $\mathcal{I}$ that are most popular and thus likely to provide some optimization potential. These evaluations are carried out in a steady set of network conditions, provided by a set of random *specimens*. At the end of a rule evaluation, the alternative $\bar{\mathcal{A}}'$ with the highest score that exceeds the original score (if any) becomes the partition's new $\mathcal{A}$. When that happens, a new set of alternatives $\mathcal{A}'$ is created from it, and in turn evaluated. This cycle ends when no alternative $\mathcal{A}'$ has a higher score than $\mathcal{A}$.

After all generations have completed, the focus of training

switches to improving the structure of the freshly modified $\mathcal{B}'$ rule base. A new evaluation identifies the partition of the WhiskerTree that is most often accessed. This partition is then split along all of its dimensions (i.e., 2^3 children) at the median of the state values that were looked up. Each new rule inherits the $\mathcal{A}$ of its parent. The final step of a training round is a regression check of the resulting WhiskerTree $\mathcal{B}'$ against its predecessor in the family. If the test passes, the round yields as its result $\mathcal{B}'$, else the previous WhiskerTree $\mathcal{B}$ is returned.

C. Distributing the Computation

A training session requires millions of simulations, most of which may be run in parallel. Our training framework distributes evaluations over a pool of servers. The training supervisor may add and remove servers during training. The software automatically removes servers that encounter networking issues. A recent server pool contained more than 200 cores from about 20 machines, the pool's size being limited only by the amount of hardware we were able to procure.

V. Performance Evaluation of ODDS

This section evaluates the performance of a single set of ODD scheduler families developed following the guidelines presented in the previous Section. Given the focus of this paper on a latency-aware utility function, our analysis will be considering the following metrics:

a) Latency and Throughput: for all simulations, we characterize the outcome of individual UEs in terms of the overall amount of data they were able to correctly receive over the duration of the experiment (average throughput), and the average latency with which they received individual data packets. When we need to aggregate the results of multiple simulations, e.g., when evaluating a batch of specimens derived from a same specification, we use as metrics the total throughput across all UEs and the average latency computed on all the delivered packets. This choice of aggregate metrics renders the results robust against variations in the number of UEs across randomized specimens.

b) Utility / Fairness: we are interested in characterizing the behavior of ODD schedulers in terms of the utility function U they were built to maximize. However, when applied to any scheduler, U also provides a metric for *combined fairness*[1] in both, throughput and delay. We calculate utility on a per-UE basis, resorting to a conventional choice of finite negative values to deal with the case of UE throughput equal to zero, which may emerge in certain simulation scenarios. A simulation outcome's utility is given by the sum of the utility of individual UEs. To aggregate utility across multiple simulations, we take the arithmetic mean of all outcome utilities.

A. Comparison: ODDS families

The framework's validation component compares the performance of ODD schedulers against NS3's implementations of PF, frequency domain maximum throughput (FdMt), and round-robin (RR) schedulers in a common set of randomized scenarios. In order to exploit parallelism, our framework distributes problem instances containing {specification, specimen, scheduler, WhiskerTree, utility function} across a pool of servers, storing the simulation output for further analysis.

vs. PF	L scenario	M scenario	H scenario
ODD(L)	$\lambda\uparrow\ \tau\sim\ \phi\downarrow$	$\lambda\Downarrow\ \tau\downarrow\ \phi\sim$	-
ODD(M)	$\lambda\Uparrow\ \tau\Uparrow\ \phi\sim$	$\lambda\Uparrow\ \tau\sim\ \phi\sim$	$\lambda\uparrow\ \tau\downarrow\ \phi\sim$
ODD(H)	$\lambda\uparrow\ \tau\Uparrow\ \phi\sim$	$\lambda\uparrow\ \tau\downarrow\ \phi\sim$	$\lambda\uparrow\ \tau\downarrow\ \phi\sim$

vs. FdMt	L scenario	M scenario	H scenario
ODD(L)	$\lambda\Downarrow\ \tau\downarrow\ \phi\Uparrow$	$\lambda\Downarrow\ \tau\Downarrow\ \phi\uparrow$	-
ODD(M)	$\lambda\Downarrow\ \tau\sim\ \phi\Uparrow$	$\lambda\Downarrow\ \tau\sim\ \phi\Uparrow$	$\lambda\Downarrow\ \tau\downarrow\ \phi\Uparrow$
ODD(H)	$\lambda\Downarrow\ \tau\sim\ \phi\Uparrow$	$\lambda\Downarrow\ \tau\downarrow\ \phi\Uparrow$	$\lambda\downarrow\ \tau\downarrow\ \phi\Uparrow$

Table II: Overall performance of ODD scheduler families compared to PF and FdMt (λ=latency, τ=throughput, ϕ=fairness; $\Uparrow$ much better, $\uparrow$ better, $\sim$ same, $\downarrow$ worse, $\Downarrow$much worse)

1) Scenario Definition: The results we present are obtained using the same parametric traffic generation model we introduced in the training phase. The baseline settings for all the specifications are detailed in Table I. We define three scenarios based on the transmission power[2] of the eNB:

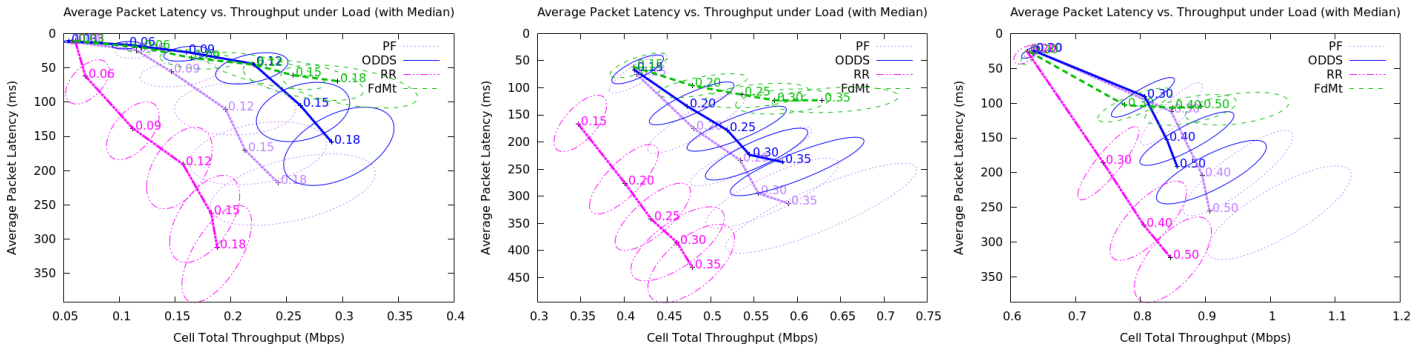
Low Power Scenario (L) TxPower set to 3 dBm
Medium Power Scenario (M) TxPower set to 9 dBm
High Power Scenario (H) TxPower set to 15 dBm

2) Qualitative Assessment : We generate three ODDS families using the scenarios above: ODD(L), ODD(M), and ODD(H). To evaluate their performance, we write specifications that differ by Cell Load for the three (L, M, H) scenarios. Each specification is used to generate a set of random specimens of cellular networks in which scheduler algorithms will be evaluated.

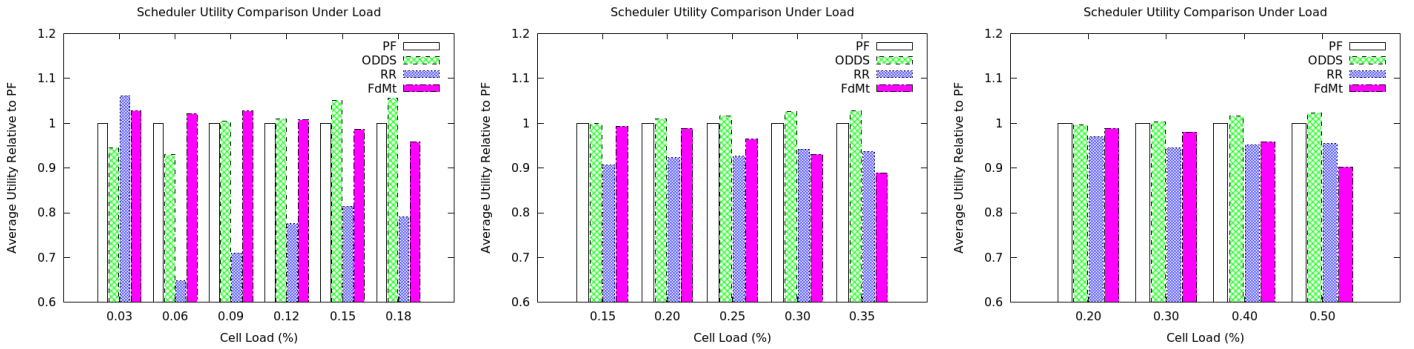
In Table II we capture at a glance the qualitative performance of these ODD schedulers after thirty rounds of improvement, compared to PF (top) and FdMt (bottom) at a high level of Cell Load (as defined for each scenario). We see clear evidence that training in low-power circumstances is ineffective, and that the quality of the outcomes is consistently bad. In the second and third family, we observe on the other hand a large improvement in the outcomes: while maintaining a level of fairness close to PF and definitely higher than FdMt, ODD(M) obtains much lower latency than PF. Interestingly, the latency edge that ODD(H) has on PF is smaller than ODD(M)'s; however, both schedulers' outcomes are similar in the H scenario latency-wise. Throughput of ODD(M) and ODD(H) is slightly lower than both PF and FdMt only in the H scenario. It is interesting to note that ODD(M)'s performance remains respectable even outside of the scenario used in its training, both toward the lower and higher power levels.

[1]Our chosen U derives from the PF definition of utility target, namely the sum of logarithmic UE throughputs which PF is proved to maximize, to which a second term logarithmic in latency is subtracted. Physically, it extends proportional fairness to include the average packet latency observed by UEs.

[2]The L scenario was chosen based on the observation of a UE CQI$\simeq$1 at the edge of the cell.The simulation scenarios we use are power-constrained but do not yet include interference effects. The objective of this early evaluation is to determine whether training can effectively inform scheduler outcomes by highlighting correlations between fading losses and node buffer status. We expect that interference will induce a comparable effect on the internal scheduler state and could also be exploited by the learning process (e.g., via correlations based on the position and speed of a UE inside the cell).



(a) Throughput vs. Latency trade-offs in the L, M, and H scenarios under increasing Cell Load (100 specimens/point)



(b) Average utility value achieved in the L, M, and H scenarios under increasing Cell Load (100 specimens/bar)

Figure 4: Comparison of behavior of ODD(M) against reference schedulers in Low, Medium, and High TxPower scenarios

We conclude from this wide-ranging overview that algorithm design by the ODDS framework is effective at meeting its goal as encoded in the chosen 'fair throughput, low-latency' utility function. When trained in sufficiently good signal power conditions, ODD schedulers demonstrate a behavior which is intermediate between PF and FdMt. We will from now on concentrate our attention on ODD(M) and present in finer detail the results of its comparison against reference schedulers across the three scenarios and at variable levels of load.

B. ODDS Behavior under Load

Figure 4(a) compares the bandwidth and latency behavior of the four scheduler algorithms when executed in one hundred randomly-generated specimens for each scenario and each Cell Load increment. Results are represented on the throughput/latency (TL) plane, with the relative load setting overlaid. In order to concisely represent and compare the outcomes of many simulations, we draw a 1-σ ellipse around the coordinates of the mean result value and provide an overlaid line marking the trend of median result values. We can thus glean a quick understanding of the underlying distribution of outcomes both directly, by observing the relative position between an ellipse and its relative median, and indirectly, by comparing the shape of the ellipses for different schedulers when executed in the same conditions.

Figure 4(b) provides a comparison of the composite fairness of the four schedulers for each scenario and load increment. We can notice by looking at Figure 4(a) that the load trajectory of the ODD scheduler on the TL plane lies between FdMt (above) and PF (below). FdMt's latency flattens out with load while Figure 4(b) reports increasing unfairness. PF's latency degrades with a steeper slope while preserving fairness. ODD shows a sustained advantage in latency over PF, which is maintained as cell load keeps increasing. Furthermore, ODD(M)'s inferior composite fairness observed in the L scenario at low traffic loads can be explained with the impact of small changes (few ms) in the numerator of a ratio with a small denominator. It is interesting to observe the faster growth of the area of the PF 1-σ ellipse in (H), whose center also slowly diverges from the median. This corresponds to a more sparse point distribution on both dimensions compared to FdMt and ODD.

Quantitatively, we see that the latency obtained under higher level of loads by the ODD(M) scheduler is better than PF's latency both on average and median results, by a margin that reaches about 100 ms for the median (and is larger for the average). A study of UE behavior achieved by the scheduling policies at increasing distance from the cell's center, omitted because of space constraints, highlights how the results achieved by edge UEs with ODD(M) are roughly similar to the ones achieved by PF. The latency advantage brought by ODD(M) applies mostly to the UEs located in the cell core.

C. Effect of Learning on Scheduler Performance

Finally, we investigate the effects of the learning process over several milestones in the training of a same scheduler family. Figure 5 represents the behavior of the ODD(M) scheduler observed at three different milestones in its growth: the first was extracted after ten rounds of improvement, with

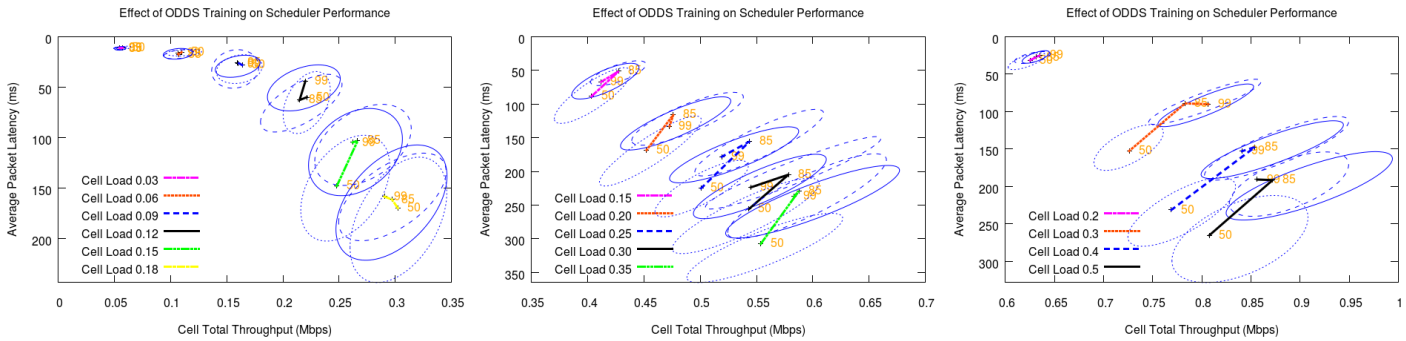


Figure 5: The ODD(M) family is evaluated for $|\mathcal{B}|$ =50, 85, and 99 rules (L, M, and H scenarios; 30 specimens/point)

$|\mathcal{B}| = 50$, the second after twenty rounds, with $|\mathcal{B}| = 85$, and the third and final milestone after thirty rounds, with $|\mathcal{B}| = 99$. We can notice, besides the decrease of the average growth speed of the rule base over subsequent training iterations, that there is a marked improvement in performance between the first two milestones in all three scenarios. The transition between the second and the third milestone has a mixed impact, with benefits for some data points and drawbacks for others. It should be kept in mind that the network specimens were randomly selected across the milestones: the variability observed is partly to be attributed to this fact. We conclude that learning has a continued beneficial effect on average scheduler performance and that effects of over-training are not yet impacting our results after thirty rounds of learning.

VI. Conclusions

In this paper we presented ODDS, a framework based on reinforcement learning for the development of LTE MAC downlink scheduler algorithms with a user-specified optimization goal. We described the use of data-driven techniques applied to scheduler algorithm design, first in abstract terms, then based on a set of design decisions aiming for a fair and low-latency scheduling outcome. We evaluated the behavior of a first batch of ODD schedulers against reference schemes such as PF and FdMt. While highly sensitive to the circumstances of the learning process, we showed that data-driven scheduling algorithms can meet their optimization goals and are thus a viable alternative to schedulers based on analytic considerations. Moreover, the good properties of ODD schedulers extend across diverse settings and scenarios and seem to hold beyond the range of training conditions.

The evaluation presented in this paper is just an early assessment of the possibilities of the ODDS framework, and its scope and development is therefore limited. More work is required to extend the underlying system model, by adding inter-cell interference and introducing more realistic traffic models. From a practical perspective, it is important to establish whether the algorithms generated by the ODDS process behave consistently also outside of simulation, i.e., as part of a real-world LTE eNB scheduler implementation. Finally, we aim to extend our scheme toward application awareness by introducing policies that recognize application types from their observed traffic behavior and consequently adapt the scheduler response based on application-specific utility functions.

Acknowledgments

The authors would like to thank former colleagues Knarig Arabshian and Bob Arlein for their participation in the initial phases of this work, Dragan Samardzjia and Jürgen Otterbach for helpful feedback, along with the many Bell Labs colleagues who supported us by contributing their spare hardware to speed up the distributed execution of simulations.

References

[1] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda. Downlink packet scheduling in lte cellular networks: Key design issues and a survey. *IEEE Comm. Surveys and Tutorials*, 15(2):678–700, 2013.

[2] F. Foukalas, V. Gazis, and N. Alonistioti. Cross-layer design proposals for wireless mobile networks: A survey and taxonomy. *Communications Surveys Tutorials, IEEE*, 10(1):70–85, 2008.

[3] Jinri Huang and Zhisheng Niu. Buffer-aware and traffic-dependent packet scheduling in wireless OFDM networks. In *IEEE WCNC 2007*, pages 1554–1558, March 2007.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *J. of Artificial Intelligence Research*, 4:237–285, 1996.

[5] L. Massoulié. Structural properties of proportional fairness: stability and insensitivity. *The Annals of Applied Probability*, pages 809–839, 2007.

[6] M. Mezzavilla, M. Miozzo, M. Rossi, N. Baldo, and M. Zorzi. A lightweight and accurate link abstraction model for system-level simulation of LTE networks in ns-3. In *ACM MSWIM 2012*, Oct 2012.

[7] G. E. Monahan. A survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, Jan 1982.

[8] G. Piro, L.A. Grieco, G. Boggia, R. Fortuna, and P. Camarda. Two-level downlink scheduling for real-time multimedia services in lte networks. *Multimedia, IEEE Transactions on*, 13(5):1052–1065, Oct 2011.

[9] A. Bin Sediq, R. Gohary, and H. Yanikomeroglu. Optimal tradeoff between efficiency and Jain's fairness index in resource allocation. In *23rd IEEE PIMRC*, 2012.

[10] G. Song and Ye Li. Cross-layer optimization for ofdm wireless networks-part i and ii. *Wireless Communications, IEEE Transactions on*, 4(2):614–634, March 2005.

[11] Guocong Song and Ye Li. Utility-based resource allocation and scheduling in ofdm-based wireless broadband networks. *Communications Magazine, IEEE*, 43(12):127–134, 2005.

[12] K. Winstein and H. Balakrishnan. TCP ex machina: computer-generated congestion control. In *ACM SIGCOMM 2013*, pages 123–134, 2013.

[13] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Proc. of 10th USENIX NSDI*, pages 459–471, 2013.

[14] Yung Yi and Mung Chiang. Stochastic network utility maximisation. *European Trans. on Telecommunications*, 19(4):421–442, 2008.