System Level Differentiable Simulation of Radio Access Networks

Dmitriy Rivkin
Samsung Electronics
Montreal, QC, Canada
d.rivkin@samsung.com

Abstract

We present a differentiable simulator of telecommunications processes, specifically cellular radio access networks (RAN). The simulator is differentiable with respect to the state of individual phones and other key variables which are of interest in a variety of network optimization tasks. This is complicated by the fact that some of these quantities are discrete variables, a challenge which we overcome using a relaxation technique. As a result, we can start to consider a range of classically difficult network configuration tasks using modern learning and optimization methods.

1 Introduction

This paper considers the configuration and tuning of telecommunication networks using methods from differentiable simulation. Such optimization is a demanding problem that is often achieved by an ensemble of heuristic and/or manually-intensive approaches. This problem is increasingly important as the demand for mobile data and the complexity of the attendant networks is steadily increasing [1]. In tuning these complex systems, network operators must solve a wide variety of optimization tasks. Some of the most challenging of these tasks are system-level optimization problems, where large sections of the network (hundreds of cells, thousands of users) must be optimized jointly. Often-studied system-level optimization problems include those of cell (tower) placement, user-to-cell assignment, and power optimization by turning off under-utilized cells. Algorithms for solving these problems are manifold and evaluated in simulation prior to real-world deployment. Even once an algorithm is deployed in the fiel a simulator is still important for diagnosis, tuning, and control (e.g. model predictive control).

Differentiable simulation has the potential to be extremely valuable for two key reasons. First, differentiability of the simulation parameters can help fit the simulator to real data, increasing the fidelity of the model and increasing the efficacy of MPC-based techniques. Second, if gradients are available for the quantities being optimized, it can significantly increase the efficiency of any optimization algorithms. In this paper, we focus primarily on the second goal of providing gradients for quantities of interest to optimization. However, our simulator is designed to work with empirically measured signal-strength maps, which should make it well-adapted to leverage available data.

In this paper, we first present the design of a differentiable system level network simulator implemented in PyTorch, then demonstrate how it can be used to tackle some key optimization problems. To our knowledge, this is the first implementation of a differentiable system-level RAN simulator.

2 Background

While differentiable simulators have rapidly become important tools is some parts of machine learning, they are seldom employed in the telecommunications literature. Several types of simulation are

Workshop on Differentiable Vision, Graphics, and Physics in Machine Learning at NeurIPS 2020.

typically used in the cellular communications literature, and can be roughly classified according to the degree of granularity with which they execute the simulation [2, 3]. A prototypical example is “NS3”, a discrete event simulator that simulates network behaviour at separate uniform discrete time steps. While “NS3” can faithfully model many aspects of a network’s behaviour, it depends critically on the step size of the discretization and can be very computationally demanding when modeling a complex network, making several kinds of optimization so demanding that they are infeasible [4, 5, 6].

In the particular case of cellular RANs (such as LTE, or 5G) addressed in this paper, the key issue we address is the interaction between the radios on individuals phones and the network as a whole. A RAN is a collection of radio-frequency antennas connected to the internet with a fiber-optic backhaul [7]. These antennas communicate with user phones (known as “user equipment” or “UEs”) over the radio spectrum in order to deliver data services to end users. Each individual antenna is referred to as a “cell", and each cell has a limited bandwidth and range. If a UE has a poor connection quality (i.e. low signal to interference plus noise ratio, SINR) to the cell, it will use up more bandwidth to transmit the same amount of data.

3 System model

The performance metric we aim to simulate is packet loss. In order to compute this, we need to know where the UEs are and how much data each one is trying to receive, which cells they are connected to, the SINR of the connection between each UE and its serving cell, and the bandwidth of the cells. Let us discuss each of these in turn. First, we discretize the world into a square grid with m rows and n columns. This is motivated by the fact that SINR is primarily a function of the position of the UE relative to the cell. Thus, for each cell, we can build an SINR map that describes the quality of the connection between a cell and UE in the area. In fact, UEs measure SINR to nearby cells as a normal part of network operation, so we can reasonably expect operators to have detailed SINR maps on hand. UE position is also modelled on the discretized grid. Since each UE can only exist in one place at a time, the position of the UE on the grid is a discrete variable. Similarly, each UE can only be assigned to a single cell, meaning that ue-cell assignment is also a discrete variable. However, we are still interested in obtaining gradients for these variables. Therefore, we relax these “locality constraints" on position and assignment in the simulator, allowing each UE to be assigned to multiple cells at once, and be distributed throughout multiple places on the map. We will show later how we can re-introduce these constraints, and use the gradients from the relaxed formulation to compute gradients for the discrete variables. In fact, with this relaxation we combine all information about UE position, demand, and assignment into a single four-dimensional vector D of shape $(m, n, n_{ues}, n_{cells})$ where each element describes how much traffic each UE is trying to receive through each cell at each point in space.

In order to compute packet loss, we allocate cell bandwidth to each UE in proportion to the demand of the UE. For example, if two UEs, with demand 1 and 2 respectively, are connected to a cell with capacity 9, then the first UE would be allocated a bandwidth of 3 and the second a bandwidth of 6. More concretely, we first compute the total load demand assigned on each cell, D_j:

$$D_j = \sum_{x,y,i} D_{x,y,i,j}, \tag{1}$$

where x and y are the row and column indices of the map, i is the UE index, and j is the cell index. We then allocate bandwidth, B:

$$B_{x,y,i,j} = D_{x,y,i,j} \frac{B_j}{D_j}, \tag{2}$$

where B_j is the bandwidth available to each cell. We then compute the amount of data that can be transmitted through a certain bandwidth using the Shannon-Hartley theorem:

$$C_{SH} = Blog_2(1 + SINR), \tag{3}$$

where C_{SH} is the theoretical maximum capacity for data transmission, B is the bandwidth allocated to the UE. According to equation 3, a link with very high SINR could transmit a lot of data even if the

bandwidth is low. In practical systems, however, there is a limit on how much data can be transmitted even with excellent SINR. Thus, we cap the maximum capacity at B:

$$C = max(C_{SH}, B) \tag{4}$$

where C is the capacity used in simulation. System level packet loss (PL) is then computed as:

$$PL = \frac{\sum_{x,y,i,j} min(B_{x,y,i,j} - C_{x,y,i,j}, 0)}{\sum_{x,y,i,j} D_{x,y,i,j}} \tag{5}$$

The computation outlined above, when implemented in an automatic differentiation framework, can provide gradients of PL with respect to each element of D and B_j.

3.1 Locality constraints

In order to re-introduce locality constraints (i.e. that each UE can only be in a single place and connected to a single cell) we augment the automatic differentiation framework with a new differentiable function, equipped with both forward and backward (i.e. back-propagating) methods. In the forward method, this function accepts UE positions (P_i), demands (D_i), and cell assignments (A_{ij}) and returns $D_{x,y,i,j}$. The positions are snapped to map grid, with x_i and y_i representing the indices of grid square to which the UE position has been snapped.

$$a_i = \arg\max_j A_{ij},\ D_{x_i,y_i,i,a_i} = D_i. \tag{6}$$

Note that since we apply the argmax operation to A_{ij} to compute the assignment, the constraint that each UE can only be assigned to a single cell is enforced. The assignment is expressed as a n_{ue} x n_{cell} matrix to conform to the shape of the gradient, which is defined between each UE-cell pair. Since equation 6 uses argmax and indexing operations, the gradients of which have no standard definition, our we must also define a custom backward method. This method accepts the gradients $\nabla_{D_{x,y,i,j}}$ and returns ∇_{P_i}, $\nabla_{A_{ij}}$, and ∇_{D_i}. Please note that all gradients are of packet loss, but this is omitted from the notation to increase legibility.

$$\nabla_{P_{i,x}} = \nabla_{D_{x_i+1,y_i,i,A_i}} - \nabla_{D_{x_i-1,y_i,i,A_i}}, \tag{7}$$

$$\nabla_{P_{i,y}} = \nabla_{D_{x_i,y_i+1,i,A_i}} - \nabla_{D_{x_i,y_i-1,i,A_i}}, \tag{8}$$

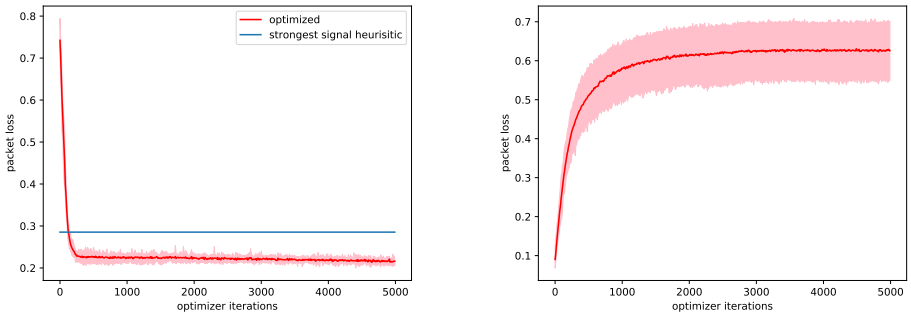
$$\nabla_{D_i} = \nabla_{D_{x_i,y_i,i,A_i}}, \tag{9}$$

$$\nabla_{A_{ij}} = \frac{\sum_{xy} \nabla_{D_{x,y,i,j}}}{mn}. \tag{10}$$

The gradients with respect to UE position (equations 7 and 8) are obtained from the difference of gradients in the neighboring grid squares. Gradients with respect to UE demand (equation 9) are equal to the gradient of D at the UEs position and current assignment. Gradients with respect to assignment (equation 10) are obtained by taking the mean value of the gradient over the entire map.

4 Optimization examples

To verify that the gradients produced by our simulator are useful for optimization we tackle two optimization problems. The first is the UE-cell assignment problem where UE positions and load demands are fixed, and the goal is to assign each UE to a cell so as to minimize packet loss. The second is the problem of finding a challenging spatial UE load distribution, where we strive to place UEs so that the packet loss is maximized. The first problem serves as a test of the utility of the cell assignment gradients, while the second tests the gradients of UE position.



(a) UE-cell assignment.

(b) Difficult load scenario finding.

Figure 1: Optimization curve showing packet loss as a function of iteration number. Optimization is performed ten times. Red line shows mean performance, pink shows (min, max) range. In panel (a) the blue line is the packet loss obtained by assigning each UE to the cell with which it has the best signal quality, an industry-standard heuristic.

UE-cell assignment We initialize each element of A_{ij} with random sample from a unit normal distribution, then optimize A_{ij} so as to minimize packet loss using Adam (with learning rate = 0.01). The resulting learning curve is presented in figure 1(a), where the process is repeated ten times with different random initializations. As expected, we obtain a decreasing learning curve, indicating that our gradients are indeed useful in performing optimization. As a baseline, figure 1(a) also plots the packet loss that would be attained if each UE was assigned to the cell to which is has the best signal strength, and our optimization approach outperforms this industry-standard heuristic.

Difficult load scenario finding Given a load balancing algorithm, it is useful to find load distribution scenarios where that algorithm performs poorly, so that it may be improved. Since our simulator is differentiable with respect to the position of UEs, we can use it to search for a worst case scenario for a given load balancing algorithm. To demonstrate this, we adopt the maximum signal strength heuristic as a load balancing algorithm, fix the demand of each UE, and then search for the most difficult scenario by maximizing packet loss by allowing the optimizer to adjust the positions of the UEs. Optimization is performed by using Adam (learning rate = 0.01) to optimize the UE position matrix after it is initialized with a uniform random distribution. Unfortunately, the max function applied in equation 4 creates a road block to the direct application of this method by causing the capacities of all areas with sufficiently high SINR to be equal, effectively removing any UE position gradients in the area. We get around this by using $C = C_{SH}$ for the optimization process. This approximation is only inaccurate for UEs that have good connections to base stations, and we intuit that the optimized solution will have few ofs those. Results are presented in figure 1(b), which confirms that a very large increase in packet loss (compared to the random uniform initialization) can be reliably attained, though the exact extent of this improvement has a non-trivial dependence on the initial state. Evaluation of packet loss for optimized UE positions using the non-approximate capacity factor (equation 4) reveals that the approximation $C = C_{SH}$ does not introduce any error for these scenarios.

5 Conclusion

This work presents a novel, differentiable, system-level RAN simulator and demonstrates its application in two important network optimization tasks. We relax locality constraints on UE position and assignment to facilitate gradient computation, then provide a method to re-introduce them. Our packet loss computation approach, though simple, captures the two most critical factors impacting system level performance: good performance requires UEs to have a high-quality connection to their serving cell, and serving cells must not be over-loaded. In future work, we would like to expand our approach to increase the fidelity of the packet loss computation by fitting it to real data. The design of the simulator is amenable to the use of empirically determined signal strength maps, meaning it can be applied to real-world scenarios without the need for extensive environment modeling.

References

[1] M. Agiwal, A. Roy, and N. Saxena. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 18(3):1617–1655, 2016.

[2] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000. Expanded version available as USC TR 99-702b at `http://www.isi.edu/%7ejohnh/PAPERS/Bajaj99a.html`.

[3] Ivan Stojmenovic. Simulations in wireless sensor and ad hoc networks: matching and advancing models, metrics, and solutions. *IEEE Communications Magazine*, 46(12):102–107, 2008.

[4] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[5] A. R. A. Kumar, S. V. Rao, and D. Goswami. Ns3 simulator for a study of data center networks. In *2013 IEEE 12th International Symposium on Parallel and Distributed Computing*, pages 224–231, 2013.

[6] Lelio Campanile, Marco Gribaudo, Mauro Iacono, Fiammetta Marulli, and Michele Mastroianni. Computer network simulation with ns-3: A systematic literature review. *Electronics*, 9(2):272, 2020.

[7] Guowang Miao, Jens Zander, Ki Won Sung, and Slimane Ben Slimane. *Fundamentals of mobile data networks*. Cambridge University Press, 2016.