Optimal-Capacity, Shortest Path Routing in Self-Organizing 5G Networks using Machine Learning

Chetana V. Murudkar* and Richard D. Gitlin, *Life Fellow, IEEE*
Innovation in Wireless Information Networking Lab (iWINLAB)
Department of Electrical Engineering,
University of South Florida,
Tampa, Florida 33620, USA
Email: cvm1@mail.usf.edu, richgitlin@usf.edu
**Sprint Corp.*, USA. Email: chetana.v.murudkar@sprint.com

Abstract—**Machine learning is expected to be a key enabler in 5G wireless self-organizing networks (SONs) that will be significantly more autonomous, smarter, adaptable and user-centric than current networks. This paper proposes a methodology, User Specific-Optimal Capacity Shortest Path (US-OCSP) routing, that uses machine learning to determine the resource-based optimum-capacity shortest path for a user between source and destination. The methodology takes into account two primary metrics, available capacity at network nodes (eNodeBs/gNodeBs) and distance, that are critical in determining the optimal path for an end-user. An *ns-3* simulation determines the capacity, which is measured by the availability of resources [i.e., Physical Resource Blocks (PRBs)] at all possible serving network nodes between the source and destination, that is followed by implementation of *Q-learning*, a *reinforcement* type of machine learning algorithm that determines the shortest path avoiding congested network nodes so as to achieve the required throughput and/or bit rate. The ability to determine the optimal-capacity shortest path route will facilitate effective resource allocation that will optimize end-user satisfaction in a 5G SON network.**

Keywords – 5G, Machine learning, ns-3, Q-learning, reinforcement learning, SON

I. Introduction

A central challenge for emerging 5G wireless communication networks, beyond the promise to deliver faster speeds and greater connectivity, is to optimize the ability of wireless network service providers to efficiently deliver the required user capacity over the available spectrum resources. A self-organizing network (SON) is recognized as central to capacity optimization of mobile networks [1]. Optimization of capacity in a SON includes mobility load balancing (MLB). MLB is a function where cells suffering congestion can transfer the load to other cells which have spare resources [2] and one of the promising ways to autonomously and intelligently manage MLB is by integrating it with machine learning (ML).

This paper proposes a methodology called User Specific, Optimal Capacity and Shortest Path (US-OCSP) that performs user-specific dynamic routing to find the shortest path with optimal capacity given a source and destination. The methodology uses the percentage of allocated Physical Resource Blocks (PRBs) to evaluate the available capacity of 4G/5G network nodes (eNodeBs/gNodeBs) [2] and *Q-learning*, an ML *reinforcement learning* technique, to determine the shortest path that meets the capacity needs of a user in a SON network. The implementation of US-OCSP is demonstrated using an *ns-3* simulator and implemented in Python.

There have been various research papers with network-centric approaches towards capacity optimization and load balancing. The authors in [3] proposed a load balancing strategy where a congested cell borrows channels from adjacent cells that are less loaded. A handover-based approach for load balancing in an LTE network is presented in [4], where a load balancing algorithm evaluates the load condition in a cell and neighboring cells and estimates the impact of changing the handover parameters to improve the network performance. The work in [5] improves capacity by making antenna tilt changes to reduce the blocking probability in a congested cell by shifting traffic to the neighboring cells.

In this paper, a user-centric approach is taken that tailors the capacity needs of the end-user to find the shortest path with optimal capacity for a given source and destination. Capacity is measured by the availability of resources [i.e., Physical Resource Blocks (PRBs)] at all possible serving eNodeBs/gNodeBs between the source and destination. The machine learning algorithm determines the shortest path avoiding congested network nodes so as to achieve the required throughput and/or bit rate. In other words, under the assumption that a user will be served by multiple network nodes while moving from its source to destination no matter what route it takes, the algorithm proposed will give the user optimal throughput by selecting a path with the least viable distance that goes through the network nodes that have good availability of resources (PRBs) to serve the user. The algorithm avoids selecting a path that goes through congested network nodes that have very high PRB utilization. So, if the user takes the recommended path, the user will be able to achieve an optimal throughput/ bit rate. The paper is organized as follows: Section II explains the proposed methodology, US-OCSP. The simulation results and observations are presented in Section III. The paper ends with concluding remarks in Section IV.

II. THE METHODOLOGY

The visual map depicted in Fig.1 is used to illustrate US-OCSP. Given a source and a destination of a user, US-OCSP first determines the available capacity of all eNodeBs (eNBs) in terms of 4G or gNodeBs (gNBs) in terms of 5G that could potentially serve the user. This is done by calculating the PRB utilization of each eNB/gNB. PRB utilization is a performance measurement used for 4G/5G network functions that provide the total usage (in percentage) of physical resource blocks and is obtained as [6]:

$$M(T) = \frac{M1(T)}{P(T)} * 100 \tag{1}$$

where $M(T)$ is the percentage of PRBs used, averaged during time period T with value range: 0-100%, $M1(T)$ is the count of all PRBs used, $P(T)$ is the total number of PRBs available during time period T, and T is the time period during which the measurement is performed.

The network scenario described in Fig.1 is simulated using the LTE-EPC simulation model of the *ns-3* simulator [7] that provides the interconnection of multiple UEs to the internet, via a radio access network of multiple eNBs connected to a single SGW/PGW node (not shown). In the simulation, 100 UEs are randomly placed in the network and are connected to the closest eNB. Areas with high user concentration are denoted by beach, university and shopping mall symbols in the visual map. PRB utilization is calculated for every eNB to evaluate its capacity. A threshold of 70% is set such that an eNB with PRB utilization above the threshold is declared to be "busy" while an eNB with PRB utilization below the threshold is declared to be "available."

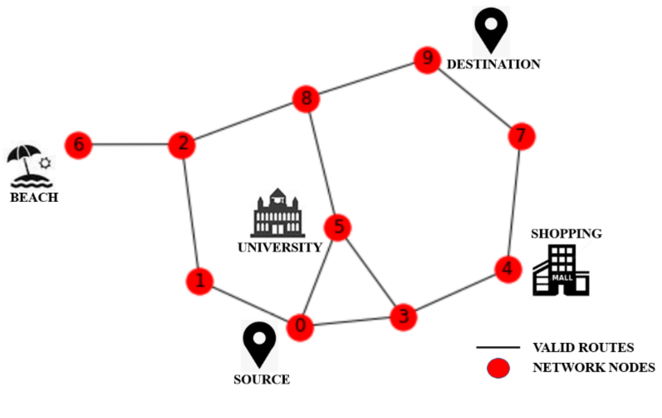
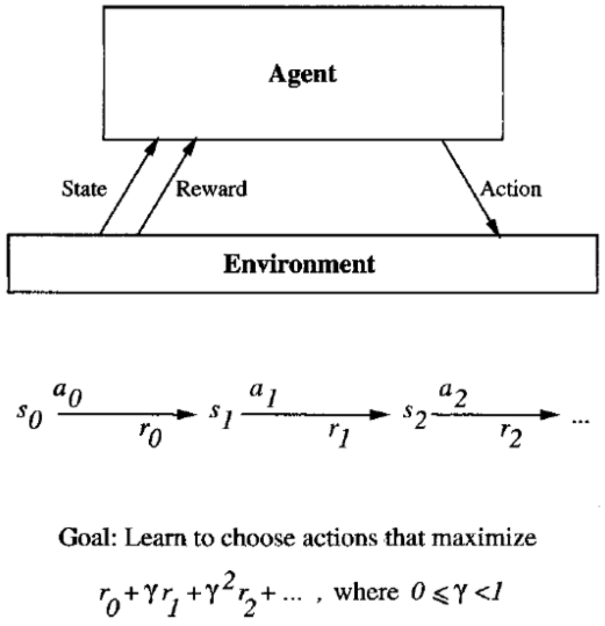


Fig. 1 A visual map describing a network scenario used to illustrate US-OCSP.

US-OCSP uses a machine learning algorithm called *Q-learning*, a form of *reinforcement learning*, to determine the shortest path to be taken by the end-user from its source to destination. *Reinforcement learning* addresses the question of how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals and is described in Fig. 2 [8]. Fig. 2 can be explained as follows [8]:

Each time the agent performs an action a from a set of possible actions A in some state s in an environment described by a set of possible states S, the agent receives a reward or penalty r that represents an immediate value of the state-action transition indicating the desirability of the resulting state. This generates a sequence of states s_i, actions a_i, and immediate rewards r_i as shown in Fig. 2. The task of the agent is to learn a control policy, $\pi : S \rightarrow A$, that would maximize the expected sum of rewards, with future rewards discounted exponentially by their delay. The discount factor is denoted by γ.



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{ , where } 0 \leq \gamma < 1$$

Fig. 2 Representation of a *reinforcement learning* system.

The *Q-learning* algorithm is described in Table I [8] and can be explained as follows [8]:

In *Q-learning*, learning the Q function corresponds to learning the optimal policy. The evaluation function the agent attempts to learn is $Q(s, a)$ such that the value of Q is the maximum discounted cumulative reward that can be achieved starting from state s and applying action a as the first action. In this algorithm, the learner represents its hypothesis $\hat{Q}$ by a large table that consists of a separate entry for each pair of state and action. The table entry for (s, a) stores the value for $\hat{Q}(s, a)$, the learner's current hypothesis about the actual but unknown value $Q(s, a)$. The initial values of the table are set to zero. The agent recurrently observes its current state s, chooses some action a, executes action a, then observes the resulting reward

$r = r(s, a)$ and the new state $s' = \delta(s, a)$ where δ is the state transition function and denotes the state resulting from applying action a to state s . It further updates the table entry for $\hat{Q}(s, a)$ following each such transition in accordance to the rule given by (2):

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a') \tag{2}$$

This training rule uses the agent's current $\hat{Q}$ values for the new state s' to refine its estimate of $\hat{Q}(s, a)$ for the previous state s. Q*-learning* propagates $\hat{Q}$ estimates one step backwards i.e. each time the agent moves forward from a previous state to a new one, Q*-learning* propagates $\hat{Q}$ estimates backward from the new state to the old state. At the same time, the immediate reward received by the agent for the state-action transition is used to augment these propagated values of $\hat{Q}$. After multiple iterations, the information that the agent collects will propagate from the transitions with non-zero rewards back through the entire state-transition space available to the agent, resulting eventually in a table that consists of the Q values. Using this algorithm, the agent's estimate $\hat{Q}$ converges in the limit to the actual Q function, provided the system can be modeled as a deterministic Markov decision process, the reward function r is bounded, and actions are chosen such that every pair of state-action is visited infinitely often. A significant aspect of Q*-learning* that makes it scalable is that it can be employed in an arbitrary environment where the agent or the learner has no prior knowledge of how its actions affect its environment. The agent is not required to be able to predict in advance the immediate result for every possible state-action. In other words, the algorithm assumes the agent does not have knowledge of $\delta(s, a)$ and $r(s, a)$, and that instead of moving about in an internal mental model of the state space, it must move about the real world and observe the consequences. Hence, the algorithm can be applied even if there are newly added states and actions.

TABLE I. Q-LEARNING ALGORITHM, ASSUMING DETERMINISTIC REWARDS AND ACTIONS. THE DISCOUNT FACTOR γ MAY BE ANY CONSTANT SUCH THAT $0 \leq \gamma < 1$.

```
For each $s$, $a$ initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state $s$
Do forever:
    • Select an action $a$ and execute it
    • Receive immediate reward $r$
    • Observe the new state $s'$
    • Update the table entry for $\hat{Q}(s, a)$ as follows:
          $\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$
    • $s \leftarrow s'$
```

Using the knowledge of PRB utilization gained from the output of the *ns-3* simulation, US-OCSP has at its core the Q-*learning* algorithm implemented in Python and finds the shortest path with optimal capacity that the user should take while in transit from a given source to destination. A table of correspondence to show the representation of network functions considered in US-OCSP with regards to the Q-*learning* parameters is given in Table II. Considering the network scenario described in Fig. 1, the agent (virtual user) will explore different paths going from the end-user's source and destination using US-OCSP to find the optimal path. Every time the agent moves from one network node to another, it will receive an immediate reward for the transition whose value depends on whether or not there is a valid link established between the two network nodes and how close or far the network nodes are from the destination. It then computes the $\hat{Q}$ value for that transition using the Q*-learning* algorithm and continues to refine the $\hat{Q}$ value until it virtually reaches the network node that serves the destination. The availability of network nodes is determined based on their PRB utilization derived from the *ns-3* simulation. The optimal path corresponds to selecting the links or routes with maximal Q values. In other words, the computed Q[1] values will help determine which node to node transitions should be selected to achieve the shortest path with optimal capacity.

TABLE II. TABLE OF CORRESPONDENCE SHOWING NETWORK MAPPING IN US-OCSP WITH Q-*learning* PARAMETERS

***Q-learning* Parameters**	**Network Mapping**
s	A state corresponds to a network node (eNB/gNB).
a	An action corresponds to the agent's virtual movement from one network node to another.
$r(s, a)$	A reward is an immediate/instant value, or score received after every virtual move of the agent from one network node to another.
$Q(s, a)$	A Q value is computed and refined recursively using the Q *-learning* algorithm until the agent virtually reaches the network node that serves the destination.

[1] Q is an evaluation function/utility function such that the value of Q for the current state and action summarizes in a single number all the information needed to determine the discounted cumulative reward that will be gained in the future if that state-action pair is selected [8].

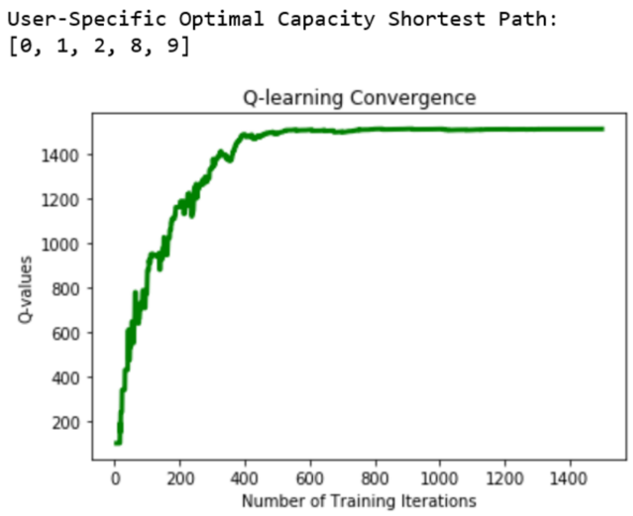
III. Simulation Results and Observations

Table III provides the simulation parameters used to implement the network scenario described in Section III using *ns-3*.

TABLE III. SIMULATION SET UP PARAMETERS

Parameter	Value
Number of UEs	100
Number of eNBs	10
eNB Bandwidth	20 MHz (100 PRBs)
Scheduler	Token Bank Fair Queue Scheduler (TBFQ)[2]
Traffic Type	Constant Bit Rate (CBR)

The output of the *ns-3* simulation gives the modulation coding scheme used and the transport block size for every UE-eNB pair per unit time. The output is further used to find the PRB utilization by referring to 3GPP standards [9] and implementing equation (1). If the PRB utilization of an eNB was above 70%, it was declared as "busy" and if the PRB utilization of an eNB was below 70%, it was declared as "available" in terms of capacity. In accordance to this, eNBs 4, 5 and 6 from Fig. 1 were declared to be "busy" eNBs whereas eNBs 0, 1, 2, 3, 7, 8 and 9 were declared to be "available." The *Q-learning* algorithm is then implemented in Python for determining the shortest path between the end-user's source and destination. When distance is the only criteria used to determine the path that the end-user should take, the recommended path given by the algorithm goes via eNBs denoted by nodes 0, 5, 8 and 9. This path is the shortest path that the end-user can take to reach to the destination, but is not the most efficient path as it does not verify if all the serving eNBs on this path have enough capacity available to serve the end-user. In order to find the most efficient path, the status of all the eNBs based on their PRB utilization derived from the *ns-3* simulation is given as an input to the *Q-learning* algorithm. With this knowledge, the *Q-learning* algorithm takes into account not only distance but also available nodal capacity (i.e., eNB or gNB availability of PRBs) while determining the most efficient path the end-user should take given a source and a destination. Subsequently, the most efficient path suggested by US-OCSP goes via eNBs denoted by nodes 0, 1, 2, 8, and 9 as shown in Fig. 3. Thus, US-OCSP not only avoids all other possible paths that may be longer and more time-consuming, but it also avoids selecting paths that may be served by eNBs that due to congestion cannot meet the capacity (throughput and/or bit rate) needs of the end-user.



```
User-Specific Optimal Capacity Shortest Path:
[0, 1, 2, 8, 9]
```

Fig. 3 A graphical representation of the Q-learning curve converging towards the optimal solution.

IV. Conclusions

This paper introduced and proposed a methodology called US-OCSP that gives the most efficient path to be followed by an end-user in terms of distance and capacity in a SON network using ML given its source and destination. The ML algorithm used in this research is *Q-learning,* a form of *reinforcement learning*. The effectiveness of this methodology is demonstrated using a network scenario simulated in *ns-3* followed by the ML implementation in Python. The results showed that the shortest path with optimum capacity is rapidly determined. This methodology could help network providers to meet the end-user demands by finding the most efficient path and to optimize network resource allocation. This paper demonstrates the potential for implementing US-OCSP in future networks that will be autonomous and user-centric by incorporating ML in SON networks where the system can provide the most optimal path for end-users while moving from a given source to destination.

References

[1] Ljupco Jorguseski, Adrian Pais, Fredrik Gunnarsson, Angelo Centonza, Colin Willcock, "Self-organizing networks in 3GPP: standardization and future trends," *IEEE Communications Magazine*, Volume: 52, Issue: 12, Pages: 28 – 34, 2014.

[2] *3GPP, The Mobile Broadband Standard*, [Online]. Available: http://www.3gpp.org/

[2] TBFQ, a channel-aware/QoS-aware scheduler derived from the leaky-bucket mechanism, guarantees the fairness by utilizing a shared token bank and can be explained as follows [7]: TBFQ maintains a shared token bank so as to balance the traffic between different flows. The user who contributes more on token bank has a higher priority to borrow tokens while the user who borrows more tokens from the bank has a lower priority to continue to withdraw tokens. Users suffering from severe interference and shadowing conditions get more opportunities to borrow tokens from the bank.

[3] S. Das, S. Sen, and R. Jayaram, “A structured channel borrowing scheme for dynamic load balancing in cellular networks,” *Proceedings of 17th International Conference on Distributed Computing Systems*, Pages: 116 – 123, 1997.
[4] Andreas Lobinger, Szymon Stefanski, Thomas Jansen, Irina Balan, “Load Balancing in Downlink LTE Self-Optimizing Networks,” *2010 IEEE 71st Vehicular Technology Conference*, Pages: 1 – 5, 2010.
[5] Vlad-Ioan Bratu, Claes Beckman, “Base station antenna tilt for load balancing,” *2013 7th European Conference on Antennas and Propagation (EuCAP)*, Pages: 2039 – 2043, 2013.
[6] 3GPP TS 28.552, “5G performance measurements,” V15.1.0, December 2018.
[7] *ns-3* [online]. Available : https://www.nsnam.org/
[8] Tom M. Mitchell, *Machine Learning*, McGraw Hill, March 1, 1997.
[9] 3GPP TS 36.213, “LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures,” version 12.3.0 Release 12, October 2014.