ADAPTIVE VNF SCALING AND FLOW ROUTING WITH PROACTIVE DEMAND PREDICTION

Chris Manriquez

ABSTRACT SUMMARY

In order to best utilize virtualized network functions (VNFs) they need to be scaled properly and dynamically. Most solutions to the VNF problem are to deal with the scalability by reaction rather than being proactive. By using Online Machine Learning and three algorithms they can efficiently instantiate new instances of VNFs and provide optimal service chain routing. Overall, it provides a solution that best utilizes cloud resources while being adaptive.

ONLINE MACHINE LEARNING

What is Online Machine Learning?

Online machine learning (now referred to as online learning) also known as sequential order or real-time learning is a method of machine learning in which data is made available as to the algorithm as soon as it is available.

Regular machine learning may follow a batched learning schedule. The variety of the schedule can change but most typically training on new data is set on a weekly schedule. Therefore, you are constantly learning from yesterday's data.

ONLINE CONVEX OPTIMIZATION

- Online Convex Optimization (OCO) is used for decision making, utilizing efficient algorithms. The goal of OCO is minimize regret (loss).
- Specifically for our problem they are utilizing the OCO algorithm variation of Follow The Leader known as Follow the regularized leader (FTRL)

- Follow The Leader: Choose the hypothesis that had the least amount of loss for all previous iterations.
- Follow The Regularized Leader: The same concept as follow the leader but utilizes a regularizer to make the solution more stable.

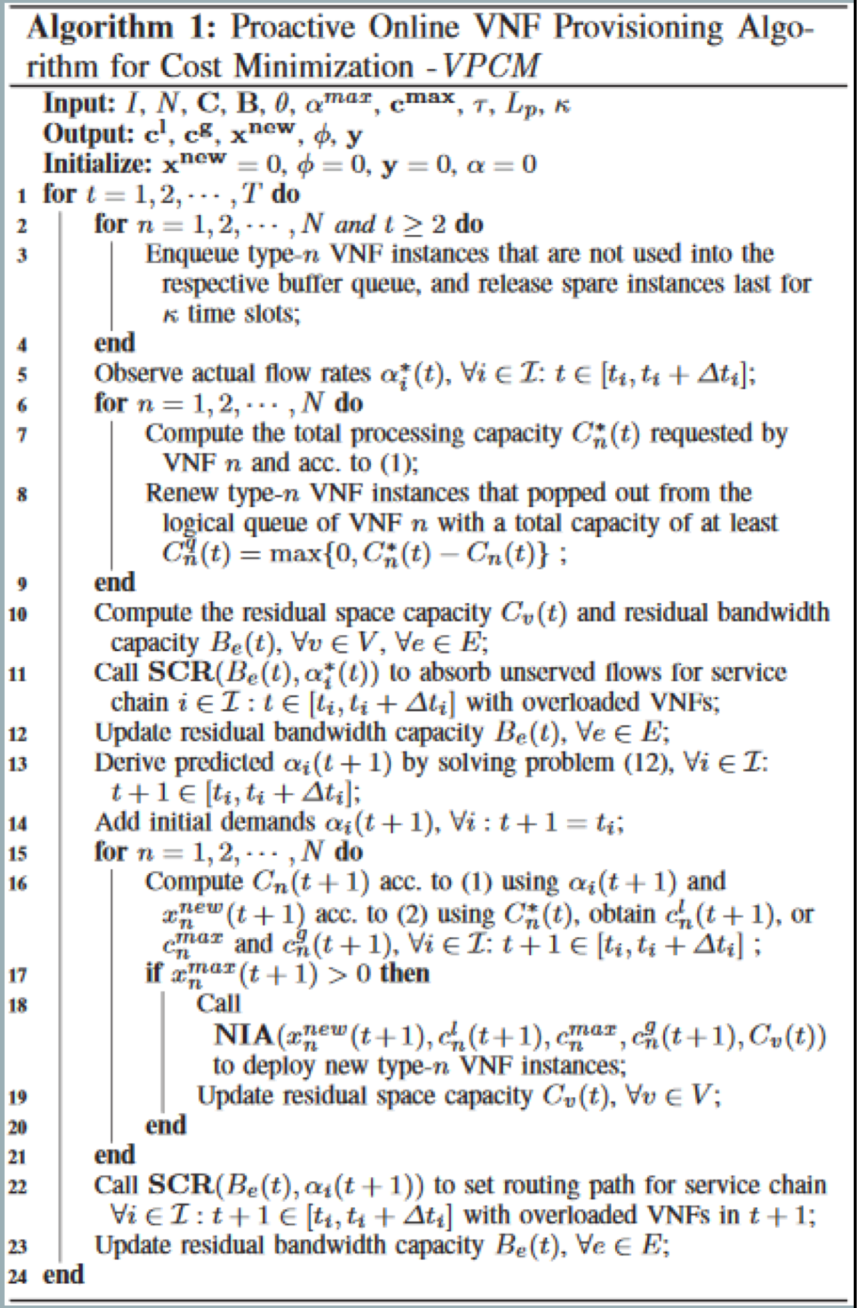
OCO AND FTRL CONTINUED

The approach taken in this paper for VNF Scaling and Flow routing relies heavily on these two concepts. As the authors use OCO and FTRL to learn how to accurately deploy VNF instances, scale them, and route traffic flow. By using FTRL it will allow the prediction error on future VNF instances to be minimized and less resource intensive.

THE ALGORITHMS

- Proactive Online VNF Provisioning Algorithm for Cost Minimization (VCPM)
- Online Algorithm for New Instance Assignment (NIA)
- Online Algorithm For Service Chain Routing (SCR)

PROACTIVE ONLINE VNF PROVISIONING ALGORITHM FOR COST MINIMIZATION



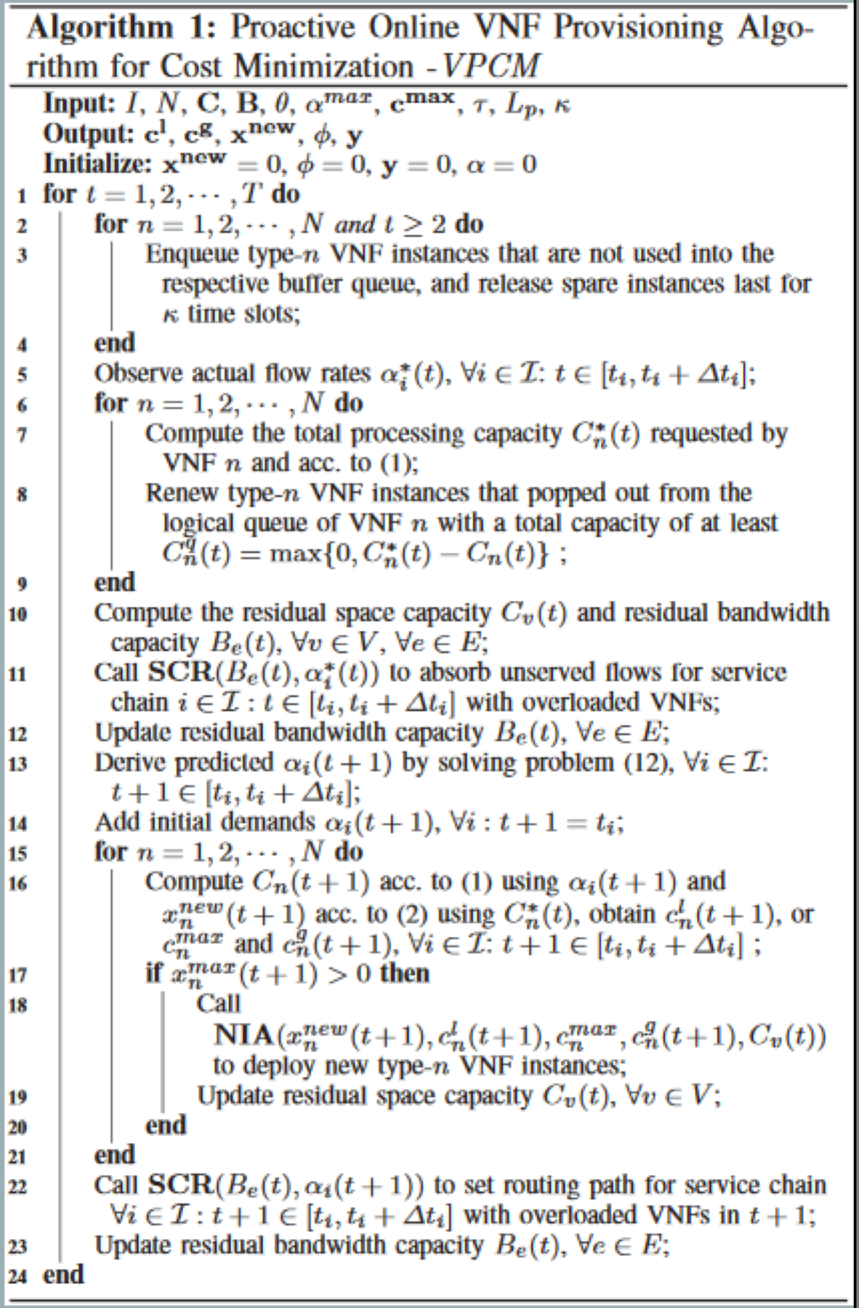
Algorithm 1: Proactive Online VNF Provisioning Algorithm for Cost Minimization - *VPCM*

```
   Input: $I$, $N$, $\mathbf{C}$, $\mathbf{B}$, $\theta$, $\alpha^{max}$, $\mathbf{c}^{\max}$, $\tau$, $L_p$, $\kappa$
   Output: $\mathbf{c}^{l}$, $\mathbf{c}^{g}$, $\mathbf{x}^{new}$, $\phi$, $\mathbf{y}$
   Initialize: $\mathbf{x}^{new} = 0$, $\phi = 0$, $\mathbf{y} = 0$, $\alpha = 0$
1  for $t = 1, 2, \cdots, T$ do
2      for $n = 1, 2, \cdots, N$ and $t \geq 2$ do
3          Enqueue type-$n$ VNF instances that are not used into the
            respective buffer queue, and release spare instances last for
            $\kappa$ time slots;
4      end
5      Observe actual flow rates $\alpha_i^*(t)$, $\forall i \in \mathcal{I}$: $t \in [t_i, t_i + \Delta t_i]$;
6      for $n = 1, 2, \cdots, N$ do
7          Compute the total processing capacity $C_n^*(t)$ requested by
            VNF $n$ and acc. to (1);
8          Renew type-$n$ VNF instances that popped out from the
            logical queue of VNF $n$ with a total capacity of at least
            $C_n^g(t) = \max\{0, C_n^*(t) - C_n(t)\}$ ;
9      end
10     Compute the residual space capacity $C_v(t)$ and residual bandwidth
        capacity $B_e(t)$, $\forall v \in V$, $\forall e \in E$;
11     Call $\mathbf{SCR}(B_e(t), \alpha_i^*(t))$ to absorb unserved flows for service
        chain $i \in \mathcal{I} : t \in [t_i, t_i + \Delta t_i]$ with overloaded VNFs;
12     Update residual bandwidth capacity $B_e(t)$, $\forall e \in E$;
13     Derive predicted $\alpha_i(t+1)$ by solving problem (12), $\forall i \in \mathcal{I}$:
        $t + 1 \in [t_i, t_i + \Delta t_i]$;
14     Add initial demands $\alpha_i(t+1)$, $\forall i : t + 1 = t_i$;
15     for $n = 1, 2, \cdots, N$ do
16         Compute $C_n(t+1)$ acc. to (1) using $\alpha_i(t+1)$ and
            $x_n^{new}(t+1)$ acc. to (2) using $C_n^*(t)$, obtain $c_n^l(t+1)$, or
            $c_n^{max}$ and $c_n^g(t+1)$, $\forall i \in \mathcal{I}$: $t + 1 \in [t_i, t_i + \Delta t_i]$ ;
17         if $x_n^{max}(t+1) > 0$ then
18             Call
                $\mathbf{NIA}(x_n^{new}(t+1), c_n^l(t+1), c_n^{max}, c_n^g(t+1), C_v(t))$
                to deploy new type-$n$ VNF instances;
19             Update residual space capacity $C_v(t)$, $\forall v \in V$;
20         end
21     end
22     Call $\mathbf{SCR}(B_e(t), \alpha_i(t+1))$ to set routing path for service chain
        $\forall i \in \mathcal{I} : t + 1 \in [t_i, t_i + \Delta t_i]$ with overloaded VNFs in $t + 1$;
23     Update residual bandwidth capacity $B_e(t)$, $\forall e \in E$;
24 end
```

VPCM is the algorithm that will employ the FTRL method to predict the flow rate while minimizing the surrogate loss function*

- Begin by enqueueing the previously unused in the last time slot. for t > 2. (Line 2 and 3)
- After it observes the flow rates and the proceeds to calculate the required capacity requested for a VNF. If the required capacity is less than the estimated, it will renew the popped instances to use them to sufficient capacity. (Line 5)
- Unlike other methods this algorithm estimates processing capacity of the next time slot by comparing it with the current capacity and determines whether the VNF will be overloaded. (Line 10)
- Then the SCR will be called. (Line 11)
- Update residual bandwidth. (Line 12)

*Surrogate loss function: When a problem does not fit the framework of OCO a surrogate loss function is used for convexification allowing the problem to fit within the framework.



Algorithm 1: Proactive Online VNF Provisioning Algorithm for Cost Minimization -*VPCM*

```
   Input: $I$, $N$, $\mathbf{C}$, $\mathbf{B}$, $\theta$, $\alpha^{max}$, $\mathbf{c}^{\mathbf{max}}$, $\tau$, $L_p$, $\kappa$
   Output: $\mathbf{c}^{\mathbf{l}}$, $\mathbf{c}^{\mathbf{g}}$, $\mathbf{x}^{\mathbf{new}}$, $\phi$, $\mathbf{y}$
   Initialize: $\mathbf{x}^{\mathbf{new}} = 0$, $\phi = 0$, $\mathbf{y} = 0$, $\alpha = 0$
1  for $t = 1, 2, \cdots, T$ do
2      for $n = 1, 2, \cdots, N$ and $t \geq 2$ do
3          Enqueue type-$n$ VNF instances that are not used into the
             respective buffer queue, and release spare instances last for
             $\kappa$ time slots;
4      end
5      Observe actual flow rates $\alpha_i^*(t)$, $\forall i \in \mathcal{I}$: $t \in [t_i, t_i + \Delta t_i]$;
6      for $n = 1, 2, \cdots, N$ do
7          Compute the total processing capacity $C_n^*(t)$ requested by
             VNF $n$ and acc. to (1);
8          Renew type-$n$ VNF instances that popped out from the
             logical queue of VNF $n$ with a total capacity of at least
             $C_n^g(t) = \max\{0, C_n^*(t) - C_n(t)\}$ ;
9      end
10     Compute the residual space capacity $C_v(t)$ and residual bandwidth
         capacity $B_e(t)$, $\forall v \in V$, $\forall e \in E$;
11     Call $\mathbf{SCR}(B_e(t), \alpha_i^*(t))$ to absorb unserved flows for service
         chain $i \in \mathcal{I} : t \in [t_i, t_i + \Delta t_i]$ with overloaded VNFs;
12     Update residual bandwidth capacity $B_e(t)$, $\forall e \in E$;
13     Derive predicted $\alpha_i(t+1)$ by solving problem (12), $\forall i \in \mathcal{I}$:
         $t + 1 \in [t_i, t_i + \Delta t_i]$;
14     Add initial demands $\alpha_i(t+1)$, $\forall i : t + 1 = t_i$;
15     for $n = 1, 2, \cdots, N$ do
16         Compute $C_n(t+1)$ acc. to (1) using $\alpha_i(t+1)$ and
             $x_n^{new}(t+1)$ acc. to (2) using $C_n^*(t)$, obtain $c_n^l(t+1)$, or
             $c_n^{max}$ and $c_n^g(t+1)$, $\forall i \in \mathcal{I}$: $t + 1 \in [t_i, t_i + \Delta t_i]$ ;
17         if $x_n^{max}(t+1) > 0$ then
18             Call
                 $\mathbf{NIA}(x_n^{new}(t+1), c_n^l(t+1), c_n^{max}, c_n^g(t+1), C_v(t))$
                 to deploy new type-$n$ VNF instances;
19             Update residual space capacity $C_v(t)$, $\forall v \in V$;
20         end
21     end
22     Call $\mathbf{SCR}(B_e(t), \alpha_i(t+1))$ to set routing path for service chain
         $\forall i \in \mathcal{I} : t + 1 \in [t_i, t_i + \Delta t_i]$ with overloaded VNFs in $t + 1$;
23     Update residual bandwidth capacity $B_e(t)$, $\forall e \in E$;
24 end
```

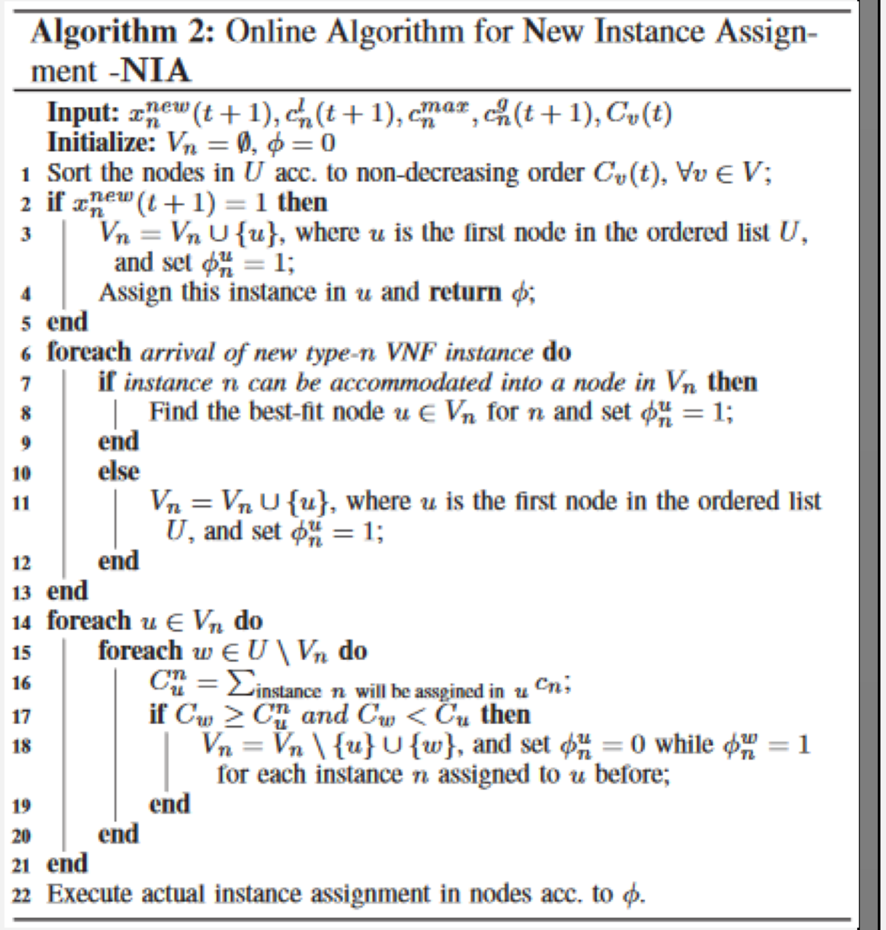
VPCM CONTINUED

The needed flow rate is then derived using the FTRL (Line 13)

Next, the initial demands allows the computation for new instances to be launched as well as their capacity. (Lines 15 and 16)

NIA will be called and deploy the new instances on the available nodes. (Line 18)

Finally, SCR is called one last time to route flows for the service chains that have overloaded VNFs. (Line 22)



Algorithm 2: Online Algorithm for New Instance Assignment -NIA

```
   Input: $x_n^{new}(t+1), c_n^l(t+1), c_n^{max}, c_n^g(t+1), C_v(t)$
   Initialize: $V_n = \emptyset$, $\phi = 0$
1  Sort the nodes in $U$ acc. to non-decreasing order $C_v(t)$, $\forall v \in V$;
2  if $x_n^{new}(t+1) = 1$ then
3      $V_n = V_n \cup \{u\}$, where $u$ is the first node in the ordered list $U$,
         and set $\phi_n^u = 1$;
4      Assign this instance in $u$ and return $\phi$;
5  end
6  foreach arrival of new type-$n$ VNF instance do
7      if instance $n$ can be accommodated into a node in $V_n$ then
8          Find the best-fit node $u \in V_n$ for $n$ and set $\phi_n^u = 1$;
9      end
10     else
11         $V_n = V_n \cup \{u\}$, where $u$ is the first node in the ordered list
             $U$, and set $\phi_n^u = 1$;
12     end
13 end
14 foreach $u \in V_n$ do
15     foreach $w \in U \setminus V_n$ do
16         $C_u^n = \sum_{\text{instance } n \text{ will be assgined in } u} c_n$;
17         if $C_w \geq C_u^n$ and $C_w < C_u$ then
18             $V_n = V_n \setminus \{u\} \cup \{w\}$, and set $\phi_n^u = 0$ while $\phi_n^w = 1$
                 for each instance $n$ assigned to $u$ before;
19         end
20     end
21 end
22 Execute actual instance assignment in nodes acc. to $\phi$.
```

ONLINE ALGORITHM FOR NEW INSTANCE ASSIGNMENT

New Instance assignment algorithm will take care of the actual assignment of VNF instances to nodes while attempting to use a node that will have maximum residual space capacity after.

If only one new instance is requested that will be assigned to the first node (Line 2-4)

Otherwise, n number instances need to be assigned (Line 6)

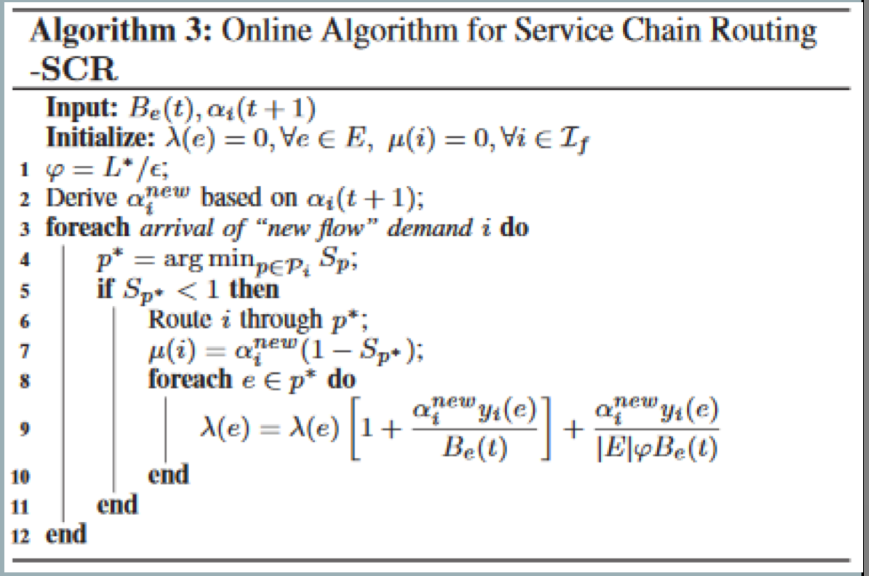
When using best fit, the attempt to assign the instance with the best fit node is attempted. (Line 7 and 8)

If it is not possible it will assign the instance to a new node (Line 10 and 11)

Lastly, swap some already in use nodes with nodes that not in use if they have a smaller capacity but enough to hold the capacity to hold the VNF instances. (Line 17 – 19)

Lastly, the actual assignment is done. (Line 22)

ONLINE ALGORITHM FOR SERVICE CHAIN ROUTING



Algorithm 3: Online Algorithm for Service Chain Routing -SCR

```
Input: $B_e(t), \alpha_i(t+1)$
Initialize: $\lambda(e) = 0, \forall e \in E,\ \mu(i) = 0, \forall i \in \mathcal{I}_f$
$\varphi = L^*/\epsilon$;
Derive $\alpha_i^{new}$ based on $\alpha_i(t+1)$;
foreach arrival of "new flow" demand $i$ do
    $p^* = \arg\min_{p \in \mathcal{P}_i} S_p$;
    if $S_{p^*} < 1$ then
        Route $i$ through $p^*$;
        $\mu(i) = \alpha_i^{new}(1 - S_{p^*})$;
        foreach $e \in p^*$ do
            $\lambda(e) = \lambda(e)\left[1 + \frac{\alpha_i^{new} y_i(e)}{B_e(t)}\right] + \frac{\alpha_i^{new} y_i(e)}{|E|\varphi B_e(t)}$
        end
    end
end
```

The SCR is responsible for routing flows for service chain demands with overloaded VNFs.

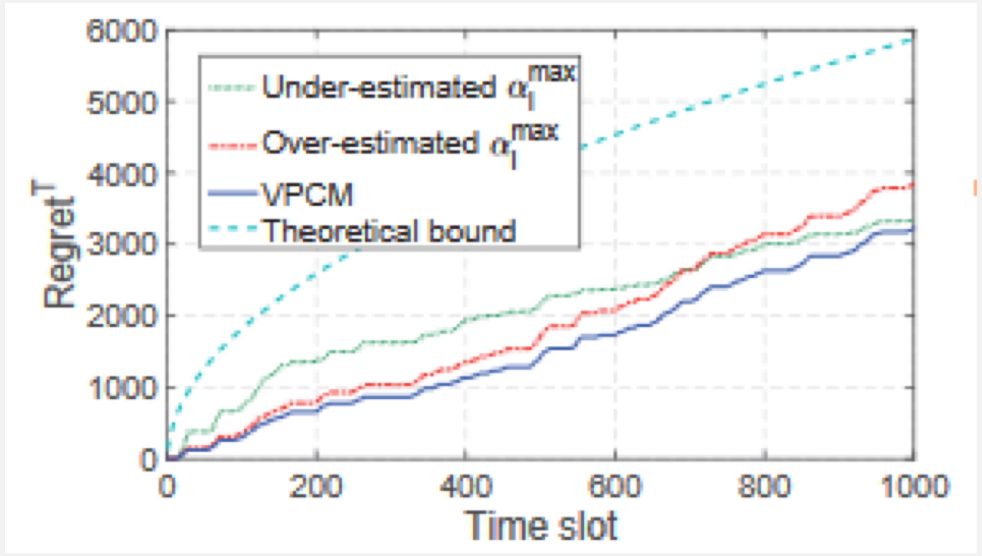
The input taken will be the residual bandwidth as well as the current flow rate.

The goal is to optimize the new flow along the paths for currently overloaded VNFs.

PERFORMANCE

In comparison to previous attempts using a graph neural network, or other reactive solutions we see that VPCM works well. The biggest advantage of it is the predictive nature used to minimize the total amount of regret.

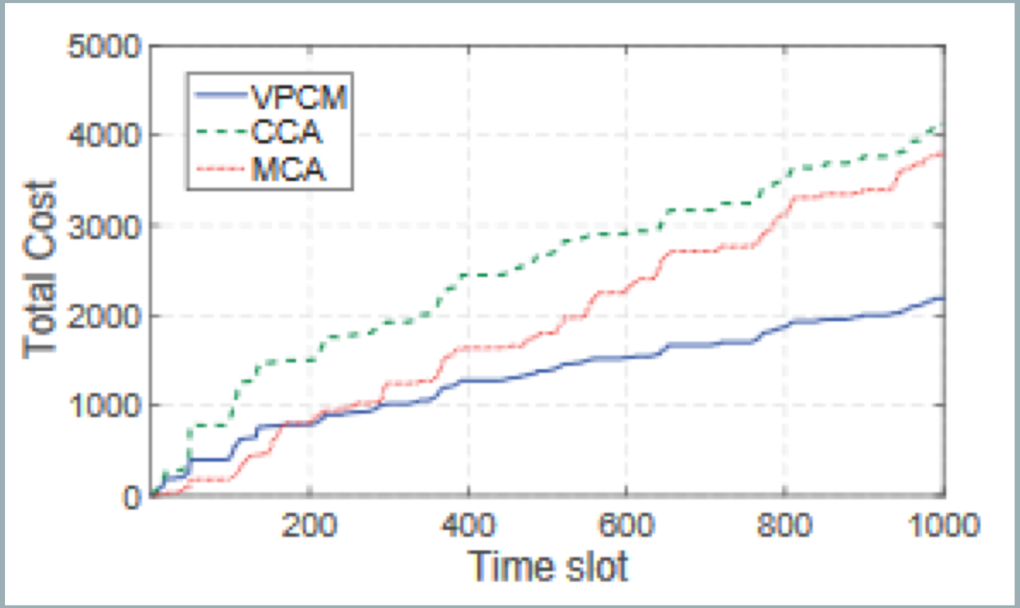
As we can see in the figure VPCM in this simulation has performed well when compared to inaccurate estimations flow rates. By utilizing online learning the simulation outperforms both under and overestimates.



PERFORMANCE

Next, we can look at the overall cost in comparison to different approaches. These other approaches include Constant Capacity Allocation (CCA) and Maximum Capacity Allocation (MCA).

For an ever-growing amount of time slots we can see the total cost of VPCM drops dramatically below the total cost of MCA or CCA.


Total Cost
Time slot
VPCM
CCA
MCA