University of Southern California
Viterbi School of Engineering

EE 550 Project – Data Center Network Analysis

Data Networks: Design and Analysis

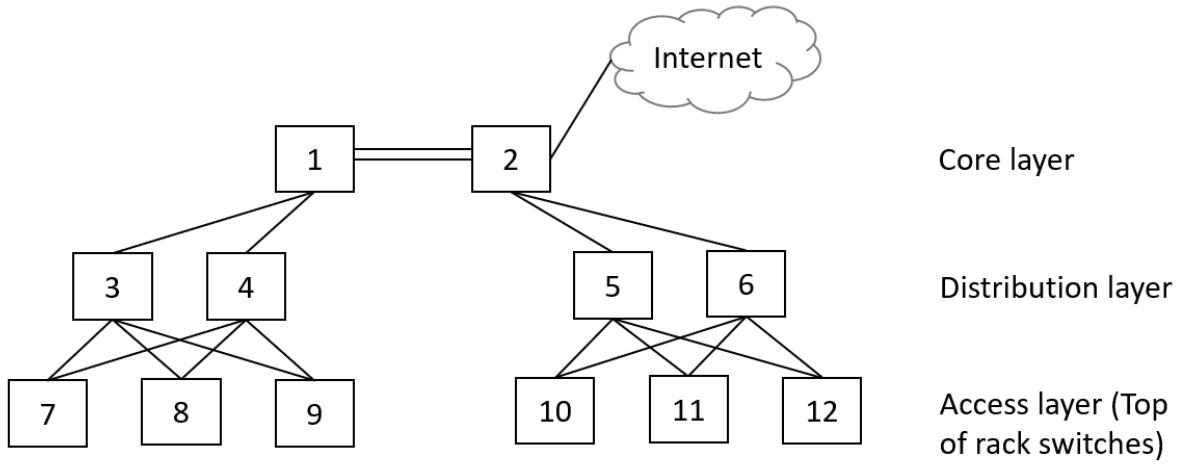
By

Wanwiset Peerapatanapokin

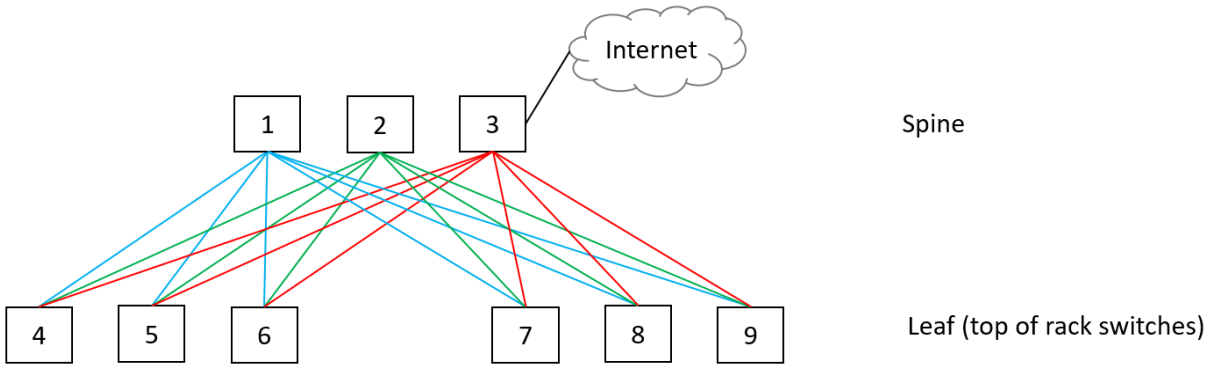
Contents

Problem Description

Traditionally, datacenter networks follow a 3 layer model. This includes the access layer, distribution layer and core layer. An example of this model is shown below



However, in recent years, cloud computing has become highly popular. This creates a demand towards East-West traffic (communication within the nodes of datacenter itself). To accommodate this trend, the new spine and leaf network design was introduced. An example of spine and leaf network architecture is shown below.



Notice that each spine node has a dedicated link to every leaf node. We want to explore how effective this new model is, compared to the traditional 3-layer model. One measure of the effectiveness of a network is the maximum throughput. We will compare various scenarios such as download between a single pair of nodes, multiple nodes download, and download throughput from the internet.

Analysis - Drift Plus Penalty Algorithm (DPP)

In class, we learned about the Drift Plus Penalty algorithm. This technique solves a convex program by utilizing virtual queues simulation. This algorithm is very good at separability. We can separate a complex model into single parameter optimization problems. Another advantage is that the resulting formulas can be applied with any network architectures, not just the 3-layer and the spine and leaf model.

The DPP algorithm is defined as the following.

DPP ALGORITHM

Consider a convex program:

$$\begin{aligned} \text{Minimize:} \quad & f(x) \\ \text{Subject to:} \quad & g_i(x) \leq c_i \quad \forall i \in \{1, \ldots, k\} \\ & x \in \mathcal{X} \end{aligned}$$

where $\mathcal{X}$ is a compact and convex set and $f, g_1, \ldots, g_k$ are convex functions. The DPP algorithm with parameter $V \geq 0$ is:

- Virtual queues: For each constraint $i \in \{1, \ldots, k\}$ define

$$Q_i(t+1) = \max[Q_i(t) + g_i(x(t)) - c_i, 0] \quad \forall t \in \{0, 1, 2, \ldots\}$$

  where $Q_i(0) = 0$.
- Choice of $x(t)$: Every slot $t \in \{0, 1, 2, \ldots\}$, we choose $x(t)$ to solve:

$$\begin{aligned} \text{Minimize:} \quad & Vf(x(t)) + \sum_{i=1}^{k} Q_i(t) g_i(x(t)) \\ \text{Subject to:} \quad & x(t) \in \mathcal{X} \end{aligned}$$

- Time average: After T iterations we form the guess: $\frac{1}{T} \sum_{t=0}^{T-1} x(t)$.

Credits to EE550 lecture notes of Prof. Michael Neely

We are going to explore the throughput as constrained by links. So, we assume that all nodes can handle unlimited amount of upload/download speed. Also, assume that the data transfer will use every possible path to reach the destination. That is where multiple flows are defined, one flow per each path.

Let us define the following variables:

- Define x_k as the rate into node k
- w_k as the weight priority of node k
- $y_{k,l}$ as the flow l of node k as there can be multiple paths to k
- And so, $x_k = \sum_{l=1}^{L} y_{k,l}$
- Let (i,j) represent the link between node i and j. If the link is valid, it belongs in the set {Links}
- Define $a_{i,j}^{k,l} = 1$ if the flow $y_{k,l}$ uses link (i,j). $a_{i,j}^{k,l} = 0$ otherwise.
- Link capacity $C_{i,j}$ for each link (i,j)

We choose to maximize a weighted log-fairness scheme to reflect a real-world data center prioritization scheme. The convex program can be formulated as following.

- Max: $\sum_{k=1}^{K} w_k \log(1 + x_k)$ (weighted log fairness)
- Subject to: $\sum_{k=1}^{K} \sum_{l=1}^{L} a_{i,j}^{k,l} y_{k,l} \leq C_{i,j}$ for all $(i,j) \in \{Links\}$

$$x_k \leq \sum_{l=1}^{L} y_{k,l} \quad \text{for all} \quad k$$

$$x_k \in [0,\ x_{max}]$$

$$y_{k,l} \in [0,\ y_{max}]$$

$x_{max} = mC_{max}$

$y_{max} = C_{max}$

where m is the maximum links of a single node, and C_{max} is the capacity of the link with the highest capacity.

Define the virtual queues:

$$Q_{i,j}(t+1) = \max\left[Q_{i,j}(t) + \sum_{k=1}^{K}\sum_{l=1}^{L} a_{i,j}^{k,l} y_{k,l} - C_{i,j}, 0\right]$$

$$Z_k(t+1) = \max\left[Z_k(t) + x_k(t) - \sum_{l=1}^{L} y_{k,l}\,(t), 0\right]$$

At every slot, choose $x_k(t), y_{k,l}(t)$ to solve:

Minimize:

$$\begin{gathered} \bullet \quad -V\sum_{k=1}^{K} w_k \log\left(1 + x_k(t)\right) \\ +\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\sum_{k=1}^{K}\sum_{l=1}^{L} a_{i,j}^{k,l} y_{k,l} \\ +\sum_{k=1}^{K} Z_k(t)\left(x_k(t) - \sum_{l=1}^{L} y_{k,l}\,(t)\right) \end{gathered}$$

Use separability:

- $x_k(t)$: $-V\sum_{k=1}^{K} w_k \log\left(1 + x_k(t)\right) + \sum_{k=1}^{K} Z_k(t) x_k(t)$
- $y_{k,l}(t)$: $\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\sum_{k=1}^{K}\sum_{l=1}^{L} a_{i,j}^{k,l} y_{k,l}(t) - \sum_{k=1}^{K} Z_k(t)\sum_{l=1}^{L} y_{k,l}\,(t)$

For each k, minimize:

- $x_k(t)$: $-V w_k \log\left(1 + x_k(t)\right) + Z_k(t) x_k(t)$
- $y_{k,l}(t)$: $\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\sum_{l=1}^{L} a_{i,j}^{k,l} y_{k,l}(t) - Z_k(t)\sum_{l=1}^{L} y_{k,l}\,(t)$

For $x_k(t)$, minimize:

- $x_k(t)$: $-Vw_k \log(1 + x_k(t))\, Z_k(t) x_k(t)$

$$\frac{d}{dx_k(t)} \rightarrow -Vw_k \frac{1}{1+x_k(t)} + Z_k(t) = 0$$

$$x_k(t) = \left[\frac{Vw_k}{Z_k(t)} - 1\right]_0^{x_{max}}$$

For $y_k(t)$, we can separate further:

- $y_{k,l}(t)$: $\sum_{(i,j)\in\{Links\}} Q_{i,j}(t) \sum_{l=1}^{L} a_{i,j}^{k,l} y_{k,l}(t) - Z_k(t) \sum_{l=1}^{L} y_{k,l}(t)$

For each l, minimize:

- $y_{k,l}(t)$: $\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\, a_{i,j}^{k,l} y_{k,l}(t) - Z_k(t) y_{k,l}(t)$
- $y_{k,l}(t)(\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\, a_{i,j}^{k,l} - Z_k(t))$

$$y_{k,l}(t) \in [0,\, y_{max}]$$

If $\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\, a_{i,j}^{k,l} > Z_k(t)$

$$y_{k,l}(t) = 0$$

If $\sum_{(i,j)\in\{Links\}} Q_{i,j}(t)\, a_{i,j}^{k,l} < Z_k(t)$

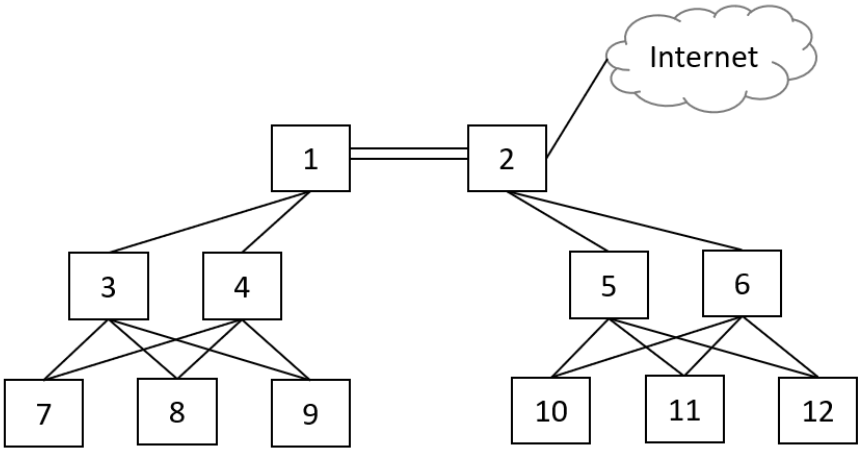
$$y_{k,l}(t) = y_{max}$$

Simulation Results

We then used MATLAB to simulate the DPP algorithm. There are problem parameters and algorithm parameters to be set up. The problem parameters define the network model and flows. The algorithm parameters define the runtime and accuracy of the result.

There are three parameters used to setup the problem. First, the model is defined by links (i, j) between nodes. This is represented by an NxN matrix, set each (i, j) to 1 to represent links. Second, $a_{i,j}^{k,l}$ represents whether a flow is using a certain link. This could be setup by a multi-dimensional array. As defined, $a_{i,j}^{k,l}$ is 1 when the link is active and 0 otherwise. Third, all links are 10 Gbps except for link (1,2) in the 3-layer model and the link to the internet. The exception links have 100 Gbps speed.

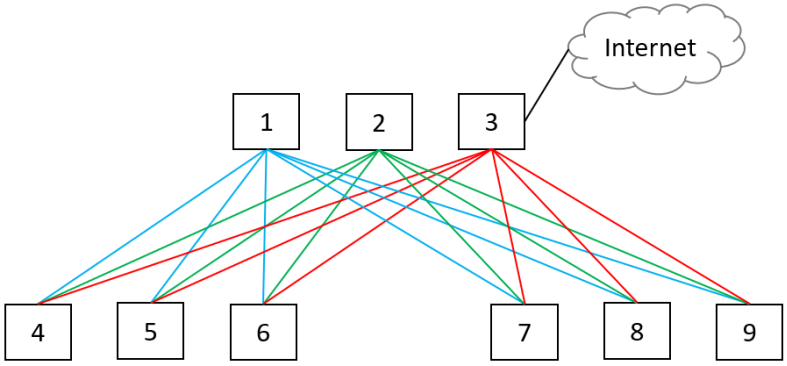
There are two algorithm parameters, the queue size T, and the learning rate V. Queue size represents the number of slots we run the algorithm before averaging over every slot to obtain the final answer. Theory shows that the DPP algorithm approaches optimality when the learning rate V is increased. However, due to the increase in fluctuation, the queue size needs to be increased to reduce to variance. This creates a tradeoff between runtime performance and accuracy. In the simulation, we use V = 1 and T = 10^4.



Core layer

Distribution layer

Access layer (Top of rack switches)



Spine

Leaf (top of rack switches)

Performance Comparison East-West traffic

- Case 1: Single node download, traditional model
- Let 7 download from 10, what is the simulation result for maximum throughput?
- $x_7 = 19.319$ Gbps

- Case 1: Single node download, spine-leaf model
- Let 4 download from 7, what is the simulation result for maximum throughput?
- $x_4 = 29.4985$ Gbps

We can use this first result to validate the DPP algorithm. In the spine and leaf model there are 3 possible flows, 4-1-7, 4-2-7, 4-3-7. There is no link that is sharing its capacity between flows, so the theoretical throughput should be 30 Gbps, which is about equal to the simulation result.

In the 3-Layer model, there are four flows,

7-3-1-2-5-10,

7-3-1-2-6-10,

7-4-1-2-5-10,

7-4-1-2-6-10

Notice that links 3-1, 4-1, 2-5, 2-6 is being shared by two flows. We get each flow throughput to 5 Gbps, combining to a total of 20 Gbps, about equal to the simulation result.

- Case 2: Multiple node download, traditional model
- Let 7 download from 10, 8 from 11, 9 from 12
- what is the simulation result of maximum throughput?
- $x_7 = 6.7197$ Gbps
- $x_8 = 6.7197$
- $x_9 = 6.7197$

- Case 2: Multiple node download, spine-leaf model
- Let 4 download from 7, 5 from 8, 6 from 9
- what is the simulation result of maximum throughput?
- $x_4 = 29.4985$ Gbps
- $x_5 = 29.4985$
- $x_6 = 29.4985$

- Case 3: Multiple node download, traditional model
- Let 7 download from 10, 8 from **10**, 9 from 12
- what is the simulation result of maximum throughput?
- $x_7 = 6.7197$ Gbps
- $x_8 = 6.7197$
- $x_9 = 6.7197$

- Case 3: Multiple node download, spine-leaf model
- Let 4 download from 7, 5 from **7**, 6 from 9
- what is the simulation result of maximum throughput?
- $x_4 = 14.9993$ Gbps
- $x_5 = 14.9993$
- $x_6 = 29.4985$

Performance Comparison North-bound traffic

- Case 1: Single node download, traditional model
- Let 7 download from the Internet
- what is the simulation result of maximum throughput?
- $x_7 = 19.499$ Gbps

- Case 1: Single node download, spine-leaf model
- Let 4 download from the Internet
- what is the simulation result of maximum throughput?
- $x_4 = 9.9995$ Gbps

- Case 2: multiple node download, traditional model
- Let 7 and 8 download from the Internet
- what is the simulation result of maximum throughput?
- $x_7 = 9.4495$ Gbps
- $x_8 = 9.4495$

- Case 2: multiple node download, spine-leaf model
- Let 4 and 5 download from the Internet
- what is the simulation of maximum throughput?
- $x_4 = 9.9995$ Gbps
- $x_5 = 9.9995$

Weighted Performance Comparison

For weighted model we find that V=1 yields inaccurate result. Using V=50 and T=50*10^3 gives result as below.

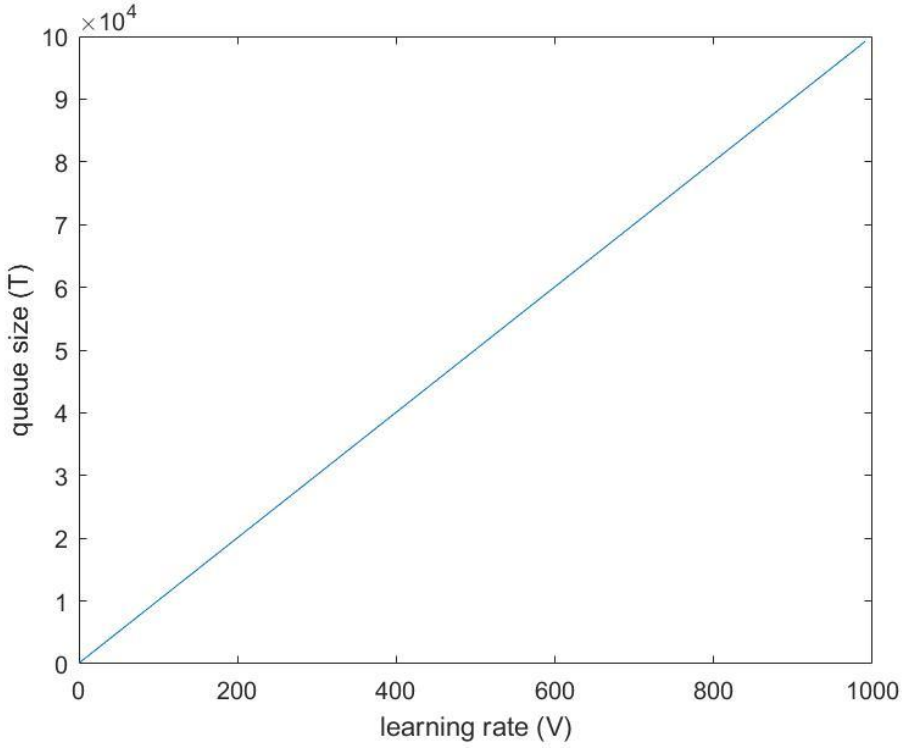
- Multiple node download, traditional model
- Let 7 download from 10, 8 from **10**, 9 from 12
- Set $w_7 = 6$, $w_8 = 1, w_9 = 1$
- $x_7 = 7.39$ Gbps
- $x_8 = 3.5882$
- $x_9 = 3.5882$

- Multiple node download, spine-leaf model
- Let 4 download from 7, 5 from **7**, 6 from 9
- Set $w_4 = 1$, $w_5 = 2, w_6 = 1$
- $x_4 = 12.8066$ Gbps
- $x_5 = 16.8094$
- $x_6 = 29.4189$

The algorithm extends to a weighted fairness model. This can be used to analyze throughput in situations where some nodes are given priority over others.

Exploring Simulation Runtime and Accuracy

The DPP algorithm states that for higher V value, the result will get closer to optimality. Let us explore the utility function $\sum_{k=1}^{K} w_k \log(1 + x_k)$ when V gets large. Here, V is varied from 1 to 1000, increasing by 10 each time. We must be careful to increase the queue size for large V to compensate for the increased variance. T is then set to (V+1)*10^2. To illustrate this, the plot of queue size T vs V is shown below.



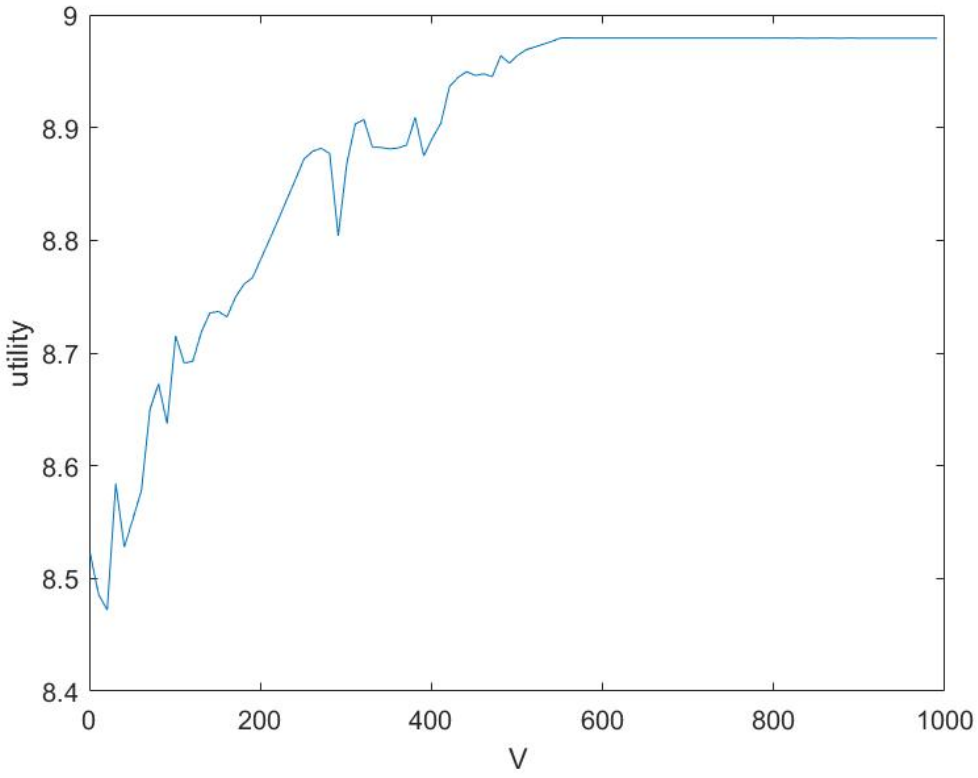
Take case 3 of the East-West traffic comparison for the spine and leaf model. The outcome is

- $x_4 = 14.9993$ Gbps
- $x_5 = 14.9993$
- $x_6 = 29.4985$

We can see that x_4, x_5 approaches 15 and x_6 approaches 30 ($w_k = 1$). Therefore, the optimal utility function is

$$\log(1+15) + \log(1+15) + \log(1+30) = 8.97916$$

Now, let us observe the resulting utility vs V plot.



This aligns with the DPP theory. Higher V yields more accurate result closer to optimality. However, there is a tradeoff that the runtime required must be increased as well. Notice that the runtime required increases linearly with V but the accuracy gained has diminishing returns.

Conclusion

From the simulation results, the new spine and leaf model performs better in East-West throughput, as it was designed to solve this problem. However, the traditional 3-layer model performs better in a single node internet download scenario. This is the main tradeoff when considering which architecture to use in the datacenter. The traditional model solves server hosting or web hosting problems and is suitable when there is high demand for connecting with the internet. The spine and leaf model solves cloud computing problems, where there is high demand for communicating within the same datacenter. Finally, the DPP algorithm derived is not specific to these two network models. It can be used to analyze any topology with defined flow paths and link capacity constraints.

Code:

Project_spineleaf.m

```
clear
N = 9;
n = 3;
C = 10;
m = 100;
k = 3;
l = 3;
xmax = m*C;
ymax = C;
V = 1;
T = (V+1)*10^4;
x = cell(1,k);
for K = 1:k
x{K} = zeros(1,T+1);
end

y = cell(1,T+1);
Q = cell(1,T+1);
Z = cell(1,T+1);
for i=1:T+10
    Q{i} = zeros(N,N);
    Z{i} = zeros(1,k);
    y{i} = zeros(k,l);
end


links = zeros(N,N);
links(1,4)=1;
links(1,5)=1;
links(1,6)=1;
links(1,7)=1;
links(1,8)=1;
links(1,9)=1;
links(2,4)=1;
links(2,5)=1;
links(2,6)=1;
links(2,7)=1;
links(2,8)=1;
links(2,9)=1;
links(3,4)=1;
links(3,5)=1;
links(3,6)=1;
links(3,7)=1;
links(3,8)=1;
links(3,9)=1;

a = cell(k,l)

for K = 1:k
    for L = 1:l
        a{K,L} = zeros(N,N);
    end
end

a{1,1}(1,4) = 1;
a{1,1}(1,7) = 1;
a{1,2}(2,4) = 1;
```

```
a{1,2}(2,7) = 1;
a{1,3}(3,4) = 1;
a{1,3}(3,7) = 1;
a{2,1}(1,5) = 1;
a{2,1}(1,7) = 1;
a{2,2}(2,5) = 1;
a{2,2}(2,7) = 1;
a{2,3}(3,5) = 1;
a{2,3}(3,7) = 1;
a{3,1}(1,6) = 1;
a{3,1}(1,9) = 1;
a{3,2}(2,6) = 1;
a{3,2}(2,9) = 1;
a{3,3}(3,6) = 1;
a{3,3}(3,9) = 1;


for t = 1:T
    for i = 1:N
        for j = 1:N
            if links(i,j) == 1
                sum2 = 0;
                for K = 1:k
                    for L = 1:l
                        sum2 = sum2 + a{K,L}(i,j)*y{t}(K,L);
                    end
                end
                sum2;
                Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-C, 0);
            end
        end
    end
    for K = 1:k
        sum3 = 0;
        for L = 1:l
            sum3 = sum3 + y{t}(K,L);
            L;
            sum3;
        end
        Z{t+1}(K) = max(Z{t}(K)-sum3+x{K}(t),0);
        temp = (V/Z{t}(K)) - 1;
        if temp < 0
            temp = 0;
        end
        if temp > xmax
            temp = xmax;
        end
        x{K}(t+1) = temp;

        for L = 1:l
            sum4 = sum(sum(Q{t}.*a{K,L}));

            if sum4 > Z{t}(K)
                y{t+1}(K,L) = 0;
            else
                y{t+1}(K,L) = ymax;
            end
        end

    end

end
avg = zeros(k,l);
for t = 1:T
```

```
    for K = 1:k
        for L = 1:l
            avg(K,L) = avg(K,L) + y{t}(K,L);
        end
    end
end

avg=avg/T

xbar1 = mean(x{1})
xbar2 = mean(x{2})
xbar3 = mean(x{3})
%util = log(1+xbar1)+log(1+xbar2)+log(1+xbar3)
```

Project_3layer.m

```
clear
N = 12;
n = 3;
C = 10;
bigC = 100;
m=120;
k = 3;
l = 4;
xmax = m*C;
ymax = bigC;
V = 1;
T = (V+1)*10^4;
x = cell(1,k);

for K = 1:k
x{K} = zeros(1,T+1);
end

y = cell(1,T+1);
Q = cell(1,T+1);
Z = cell(1,T+1);

for i=1:T+10
    Q{i} = zeros(N,N);
    Z{i} = zeros(1,k);
    y{i} = zeros(k,l);
end



links = zeros(N,N);
links(1,2)=1;
links(1,3)=1;
links(1,4)=1;
links(2,5)=1;
links(2,6)=1;
links(3,7)=1;
links(3,8)=1;
links(3,9)=1;
links(4,7)=1;
links(4,8)=1;
links(4,9)=1;
links(5,10)=1;
links(5,11)=1;
links(5,12)=1;
```

```
links(6,10)=1;
links(6,11)=1;
links(6,12)=1;

a = cell(k,l)

for K = 1:k
    for L = 1:l
        a{K,L} = zeros(N,N);
    end
end

a{1,1}(3,7) = 1;
a{1,1}(1,3) = 1;
a{1,1}(1,2) = 1;
a{1,1}(2,5) = 1;
a{1,1}(5,10) = 1;
a{1,2}(4,7) = 1;
a{1,2}(1,4) = 1;
a{1,2}(1,2) = 1;
a{1,2}(2,5) = 1;
a{1,2}(5,10) = 1;
a{1,3}(3,7) = 1;
a{1,3}(1,3) = 1;
a{1,3}(1,2) = 1;
a{1,3}(2,6) = 1;
a{1,3}(6,10) = 1;
a{1,4}(4,7) = 1;
a{1,4}(1,4) = 1;
a{1,4}(1,2) = 1;
a{1,4}(2,6) = 1;
a{1,4}(6,10) = 1;

a{2,1}(3,8) = 1;
a{2,1}(1,3) = 1;
a{2,1}(1,2) = 1;
a{2,1}(2,5) = 1;
a{2,1}(5,10) = 1;
a{2,2}(4,8) = 1;
a{2,2}(1,4) = 1;
a{2,2}(1,2) = 1;
a{2,2}(2,5) = 1;
a{2,2}(5,10) = 1;
a{2,3}(3,8) = 1;
a{2,3}(1,3) = 1;
a{2,3}(1,2) = 1;
a{2,3}(2,6) = 1;
a{2,3}(6,10) = 1;
a{2,4}(4,8) = 1;
a{2,4}(1,4) = 1;
a{2,4}(1,2) = 1;
a{2,4}(2,6) = 1;
a{2,4}(6,10) = 1;

a{3,1}(3,9) = 1;
a{3,1}(1,3) = 1;
a{3,1}(1,2) = 1;
a{3,1}(2,5) = 1;
a{3,1}(5,11) = 1;
a{3,2}(4,9) = 1;
a{3,2}(1,4) = 1;
a{3,2}(1,2) = 1;
a{3,2}(2,5) = 1;
```

```
a{3,2}(5,11) = 1;
a{3,3}(3,9) = 1;
a{3,3}(1,3) = 1;
a{3,3}(1,2) = 1;
a{3,3}(2,6) = 1;
a{3,3}(6,11) = 1;
a{3,4}(4,9) = 1;
a{3,4}(1,4) = 1;
a{3,4}(1,2) = 1;
a{3,4}(2,6) = 1;
a{3,4}(6,11) = 1;


for t = 1:T
    for i = 1:N
        for j = 1:N
            if links(i,j) == 1
                sum2 = 0;
                for K = 1:k
                    for L = 1:l
                        sum2 = sum2 + a{K,L}(i,j)*y{t}(K,L);
                    end
                end
                sum2;
                if i == 1 && j == 2
                    Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-bigC, 0);
                else
                    Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-C, 0);
                end
            end
        end
    end
    for K = 1:k
        sum3 = 0;
        for L = 1:l
            sum3 = sum3 + y{t}(K,L);
            L;
            sum3;
        end
        Z{t+1}(K) = max(Z{t}(K)-sum3+x{K}(t),0);
        temp = (V/Z{t}(K)) - 1;
        if temp < 0
            temp = 0;
        end
        if temp > xmax
            temp = xmax;
        end
        x{K}(t+1) = temp;

        for L = 1:l

            sum4 = sum(sum(Q{t}.*a{K,L}));

            if sum4 > Z{t}(K)
                y{t+1}(K,L) = 0;
            else
                y{t+1}(K,L) = ymax;
            end
        end

    end

end
```

```
avg = zeros(k,l);
for t = 1:T
    for K = 1:k
        for L = 1:l
            avg(K,L) = avg(K,L) + y{t}(K,L);
        end
    end
end

avg=avg/T

xbar1 = mean(x{1})
xbar2 = mean(x{2})
xbar3 = mean(x{3})
%util = log(1+xbar1)+log(1+xbar2)+log(1+xbar3)
```

Project_spineleaf_weighted.m

```
clear
N = 9;
n = 3;
C = 10;
m = 100;
k = 3;
l = 3;
xmax = m*C;
ymax = C;
V = 50;
T = (V+1)*10^3;
x = cell(1,k);
for K = 1:k
x{K} = zeros(1,T+1);
end
w = ones(1,k);
w(1) = 6;

y = cell(1,T+1);
Q = cell(1,T+1);
Z = cell(1,T+1);
for i=1:T+10
    Q{i} = zeros(N,N);
    Z{i} = zeros(1,k);
    y{i} = zeros(k,l);
end



links = zeros(N,N);
links(1,4)=1;
links(1,5)=1;
links(1,6)=1;
links(1,7)=1;
links(1,8)=1;
links(1,9)=1;
links(2,4)=1;
links(2,5)=1;
links(2,6)=1;
links(2,7)=1;
links(2,8)=1;
links(2,9)=1;
links(3,4)=1;
links(3,5)=1;
links(3,6)=1;
```

```
links(3,7)=1;
links(3,8)=1;
links(3,9)=1;

a = cell(k,l)

for K = 1:k
    for L = 1:l
        a{K,L} = zeros(N,N);
    end
end

a{1,1}(1,4) = 1;
a{1,1}(1,7) = 1;
a{1,2}(2,4) = 1;
a{1,2}(2,7) = 1;
a{1,3}(3,4) = 1;
a{1,3}(3,7) = 1;
a{2,1}(1,5) = 1;
a{2,1}(1,7) = 1;
a{2,2}(2,5) = 1;
a{2,2}(2,7) = 1;
a{2,3}(3,5) = 1;
a{2,3}(3,7) = 1;
a{3,1}(1,6) = 1;
a{3,1}(1,9) = 1;
a{3,2}(2,6) = 1;
a{3,2}(2,9) = 1;
a{3,3}(3,6) = 1;
a{3,3}(3,9) = 1;



for t = 1:T
    for i = 1:N
        for j = 1:N
            if links(i,j) == 1
                sum2 = 0;
                for K = 1:k
                    for L = 1:l
                        sum2 = sum2 + a{K,L}(i,j)*y{t}(K,L);
                    end
                end
                sum2;
                Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-C, 0);
            end
        end
    end
    for K = 1:k
        sum3 = 0;
        for L = 1:l
            sum3 = sum3 + y{t}(K,L);
            L;
            sum3;
        end
        Z{t+1}(K) = max(Z{t}(K)-sum3+x{K}(t),0);
        temp = (V*w(K))/Z{t}(K) - 1;
        if temp < 0
            temp = 0;
        end
        if temp > xmax
            temp = xmax;
```

```
        end
        x{K}(t+1) = temp;

        for L = 1:l
            sum4 = sum(sum(Q{t}.*a{K,L}));

            if sum4 > Z{t}(K)
                y{t+1}(K,L) = 0;
            else
                y{t+1}(K,L) = ymax;
            end
        end

    end

end
avg = zeros(k,l);
for t = 1:T
    for K = 1:k
        for L = 1:l
            avg(K,L) = avg(K,L) + y{t}(K,L);
        end
    end
end

avg=avg/T

xbar1 = mean(x{1})
xbar2 = mean(x{2})
xbar3 = mean(x{3})
%util = log(1+xbar1)+log(1+xbar2)+log(1+xbar3)
```

Project_3layer_weighted.m

```
clear
N = 12;
n = 3;
C = 10;
bigC = 100;
m=50;
k = 3;
l = 4;
xmax = m*C;
ymax = bigC;
V = 50;
T = (V+1)*10^3;
x = cell(1,k);

for K = 1:k
x{K} = zeros(1,T+1);
end
w = ones(1,k);
w(1) = 6;

y = cell(1,T+1);
Q = cell(1,T+1);
Z = cell(1,T+1);
for i=1:T+10
    Q{i} = zeros(N,N);
    Z{i} = zeros(1,k);
    y{i} = zeros(k,l);
end
```

```
links = zeros(N,N);
links(1,2)=1;
links(1,3)=1;
links(1,4)=1;
links(2,5)=1;
links(2,6)=1;
links(3,7)=1;
links(3,8)=1;
links(3,9)=1;
links(4,7)=1;
links(4,8)=1;
links(4,9)=1;
links(5,10)=1;
links(5,11)=1;
links(5,12)=1;
links(6,10)=1;
links(6,11)=1;
links(6,12)=1;

a = cell(k,l)

for K = 1:k
    for L = 1:l
        a{K,L} = zeros(N,N);
    end
end

a{1,1}(3,7) = 1;
a{1,1}(1,3) = 1;
a{1,1}(1,2) = 1;
a{1,1}(2,5) = 1;
a{1,1}(5,10) = 1;
a{1,2}(4,7) = 1;
a{1,2}(1,4) = 1;
a{1,2}(1,2) = 1;
a{1,2}(2,5) = 1;
a{1,2}(5,10) = 1;
a{1,3}(3,7) = 1;
a{1,3}(1,3) = 1;
a{1,3}(1,2) = 1;
a{1,3}(2,6) = 1;
a{1,3}(6,10) = 1;
a{1,4}(4,7) = 1;
a{1,4}(1,4) = 1;
a{1,4}(1,2) = 1;
a{1,4}(2,6) = 1;
a{1,4}(6,10) = 1;

a{2,1}(3,8) = 1;
a{2,1}(1,3) = 1;
a{2,1}(1,2) = 1;
a{2,1}(2,5) = 1;
a{2,1}(5,10) = 1;
a{2,2}(4,8) = 1;
a{2,2}(1,4) = 1;
a{2,2}(1,2) = 1;
a{2,2}(2,5) = 1;
a{2,2}(5,10) = 1;
a{2,3}(3,8) = 1;
a{2,3}(1,3) = 1;
```

```
a{2,3}(1,2) = 1;
a{2,3}(2,6) = 1;
a{2,3}(6,10) = 1;
a{2,4}(4,8) = 1;
a{2,4}(1,4) = 1;
a{2,4}(1,2) = 1;
a{2,4}(2,6) = 1;
a{2,4}(6,10) = 1;

a{3,1}(3,9) = 1;
a{3,1}(1,3) = 1;
a{3,1}(1,2) = 1;
a{3,1}(2,5) = 1;
a{3,1}(5,11) = 1;
a{3,2}(4,9) = 1;
a{3,2}(1,4) = 1;
a{3,2}(1,2) = 1;
a{3,2}(2,5) = 1;
a{3,2}(5,11) = 1;
a{3,3}(3,9) = 1;
a{3,3}(1,3) = 1;
a{3,3}(1,2) = 1;
a{3,3}(2,6) = 1;
a{3,3}(6,11) = 1;
a{3,4}(4,9) = 1;
a{3,4}(1,4) = 1;
a{3,4}(1,2) = 1;
a{3,4}(2,6) = 1;
a{3,4}(6,11) = 1;
```

```
for t = 1:T
    for i = 1:N
        for j = 1:N
            if links(i,j) == 1
                sum2 = 0;
                for K = 1:k
                    for L = 1:l
                        sum2 = sum2 + a{K,L}(i,j)*y{t}(K,L);
                    end
                end
                sum2;
                if i == 1 && j == 2
                    Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-bigC, 0);
                else
                    Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-C, 0);
                end
            end
        end
    end
    for K = 1:k
        sum3 = 0;
        for L = 1:l
            sum3 = sum3 + y{t}(K,L);
            L;
            sum3;
        end
        Z{t+1}(K) = max(Z{t}(K)-sum3+x{K}(t),0);
        temp = ((V*w(K))/Z{t}(K)) - 1;
        if temp < 0
```

```
            temp = 0;
        end
        if temp > xmax
            temp = xmax;
        end
        x{K}(t+1) = temp;

        for L = 1:l

            sum4 = sum(sum(Q{t}.*a{K,L}));

            if sum4 > Z{t}(K)
                y{t+1}(K,L) = 0;
            else
                y{t+1}(K,L) = ymax;
            end
        end

    end

end
avg = zeros(k,l);
for t = 1:T
    for K = 1:k
        for L = 1:l
            avg(K,L) = avg(K,L) + y{t}(K,L);
        end
    end
end

avg=avg/T

xbar1 = mean(x{1})
xbar2 = mean(x{2})
xbar3 = mean(x{3})
%util = log(1+xbar1)+log(1+xbar2)+log(1+xbar3)
```

Project_performance.m

```
function [util] = performance(v)
    N = 9;
    n = 3;
    C = 10;
    m = 10;
    k = 3;
```

```
l = 3;
xmax = m*C;
ymax = C;
V = v;
T = (V+1)*10^2;
x = cell(1,k);
for K = 1:k
x{K} = zeros(1,T+1);
end

y = cell(1,T+1);
Q = cell(1,T+1);
Z = cell(1,T+1);

for i=1:T+10
    Q{i} = zeros(N,N);
    Z{i} = zeros(1,k);
    y{i} = zeros(k,l);
end



links = zeros(N,N);
links(1,4)=1;
links(1,5)=1;
links(1,6)=1;
links(1,7)=1;
links(1,8)=1;
links(1,9)=1;
links(2,4)=1;
links(2,5)=1;
links(2,6)=1;
links(2,7)=1;
links(2,8)=1;
links(2,9)=1;
links(3,4)=1;
links(3,5)=1;
links(3,6)=1;
links(3,7)=1;
links(3,8)=1;
links(3,9)=1;

a = cell(k,l);

for K = 1:k
    for L = 1:l
        a{K,L} = zeros(N,N);
    end
end

a{1,1}(1,4) = 1;
a{1,1}(1,7) = 1;
a{1,2}(2,4) = 1;
a{1,2}(2,7) = 1;
a{1,3}(3,4) = 1;
a{1,3}(3,7) = 1;
a{2,1}(1,5) = 1;
a{2,1}(1,7) = 1;
a{2,2}(2,5) = 1;
a{2,2}(2,7) = 1;
a{2,3}(3,5) = 1;
a{2,3}(3,7) = 1;
a{3,1}(1,6) = 1;
```

```
a{3,1}(1,9) = 1;
a{3,2}(2,6) = 1;
a{3,2}(2,9) = 1;
a{3,3}(3,6) = 1;
a{3,3}(3,9) = 1;



for t = 1:T
    for i = 1:N
        for j = 1:N
            if links(i,j) == 1
                sum2 = 0;
                for K = 1:k
                    for L = 1:l
                        sum2 = sum2 + a{K,L}(i,j)*y{t}(K,L);
                    end
                end
                sum2;
                Q{t+1}(i,j) = max(Q{t}(i,j)+sum2-C, 0);
            end
        end
    end
    for K = 1:k
        sum3 = 0;
        for L = 1:l
            sum3 = sum3 + y{t}(K,L);
            L;
            sum3;
        end
        Z{t+1}(K) = max(Z{t}(K)-sum3+x{K}(t),0);
        temp = (V/Z{t}(K)) - 1;
        if temp < 0
            temp = 0;
        end
        if temp > xmax
            temp = xmax;
        end
        x{K}(t+1) = temp;

        for L = 1:l
            sum4 = sum(sum(Q{t}.*a{K,L}));

            if sum4 > Z{t}(K)
                y{t+1}(K,L) = 0;
            else
                y{t+1}(K,L) = ymax;
            end
        end

    end

end
avg = zeros(k,l);
for t = 1:T
    for K = 1:k
        for L = 1:l
            avg(K,L) = avg(K,L) + y{t}(K,L);
        end
    end
end

avg=avg/T;
```

```
    xbar1 = mean(x{1});
    xbar2 = mean(x{2});
    xbar3 = mean(x{3});
    util = log(1+xbar1)+log(1+xbar2)+log(1+xbar3);
end
```

Project_run_performance.m

```
y = zeros(1,100);
x = 1:10:1000;

for i=1:100
    y(i) = project_performance(x(i));

end

y
plot(x,y)

T = (x+1)*10^2;
plot(x,T)
xlabel('learning rate (V)')
ylabel('queue size (T)')
```