Service Function Chain Placement in Cloud Data Center Networks: a Cooperative Multi-Agent Reinforcement Learning Approach

Lynn Gao, University of California, Irvine

Yutian Chen, California State University, Long Beach

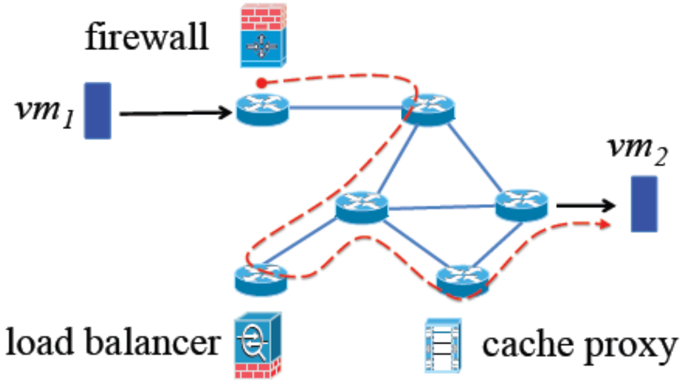
Bin Tang, California State University, Dominguez Hills

Outline

- Introduction
    - Service Function Chain (SFC)
    - SFC Placement Problem
- Algorithms
    - Multi-agent Reinforcement Learning (MARL)
    - Dynamic Programming (DP)
    - Optimal Exhaustive (Optimal)
- Performance Evaluation
- Conclusions and Future Work

Service Function Chaining (SFC)

- An SFC, consisting of a sequence of virtual network functions (VFNs) (i.e., firewalls and load balancers), is an effective service provisioning technique in cloud data centers
- Virtual machine (VM) cloud traffic traverses the chain of VNFs in a order to achieve security and performance guarantees
- An example of an SFC of three VNFs:
  - First, the firewall blocks malicious traffic
  - Next, the load balancer avoids network congestion
  - Finally, the cache proxy caches network packets for data access by other cloud



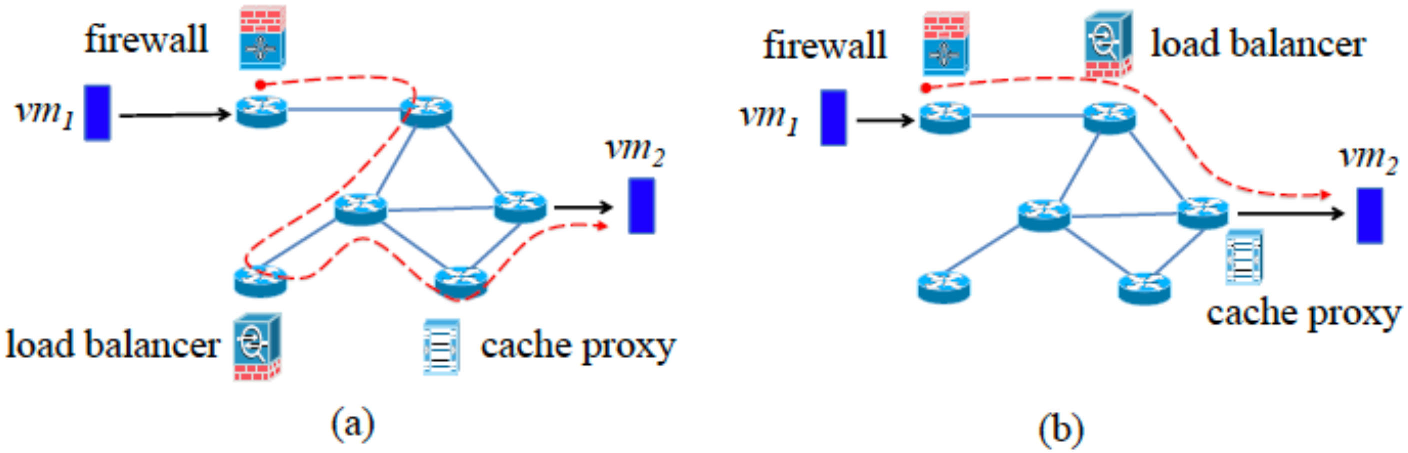
SFC Placement Problem

- As cloud network resources such as bandwidth and energy are limited in a cloud data center network, one important task for the cloud operator is to install the VNFs at the right locations inside the network to optimize the cloud traffic or user-perceived VM communication delay.

- SFC Placement Problem: Given a VM flow in the cloud data center and an SFC, how to place the VNFs of the SFC inside the cloud data center to minimize the communication cost (i.e., network traffic or delay) of the VM flow?

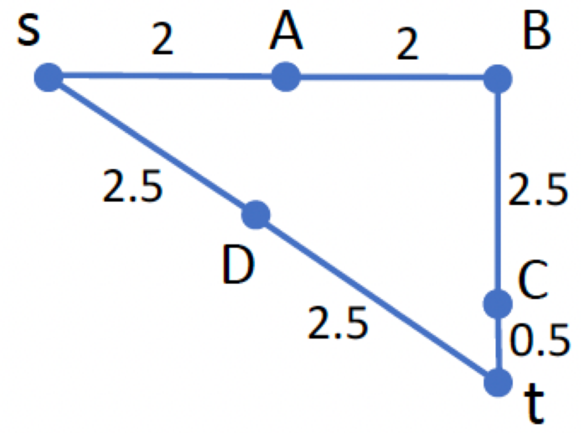
An Example of SFC Placement

- A small cloud data center of six switches
- One VM communication flow (vm_1, vm_2)
- SFC placement (a): traversing six switches, with six network hops
- SFC placement (b): traversing three switches, with two network hops
- (b) is more network-efficient than (a)



SFC Placement is Equivalent to k-Stroll Problem

- **k-Stroll Problem**
  - Given
    - a weighted undirected graph
    - source s and destination t,
    - an integer k
  - Goal: find an s-t path or walk (i.e., a stroll) of minimum length that visits at least k distinct nodes excluding s and t.
- k-stroll is NP-hard

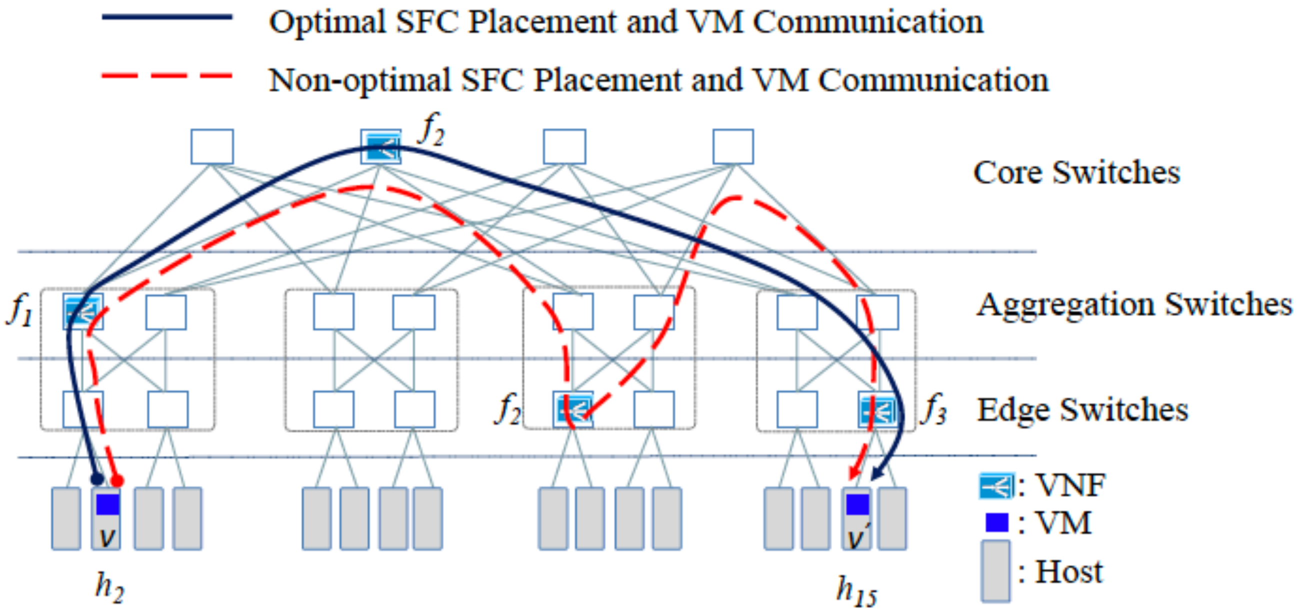


Optimal 2-stroll between s and t: s, D, t, C, t with cost 6

SFC Placement in Fat-tree Data Centers

- One pair of communicating VMs (v, v'), v is at host $s(v)$ and v' at $s(v')$
- SFC with n VNFs: f_1, f_2, …, *and* f_n
- *Place the* n VNFs to *minimize communication cost:*

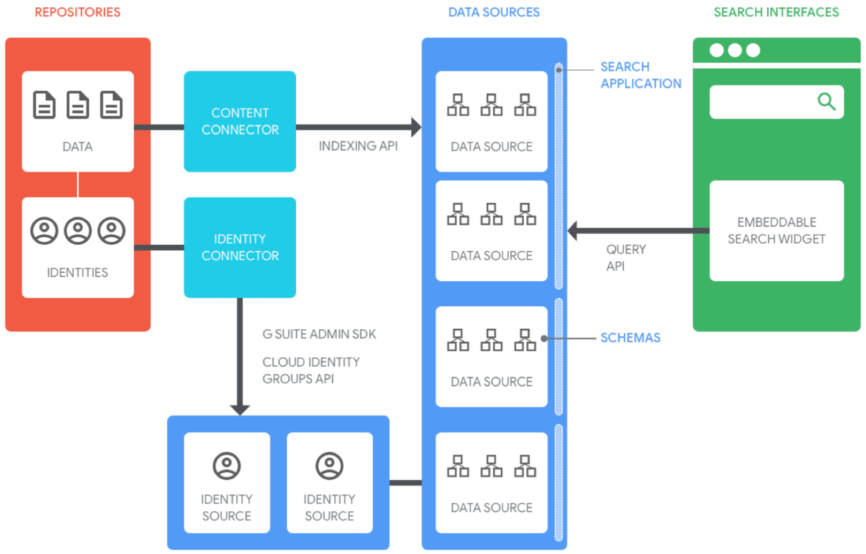
$$C_c(p) = \sum_{j=1}^{n-1} c\big(p(j), p(j+1)\big) + \Big(c\big(s(v_i), p(1)\big) + c\big(p(n), s(v_i')\big)\Big).$$



SFC Placement in Fat-tree Data Centers

- One pair of communicating VMs (v, v'), v is at host $s(v)$ and v' at $s(v')$
- SFC with n VNFs: f_1, f_2, …, *and* f_n
- *Place the* n VNFs to *minimize communication cost:*

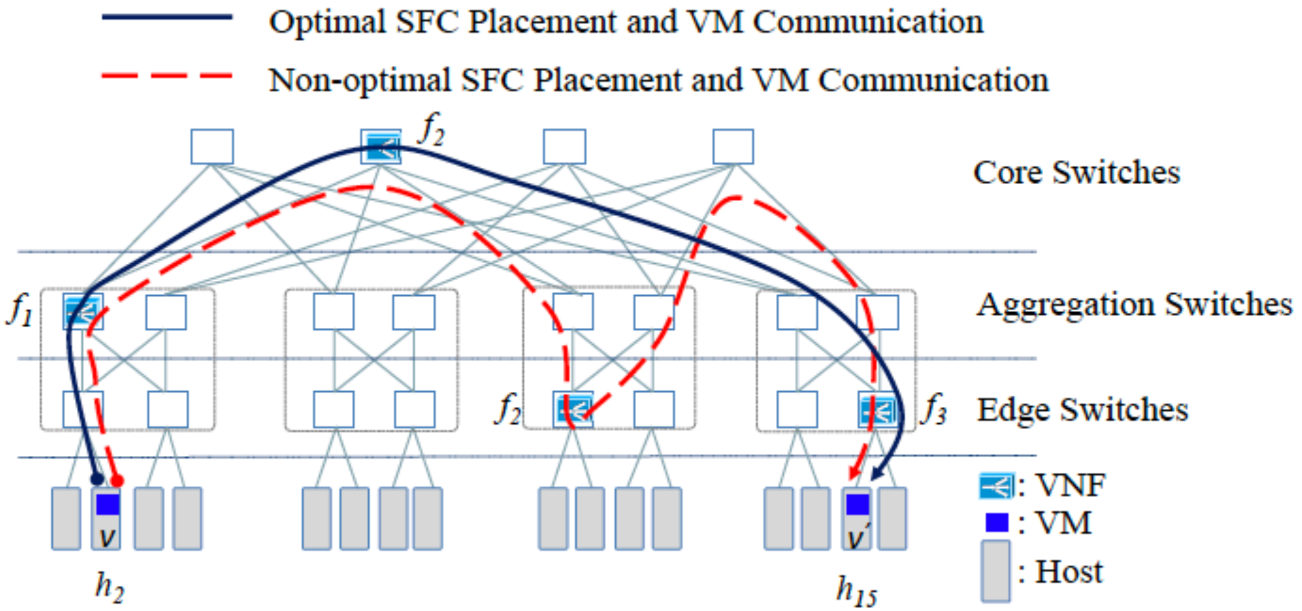
$$C_c(p) = \sum_{j=1}^{n-1} c\big(p(j), p(j+1)\big) + \Big(c\big(s(v_i), p(1)\big) + c\big(p(n), s(v_i')\big)\Big).$$



SFC Placement in Fat-tree Data Centers

- One pair of communicating VMs (v, v'), v is at host $s(v)$ and v' at $s(v')$
- SFC with n VNFs: f_1, f_2, …, *and* f_n
- *Place the* n VNFs to *minimize communication cost:*

$$C_c(p) = \sum_{j=1}^{n-1} c\big(p(j), p(j+1)\big) + \Big(c\big(s(v_i), p(1)\big) + c\big(p(n), s(v_i')\big)\Big).$$



Algorithms for SFC Placement

- MARL algorithm
- DP algorithm
- Optimal algorithm

MARL Algorithm

- Uses reinforcement learning and Q-learning
  - Finite set of states S
  - Finite set of actions A
  - State transition function t
  - Reward function r
  - Value based methods
  - Agents explore the world by taking actions based on the state of the world

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot [r(s,a) + \gamma \cdot \mathrm{max}_b Q(t,b)]$$

MARL Algorithm - Action Selection Rule

- Exploitation: move to node t where

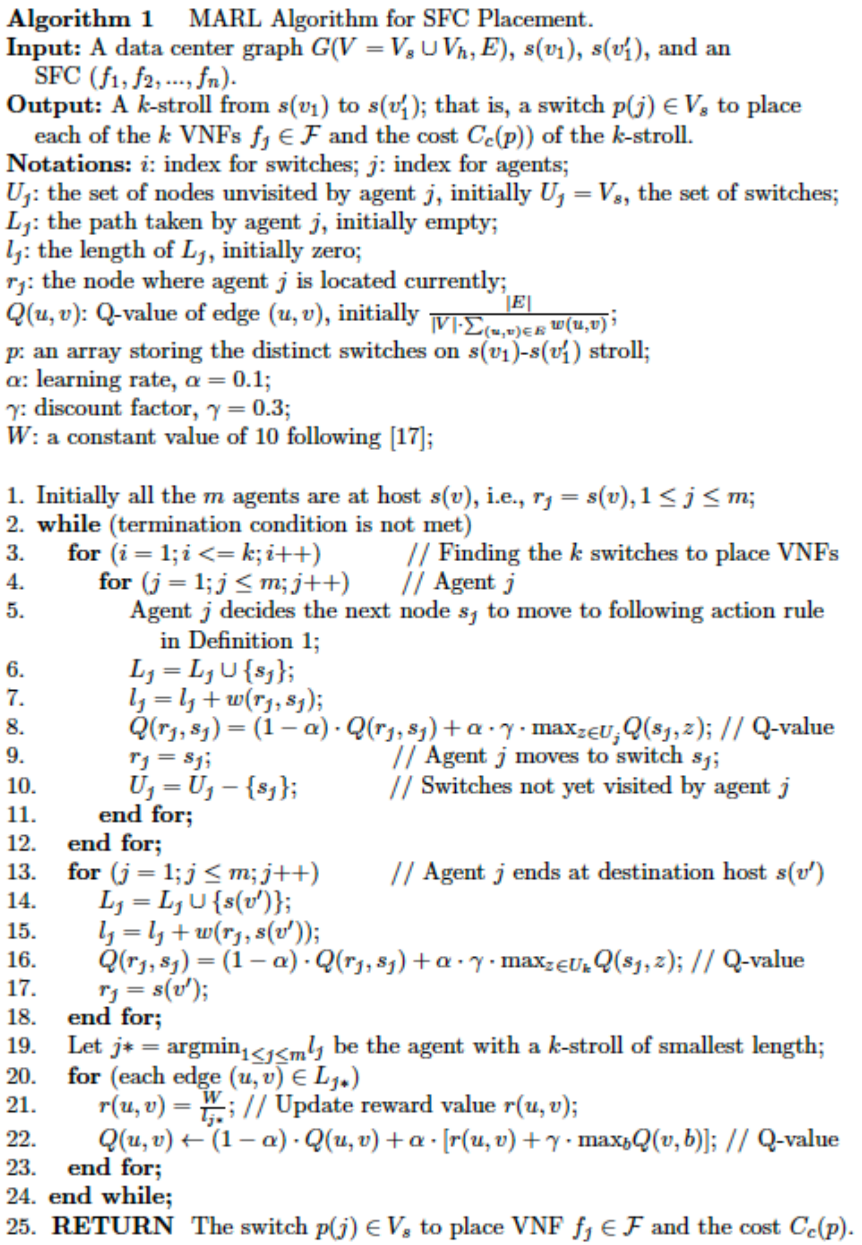
$$t = argmax_{u \in U}\{\frac{[Q(s,u)]^{\delta}}{[w(s,u)]^{\beta}}\}$$

- Exploration: move to node t following distribution

$$p(s,t) = \frac{[Q(s,t)]^{\delta}/[w(s,t)]^{\beta}}{\sum_{u \in U}[Q(s,u)]^{\delta}/[w(s,u)]^{\beta}}$$

MARL Algorithm

- All the m agents start at host $s(v)$
- Takes place in episodes (iterations)
- Each episode has two stages:
  - Stage 1: each agent finds its own k-stroll following action rules
  - Stage 2: finds the smallest cost k-stroll and updates reward and Q-value for edges in this k-stroll
- Terminate after specified number of iterations or within proximity to compared DP or Optimal algorithms
- Time complexity: $O(Nmk)$
  - N: number of episodes
  - M: number of agents
  - k: number of VNFs



Algorithm 1 MARL Algorithm for SFC Placement.

Input: A data center graph $G(V = V_s \cup V_h, E)$, $s(v_1)$, $s(v_1')$, and an SFC $(f_1, f_2, ..., f_n)$.

Output: A k-stroll from $s(v_1)$ to $s(v_1')$; that is, a switch $p(j) \in V_s$ to place each of the k VNFs $f_j \in \mathcal{F}$ and the cost $C_c(p)$ of the k-stroll.

Notations: i: index for switches; j: index for agents;
U_j: the set of nodes unvisited by agent j, initially $U_j = V_s$, the set of switches;
L_j: the path taken by agent j, initially empty;
l_j: the length of L_j, initially zero;
r_j: the node where agent j is located currently;
$Q(u,v)$: Q-value of edge (u,v), initially $\frac{|E|}{|V| \cdot \sum_{(u,v) \in E} w(u,v)}$;
p: an array storing the distinct switches on $s(v_1)$-$s(v_1')$ stroll;
α: learning rate, $\alpha = 0.1$;
γ: discount factor, $\gamma = 0.3$;
W: a constant value of 10 following [17];

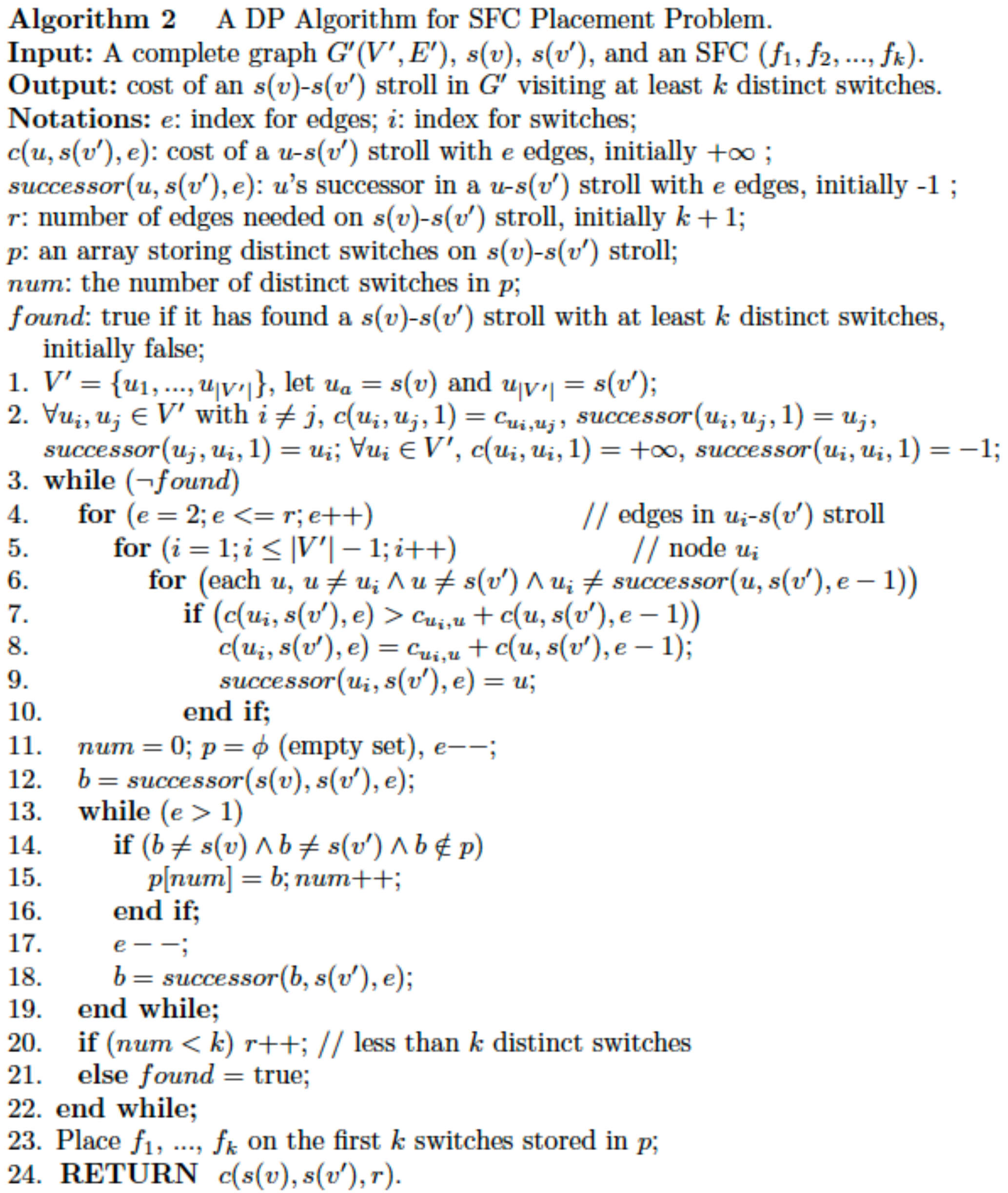
```
Initially all the $m$ agents are at host $s(v)$, i.e., $r_j = s(v), 1 \leq j \leq m$;
while (termination condition is not met)
    for ($i = 1; i <= k; i{+}{+}$)          // Finding the $k$ switches to place VNFs
        for ($j = 1; j \leq m; j{+}{+}$)     // Agent $j$
            Agent $j$ decides the next node $s_j$ to move to following action rule
              in Definition 1;
            $L_j = L_j \cup \{s_j\}$;
            $l_j = l_j + w(r_j, s_j)$;
            $Q(r_j, s_j) = (1-\alpha) \cdot Q(r_j, s_j) + \alpha \cdot \gamma \cdot \max_{z \in U_j} Q(s_j, z)$; // Q-value
            $r_j = s_j$;                     // Agent $j$ moves to switch $s_j$;
            $U_j = U_j - \{s_j\}$;           // Switches not yet visited by agent $j$
        end for;
    end for;
    for ($j = 1; j \leq m; j{+}{+}$)         // Agent $j$ ends at destination host $s(v')$
        $L_j = L_j \cup \{s(v')\}$;
        $l_j = l_j + w(r_j, s(v'))$;
        $Q(r_j, s_j) = (1-\alpha) \cdot Q(r_j, s_j) + \alpha \cdot \gamma \cdot \max_{z \in U_k} Q(s_j, z)$; // Q-value
        $r_j = s(v')$;
    end for;
    Let $j* = \text{argmin}_{1 \leq j \leq m} l_j$ be the agent with a $k$-stroll of smallest length;
    for (each edge $(u,v) \in L_{j*}$)
        $r(u,v) = \frac{W}{l_{j*}}$; // Update reward value $r(u,v)$;
        $Q(u,v) \leftarrow (1-\alpha) \cdot Q(u,v) + \alpha \cdot [r(u,v) + \gamma \cdot \max_b Q(v,b)]$; // Q-value
    end for;
end while;
RETURN The switch $p(j) \in V_s$ to place VNF $f_j \in \mathcal{F}$ and the cost $C_c(p)$.
```

Stage 1

Stage 2

Dynamic Programming Algorithm

- Based on the observation that although finding the shortest stroll visiting k distinct nodes is NP-hard, finding the shortest stroll of k edges can be solved optimally and efficiently using DP
- Time complexity: $O(k|V|^4)$



Algorithm 2 A DP Algorithm for SFC Placement Problem.

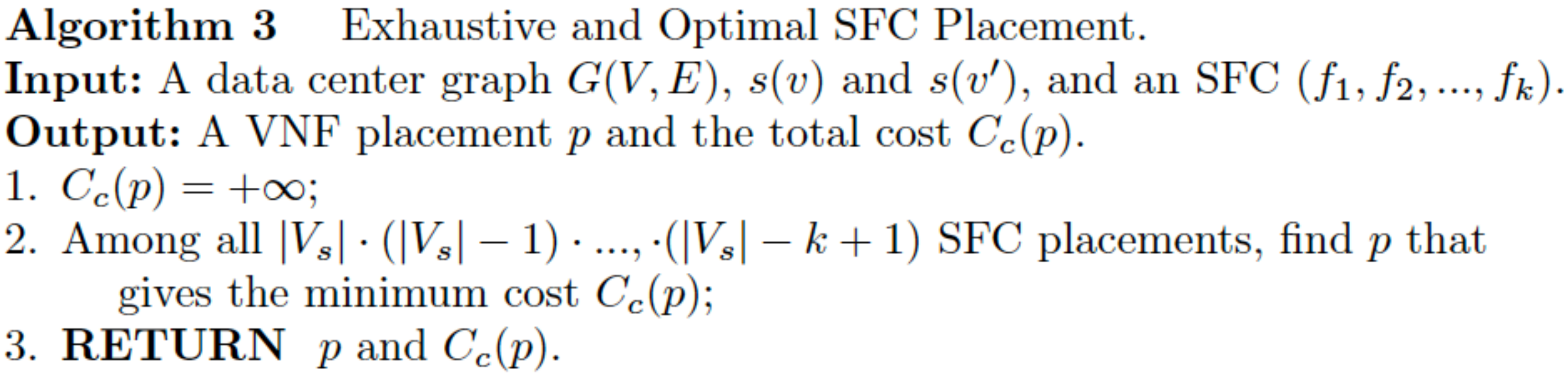
Input: A complete graph $G'(V', E')$, $s(v)$, $s(v')$, and an SFC $(f_1, f_2, ..., f_k)$.

Output: cost of an $s(v)$-$s(v')$ stroll in G' visiting at least k distinct switches.

Notations: e: index for edges; i: index for switches;
$c(u, s(v'), e)$: cost of a u-$s(v')$ stroll with e edges, initially $+\infty$;
$successor(u, s(v'), e)$: u's successor in a u-$s(v')$ stroll with e edges, initially -1 ;
r: number of edges needed on $s(v)$-$s(v')$ stroll, initially $k+1$;
p: an array storing distinct switches on $s(v)$-$s(v')$ stroll;
num: the number of distinct switches in p;
$found$: true if it has found a $s(v)$-$s(v')$ stroll with at least k distinct switches, initially false;

```
$V' = \{u_1, ..., u_{|V'|}\}$, let $u_a = s(v)$ and $u_{|V'|} = s(v')$;
$\forall u_i, u_j \in V'$ with $i \neq j$, $c(u_i, u_j, 1) = c_{u_i,u_j}$, $successor(u_i, u_j, 1) = u_j$,
    $successor(u_j, u_i, 1) = u_i$; $\forall u_i \in V'$, $c(u_i, u_i, 1) = +\infty$, $successor(u_i, u_i, 1) = -1$;
while $(\neg found)$
    for $(e = 2; e <= r; e{+}{+})$                          // edges in $u_i$-$s(v')$ stroll
        for $(i = 1; i \leq |V'| - 1; i{+}{+})$                   // node $u_i$
            for (each $u$, $u \neq u_i \wedge u \neq s(v') \wedge u_i \neq successor(u, s(v'), e-1)$)
                if $(c(u_i, s(v'), e) > c_{u_i,u} + c(u, s(v'), e-1))$
                    $c(u_i, s(v'), e) = c_{u_i,u} + c(u, s(v'), e-1)$;
                    $successor(u_i, s(v'), e) = u$;
                end if;
    $num = 0$; $p = \phi$ (empty set), $e{-}{-}$;
    $b = successor(s(v), s(v'), e)$;
    while $(e > 1)$
        if $(b \neq s(v) \wedge b \neq s(v') \wedge b \notin p)$
            $p[num] = b$; $num{+}{+}$;
        end if;
        $e - -$;
        $b = successor(b, s(v'), e)$;
    end while;
    if $(num < k)$ $r{+}{+}$; // less than $k$ distinct switches
    else $found$ = true;
end while;
Place $f_1$, ..., $f_k$ on the first $k$ switches stored in $p$;
RETURN $c(s(v), s(v'), r)$.
```

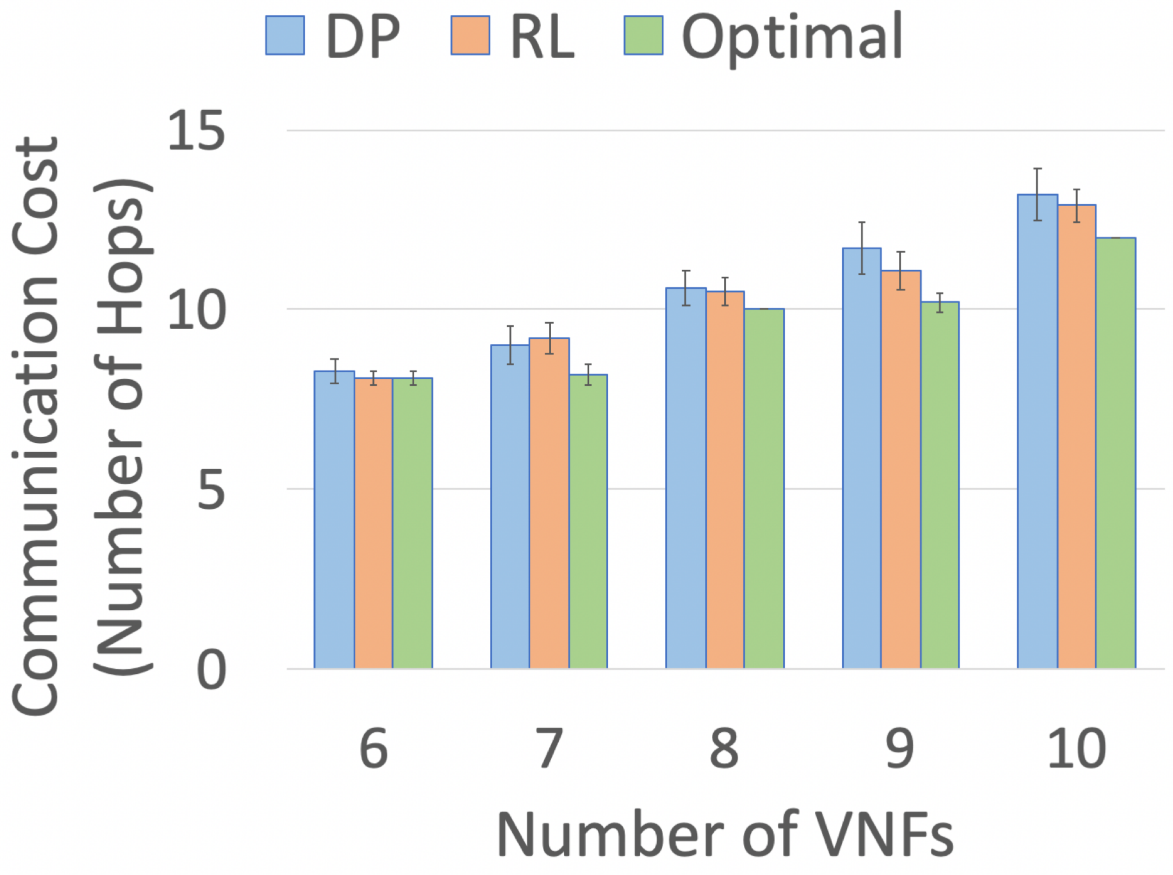
Optimal Algorithm

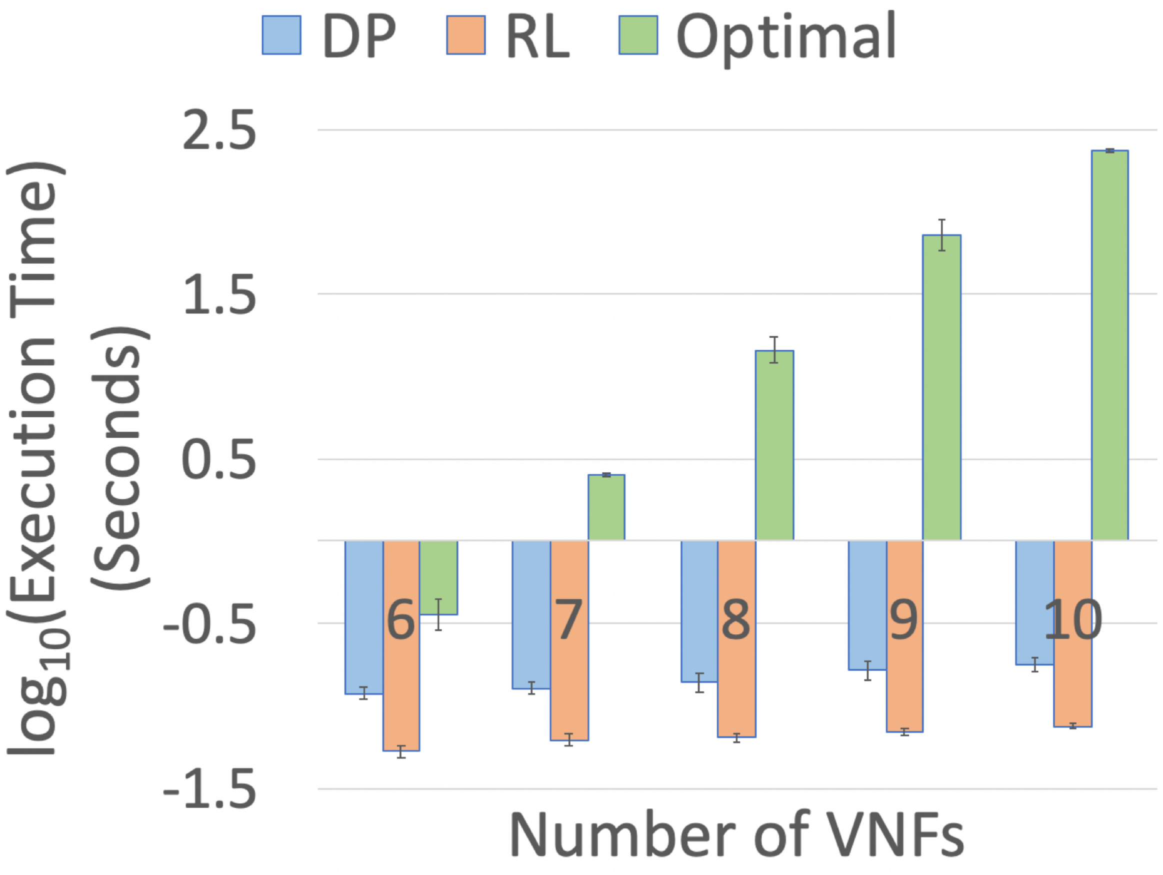


```
Algorithm 3    Exhaustive and Optimal SFC Placement.
Input: A data center graph $G(V, E)$, $s(v)$ and $s(v')$, and an SFC $(f_1, f_2, ..., f_k)$.
Output: A VNF placement $p$ and the total cost $C_c(p)$.
1. $C_c(p) = +\infty$;
2. Among all $|V_s| \cdot (|V_s| - 1) \cdot ..., \cdot(|V_s| - k + 1)$ SFC placements, find $p$ that
       gives the minimum cost $C_c(p)$;
3. RETURN  $p$ and $C_c(p)$.
```

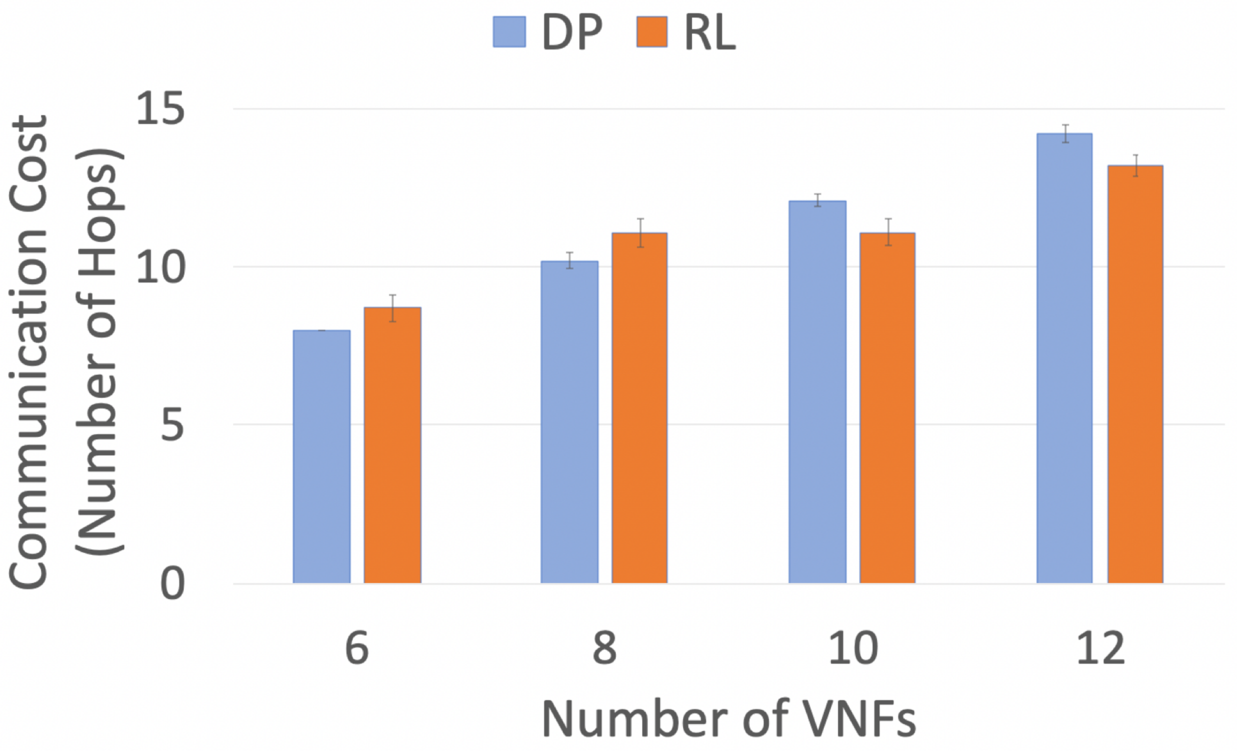
- Enumerates all the SFC placements and finds the one with minimum cost
- Time complexity: $O(|V|^k)$

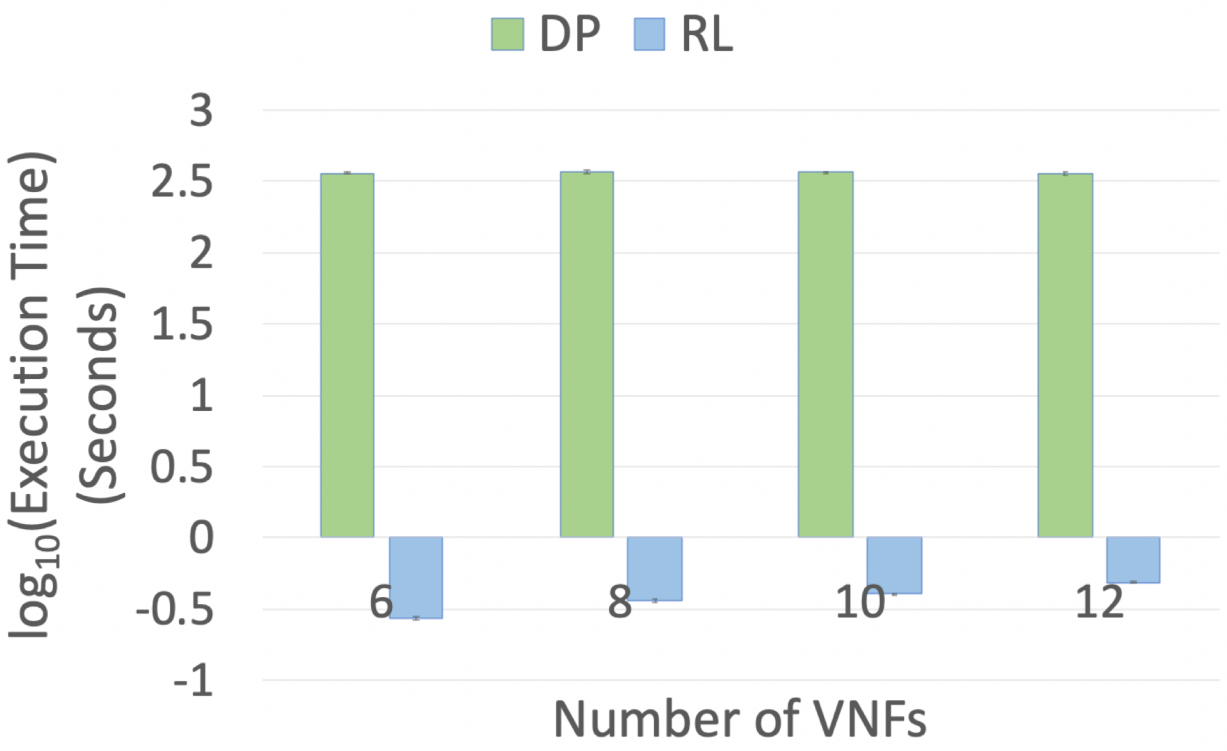
Performance Evaluation – K=4 Fat Trees



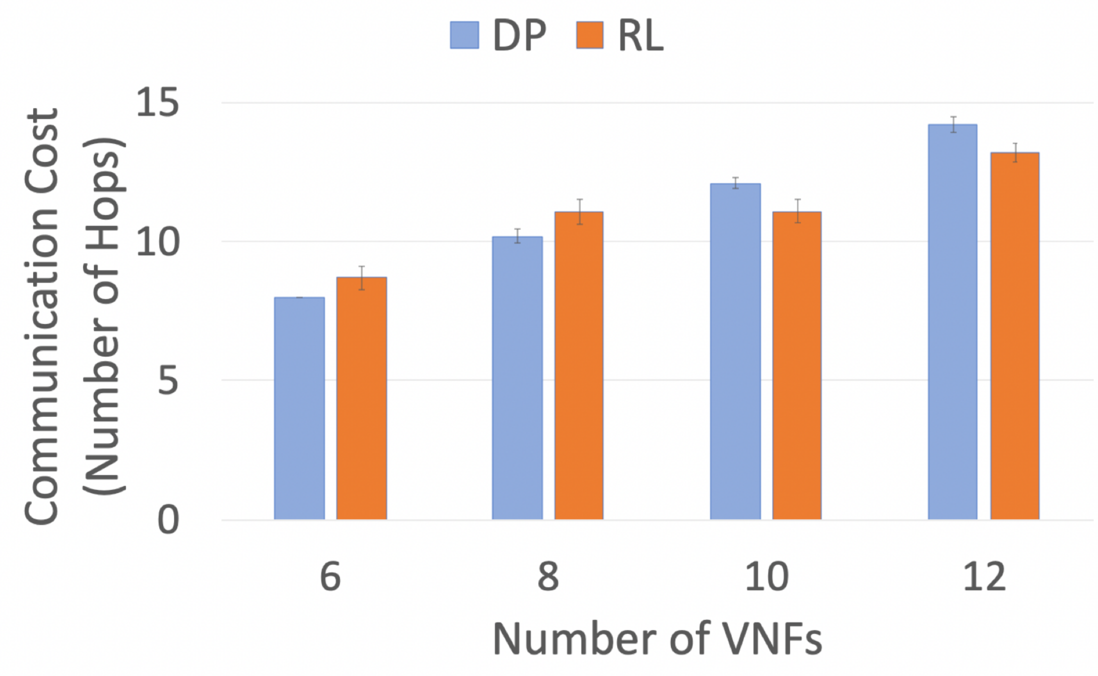


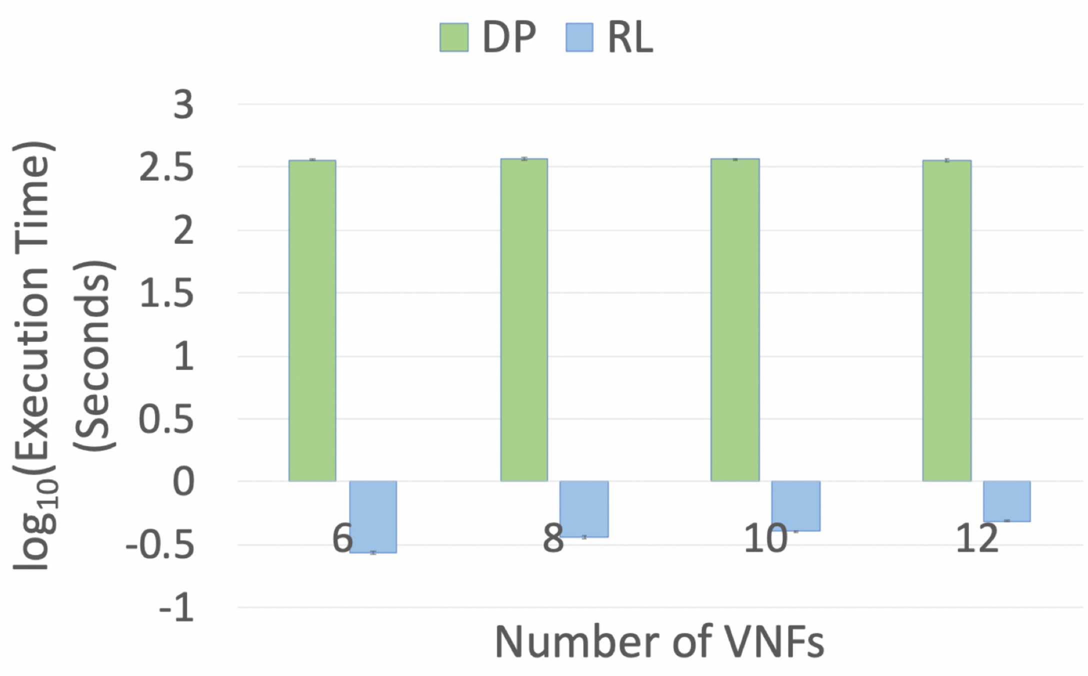
Performance Evaluation – K=8 Fat Trees





Performance Evaluation – K=8 Fat Trees





$$|V| = \frac{5}{4} * K^2$$

Algorithm	Time Complexity
MARL Algorithm	$O(N * K^3 * k)$
Dynamic Programming	$O(k * K^8)$

Performance Evaluation - # of Agents in MARL

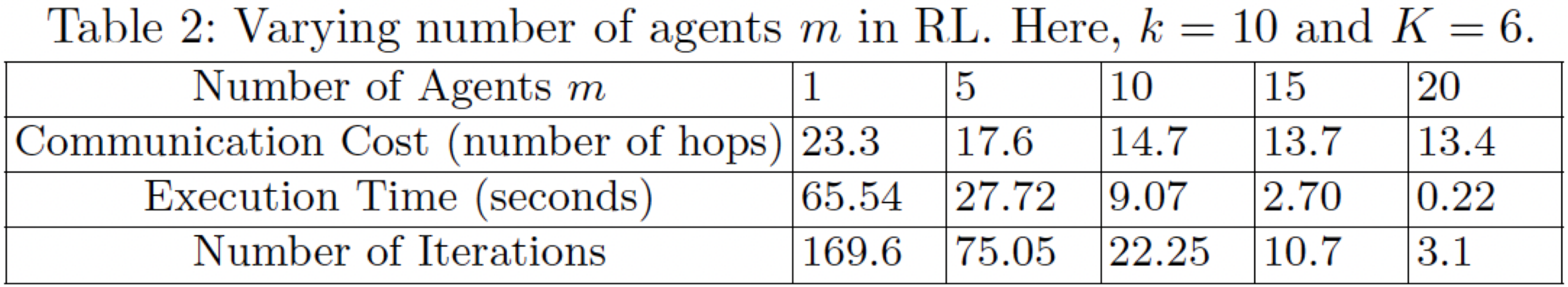


Table 2: Varying number of agents m in RL. Here, $k = 10$ and $K = 6$.

Number of Agents m	1	5	10	15	20
Communication Cost (number of hops)	23.3	17.6	14.7	13.7	13.4
Execution Time (seconds)	65.54	27.72	9.07	2.70	0.22
Number of Iterations	169.6	75.05	22.25	10.7	3.1

Conclusions and Future Work

- Novelties
    - The core of the the SFC placement problem is the k-stroll problem.
    - Solved k-stroll with RL with good performance
    - Under the k-stroll modeling of SFC placement, deep reinforcement learning is not necessary.
- Future
    - Convergence and convergence speed of our RL algorithm.
    - Balance cooperation with competition to find a near-optimal SFC placement efficiently in a rational and game theoretical environment.