MDP and Machine Learning-Based Cost-Optimization of Dynamic Resource Allocation for Network Function Virtualization

Runyu Shi[1], Jia Zhang[2], Wenjing Chu[1], Qihao Bao[2], Xiatao Jin[2], Chenran Gong[2], Qihao Zhu[2], Chang Yu[2], Steven Rosenberg[2]
[1]Dell Research, USA
[2]Carnegie Mellon University-Silicon Valley, USA
runyu_shi@dell.com, jia.zhang@sv.cmu.edu, wenjing_chu@dell.com

***Abstract*—The introduction of Network Functions Virtualization (NFV) enables service providers to offer software-defined network functions with elasticity and flexibility. Its core technique, dynamic allocation procedure of NFV components onto cloud resources requires rapid response to changes on-demand to remain cost and QoS effective. In this paper, Markov Decision Process (MDP) is applied to the NP-hard problem to dynamically allocate cloud resources for NFV components. In addition, Bayesian learning method is applied to monitor the historical resource usage in order to predict future resource reliability. Experimental results show that our proposed strategy outperforms related approaches.**

Keywords—Network Functions Virtualization, Resource Allocation, Markov Decision Process, Bayesian Learning

I. INTRODUCTION

While the advancement of cloud computing has encouraged vendors to deliver everything as a service (XaaS), delivering network functions as a service has become a new trend in the recent years. Network Functions Virtualization (NFV) [1] is the core enabling architectural concept that proposes to decouple network functions from proprietary hardware appliances and run them in software, so that they may be connected or chained to create communication services with elasticity and flexibility. Currently, virtualizable network functions include firewalls, WAN acceleration, message router, message border controller, intrusion detection, network address translation (NAT), and domain name service (DNS). Such virtualized network functions are typically run on commodity servers or datacenters in the cloud [2]. Existing virtualization technology allows a virtual machine (VM) to be relocated from one server to another without shutting it down, thus giving an opportunity of dynamically optimizing resource allocation with limited impact on performance [3].

However, how to strategically choose resources to allocate NFV components at run time to minimize overall resource cost remains a challenge [4]. Although this problem shares many similarities with the traditional placement problem [5, 6, 7], real-time NFV has posed significant new challenges due to its dynamic features. First, resource allocation inside a physical server may have to change due to dynamic workloads. NFV instances may be deployed or removed at any time in an unpredictable manner, depending on a specific service chaining. Thus, resource allocation needs to be adapted continuously. Second, the QoS demand of NFV may change when service request changes. For example, the real-time latency requirement lowers down when a content streaming is established. When a new coming request asks for accelerated delivery of streaming data, resource reallocation is desired. A new allocation plan must be recomputed taking into account the changed environment. Third, commodity cloud resources may imply potential reliability problem, thus requiring constant monitoring. In this paper, reliability represents the ability of a resource to ensure constant system operation without disruption.

The majority of existing orchestration tools is not close to our optimal goal. For example, OpenStack provides two primary resource schedulers adopting the strategy of fill-first and spread-first [8]. Fill-first, same as greedy placement, packs VMs tightly onto Physical machines (PMs). Spread-first distributes VMs across PMs in a round-robin fashion [9], but schedules VMs first on the PMs with the highest number of available CPU cores and memory. Data centers typically also adopt these two resource allocation strategies. Greedy allocation deploys all VMs to a single server first. When the server's resources are exhausted, another server is selected and the process is repeated. Round-robin placement distributes VMs to each server in succession, balancing the VM hosting load across the cluster [10]. Both methods, however, may not provide globally optimal solutions. Meanwhile, researchers have found many data centers remain to allocate resources to jobs in a static mode [11]. Gartner reported that resources are usually either under-used or misused, leading to a low 20% of CPU utilization [12]. In recent years, some researchers have strived to generate optimal resource scheduling using various advanced algorithms, such as Genetic Algorithms [13, 14] in cloud computing. Another issue is the overhead. Similar to cloud resource scheduling optimization, modeling the entire NFV resource allocation as an optimization problem will produce large scheduling overhead. It is an NP-hard problem, which lacks an efficient solution.

In this paper, we present a novel method for NFV resource allocation. In contrast to the related work, our method leverages Markov Decision Processes (MDP) to dynamically allocate NFV components to cloud resources, and applies machine learning method on dynamically collected data to predict resource reliability. MDP has been used for dynamic resource allocation problems. It centers on a policy establishment that considers long-term effect, balances all cost factors, and guides placement towards an optimal strategy [15, 16, 17]. One issue of MDP is its overhead. We exploit Bayesian learning methods to dynamically predict the reliability of cloud resources based on their historical usages, so as to further improve the MDP model performance. We have developed algorithms and have designed and conducted a collection of experiments to compare our proposed strategy with related approaches. Experimental results show that our presented strategy outperforms the related methods.

978-1-4673-7281-7/15 $31.00 © 2015 IEEE
DOI 10.1109/SCC.2015.19

The remainder of the paper is organized as follows. In Section II, we describe a motivating example that will be used to discuss our work throughout the paper. In Sections III, IV, V, and VI, we successively present our modeling, notations, learning algorithms, and performance evaluation and discussion. In Section VII, we discuss related work. In Section VIII, we draw conclusions.

II. Motivating Example

In this section, we explain a motivating example that also helps to introduce NFV design considerations. The upper portion of Fig. 1 illustrates a simplified Content Delivery Network (CDN) [18] for media delivery, highlighted by enabling network devices. Users send media service requests through home equipment to aggregator. The content distributor and manager are in charge of gathering user inputs for decision making and distributing media content. It relies on content router and content switcher to decide the routing of forwarding data packet over networks to destination sites. Content edge delivery serves as a cache to enable seamless media delivery. Content gateway joins together different networks. The functionalities of the network devices are summarized in Table 1.

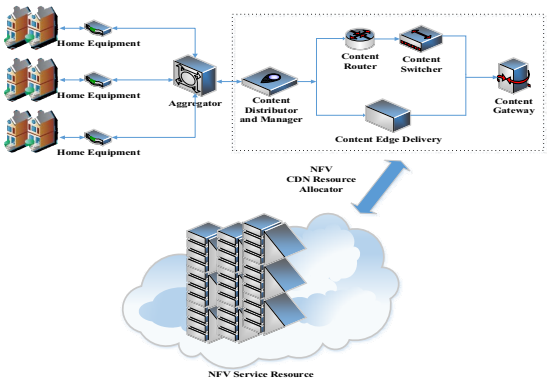


Fig. 1 Motivating example

Delivering real-time streaming data poses significant challenges to CDN hardware. One major requirement derives from dynamic massive growing amount of traffic to be delivered to end users. Meanwhile, media delivery bandwidth needs over an area may change significantly at any time, for example, when many people living in a village happen to watch online an award-winning movie on a specific Friday night. Thus, resource (i.e., bandwidth) elasticity becomes a critical demand. To address such challenges, the new trend is to virtualize the above network functions so that the real-time computation can be performed by high-performance servers in some cloud data centers, as shown in the lower portion of Fig. 1.

Table 1. NFV for CDN Components

CDN Components	Functionality	Virtualized CDN	Virtualized Functionality
Content Distributor and Manager	Gather user input, distribute content	VNF1	Virtual configuration and orchestration
Content Router	Forward data packets between networks	VNF2	Virtual routing and forwarding, segmented without using multiple devices
Content Switcher	Forward data to destination device	VNF3	Forward data between virtual and physical layers of the network. Intelligently load-balance traffic across servers
Content Edge Delivery	Cache streaming data	VNF4	Virtualized CDN cache node, cross resources implementation
Content Gateway	Join together different networks	VNF5	Virtual connection between physical and virtualized networks

In contrast to the hardware-oriented CDN, the software-oriented NFV approach promises flexibility and elasticity. However, ensuring acceptable performance (such as throughput and latency) remains a big challenge for NFV.

III. Problem Modeling and Strategy

To tackle the NFV performance problem, we study how to create a VNF allocation plan dynamically over available resources, with the goal of minimizing the cost while fulfilling predefined quality of service.

We turn the VNF allocation planning into a workflow scheduling problem. As shown in Fig. 2, the identified CDN functions are mapped (i.e., implemented) to software implementation of virtualized CDN functions. Table 1 summarizes the virtualized functions. As shown in Fig. 3, since dependencies exist among network functions, the virtualized functions inherit such dependencies and form a multi-step workflow. Operator network providers thus need to dynamically allocate virtualized CDN components to their cloud resources, as shown in Fig. 2.

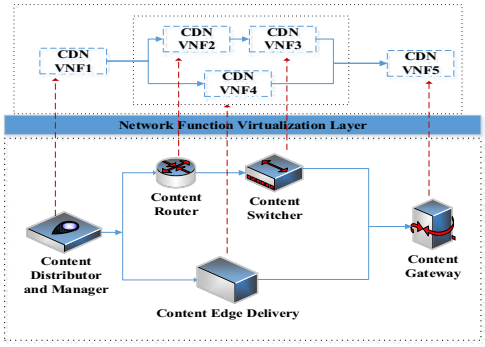


Fig. 2 Virtualization of CDN Functions

Consider a case of live streaming data content delivery. Unlike traditional workflow scheduling, an NFV workflow scheduling has to keep all resources. As shown in Fig. 2, the dependency among the NFV workflow tasks decides the order of resource scheduling. For example, a resource has to be allocated for CDN-vnf2 before being allocated for CDN-vnf3. After a virtualized network function is allocated to resource (e.g., a VM), it will start to run and remain running on the resource until the entire workflow is stopped. This unique feature poses further performance challenge on resource allocation. We thus propose a preemptive resource allocation strategy.

As shown in Fig. 3, we propose a phased allocation strategy, where workflow-level resource reallocation is enforced at each phase. Synchronization points are used to

divide a NFV workflow into phases. The example workflow in Fig. 2 is divided into three phases. At phase 1, CDN-vnf1 is allocated to Resource 2. At phase 2, an optimization algorithm (which will be discussed in detail in later sections) may allocate CDN-vnf2 and CDN-vnf3 to Resource 1, and CDN-vnf4 to Resource 4. At phase 3, the algorithm may reallocate CDN-vnf1 to Resource 4 running together with CDN-vnf5, for an overall higher performance and lower cost (e.g., based on a fitness function on capacity occupancy).

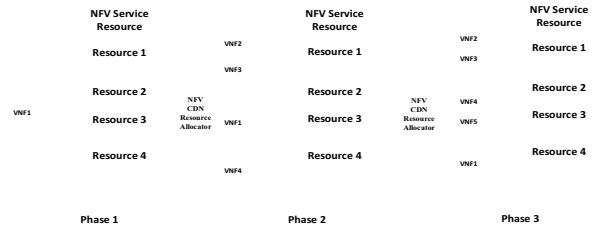


Fig.3 Phased Reallocation Strategy.

IV. Formalization & Notations

To realize our strategy, we model the NFV resource allocation as a Markov Decision Process (MDP).

A. MDP Brief Overview

MDP [17] is a mathematical framework known for modeling multi-criteria decision making, taking into consideration both immediate rewards and long-term gains. A MDP model is a 4-tuple: a set of states, a set of actions, the effects of the actions and the immediate values of the actions.

$$\text{MDP} = < \text{S}, \text{A}, \text{e(A)}, \text{o(A)} >$$

State: A state represents how the world currently exists. An action will be able to change the state of the world. All possible states represent how a world could be, which form the state space in an MDP: a set $S = < s1 \dots si \dots sn >$ denotes a finite set of states.

Action: The set of actions represent possible alternatives one can make over a set of states: a set $A = < a1 \dots ai \dots an >$ denotes a finite set of actions.

Transition: A transition specifies how each of the actions will change a state: $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ indicates the probability that action a in state S at time t will lead to state S' at time t+1.

Reward: A reward is a measure of the immediate value after performing an action in a state. $R_a(s, s')$ indicates the immediate reward received after transition to state s' from state s.

The solution to an MDP is called a policy which specifies the best action to take for each of the states [17]. To find a policy, a value function is usually pursued that specifies a numerical value for each state [19]. An MDP solution often applies a standard dynamic programming algorithm such as value iteration or policy iteration [20, 21]. Value iteration computes a new value function for each state based on the current value of its next state. Value iteration proceeds in an iterative fashion thus can converge to the optimal solution quickly [22].

B. NFV MDP Model

Based on NFV forwarding graph [13], we first model NFV components as a Directed Acyclic Graph (DAG). Let n be the total number of components; T be the finite set of components:

$$< T1 \dots Ti \dots Tn > (1 \le i \le n, 0 \le Ti \le 1)$$

Let m be the total number of resources available. Vi = 1 represents that the resource is active and has been bound to a component; while Vi = 0 means that the resource is inactive.

$$< V1 \dots Vi \dots Vm > (1 \le i \le m, 0 \le Vi \le 1)$$

We now model NFV resource allocation as an MDP problem.

Definition 1 (NFV Allocation State): A state $s \in S$ represents the current executing NFV components and their assigned resources. It is a 2-dimension m×n matrix, with rows representing components and columns representing resources. $s_{ij} = 1$ represents that component i is executed at resource j. $si = 0$ represents that the component i is not active.

Definition 2 (NFV Allocation Action): An action $a \in A$ aims to allocate a time slot on a resource to an NFV component. A is a 2-dimension m×n matrix, with m rows represent the components and n columns represent the resources. a_{ij} indicates that NFV component i is allocated to resource j with time slot a.

Definition 3 (NFV Allocation Reward): R(s,a,s') is the immediate reward received from taking action a, at state s, and transitioning to state s'.

Definition 4 (NFV Allocation Transition): An NFV allocation transition p(s|a,s') indicates the probability of whether a transition triggered by an action from one state to another can be successful.

Based on our NFV modeling, the objective function can be formalized as:

$$U(s) = \min\{R(s, a, s') + \textstyle\sum_{j \in S} p(s|a, s') \times \gamma U(s')\} \quad (1)$$

The expected optimal solution can be represented as:

$$\pi(s) = arg\ \min_{a \in A}\{R(s, a, s') + \textstyle\sum_{j \in S} \text{p(s|a, s')} \times \gamma \text{U(s')}\} \quad (2)$$

where γ is the discount factor and satisfies 0<γ<1. (For example, γ = 1/(1+r) when the discount rate is r.) γ is typically close to 1.

Without losing generality, an NFV transition probability can be calculated based the reliability of resources under investigation, which will be discussed in detail in the next section.

V. Model Learning

In the MDP model at Section VI, we define reliability of a cloud resource as its transition probability. It represents the ability of a resource to ensure constant system operation without disruption. The estimates of reliability are the measure of the resource quality related to QoS requirements. Clearly,

the optimal policy $\pi(s)$ is dependent on such an estimation. In the context of a commodity cloud environment, learning and predicting the dynamic resource reliability is critical. Thus, we propose a machine learning-based method to assess the reliability of NFV resources based on historical data.

Bayesian learning algorithm is a method of inference, in which Bayes' rule is used to update the probability estimate for a hypothesis as an additional evidence is acquired. Bayesian updating is an important technique in statistics, especially in mathematical statistics [23]. Although it requires conditional independence of the features under consideration, in practice, Bayesian learning yields good performance.

In an NFV dynamic resource allocation process, we leverage Bayesian updating to conduct the dynamic analysis of the learning data. When an NFV component is created and allocated to cloud resources, the Bayesian learning algorithm is triggered. It captures resource reliability responses and updates the transition probability of the corresponding cloud resources. As time goes by, resource reliability is trained by Bayesian learning. Note that Bayesian learning is able to track the changes of resource reliability in an evolving environment. We will discuss the detailed technique below.

After allocation action a is processed, the reliability response of the cloud resource is obtained. Such a model fulfills the Markov property [24], in which each sub-function is conditionally independent of its non-descendants given its parent sub-function. Assume that each resource Xi has a conditional probability distribution:

$$\mathrm{P}\left(\mathrm{Xi}\,|\mathrm{a},\quad \mathrm{Parents(Xi)} = \mathrm{X}'\right)$$

The calculation of the joint probability will generate the transitions probabilities. Guarded by the chain rules and the Markov property, the joint probability distribution can be written as:

$$\mathrm{P(X1,\dots Xn)} = \mathrm{P(X1)}\prod_{\mathrm{i=2}}^{\mathrm{n}}\mathrm{P(Xi}\mid \mathrm{Xi}-1) \tag{3}$$

The Bayesian learning algorithm maintains a learning counter, C, initialized to 1, for each value of a random variable. During an NFV components allocation process, when resource allocation, a, causes state variable, X, to change its value from x to x' , the learning associated with the new value is incremented by 1. The updated probability can be calculated from the prior probability:

$$\mathrm{P}'(\mathrm{X}=\mathrm{x}'|\mathrm{a},\quad \mathrm{X}=\mathrm{x}) = \frac{\mathrm{P}'(\mathrm{X}=\mathrm{x}'|\mathrm{a},\ \mathrm{X}=\mathrm{x})\ \times C+1}{C'} \tag{4}$$

where C' is the incremented learning counter.

In order to make the probability distribution over X sum to 1, the probabilities of the rest of the values of X are updated in the following pattern:

$$\mathrm{P}'\ (\mathrm{X}=\mathrm{x}'|\mathrm{a},\quad \mathrm{X}=\mathrm{x}) = \frac{\mathrm{P}'(\mathrm{X}=\mathrm{x}'|\mathrm{a},\ \mathrm{X}=\mathrm{x})\ \times C}{C'} \tag{5}$$

In our MDP model, the learning model will continuously calculate and update the probability under the allocating process.

Let us use our motivation example to explain. In an NFV component allocation process, the NFV component VNF_i is ready to allocate resource in state X_i. The learning method will record whether the previous state X_{i-1} successfully allocated the parent NFV component. It will then use the conditional probability calculation to predict the P' probability distributions. The service provider initially assigns a probability distribution according to the latest resource reliability record. If a state X is successfully accomplished, X is assigned to 1; if it failed, X is assigned to 0; otherwise X is assigned to unknown. Assume an initialization in state X_0 (initially assigned a uniform probability distribution, i.e., unknown). In state X_1, the updated probability distribution can be calculated as follows using formula (4) and (5):

$$\mathrm{P}'(\mathrm{X1}=1\,|\mathrm{a},\mathrm{X0}=\mathrm{unknown}) = \frac{0.33*1+1}{2} = 0.67$$

$$\mathrm{P}'(\mathrm{X1}=0\,|\mathrm{a},\mathrm{X0}=\mathrm{unknown}) = \frac{0.33*1}{2} = 0.165$$

$$\mathrm{P}'(\mathrm{X1}=\mathrm{unknown}|\mathrm{a},\mathrm{X0}=\mathrm{unknown}) = \frac{0.33*1}{2} = 0.165$$

P' thus becomes the new prior probability in support of the model learning calculation for the next stage.

VI. NFV Resource Allocation

A. Resource Allocation

Recall that the allocation problem here is how to map NFV components onto cloud resources, in order to achieve the minimum overall cost. Leveraging the concept of asynchronous partition developed in workflow scheduling [25], we have designed the following NFV allocation methodology illustrated in Fig. 4:

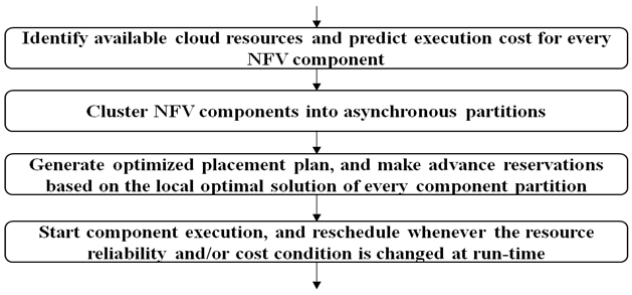


Fig. 4 NFV allocation methodology.

The initialization phase identifies available cloud resources, and evaluates the computing cost of every NFV component. We will provide details of steps in the following subsections.

Similar to scientific workflows, an NFV-oriented workflow graph comprises various structural patterns: sequential, parallel and hybrid. Since MDP is suitable for sequential pattern, we apply the method introduced by Yu et al. [25] to partition an NFV workflow graph into a collection of sequential branches jointed by synchronization components.

A synchronization component refers to a component that has more than one parent or child component. As shown in Fig. 5, VNF1 and VNF5 are both synchronization components. Other components with only one parent component and one child component are considered simple components, e.g., VNF2, VNF3, and VNF4 in Fig. 5. A branch refers to the sequential set of interdependent simple components between two synchronization components. In Fig. 5, VNF2 and VNF3 represents a branch, VNF4 is another branch. Once simple components and synchronization components are identified, branches can be identified automatically.

As a result, an NFV workflow will be partitioned into a group of sequential branches. Therefore, we can apply MDP to each branch to gain local optimization, which as a whole leading to an optimized cost solution for the entire NFV workflow.

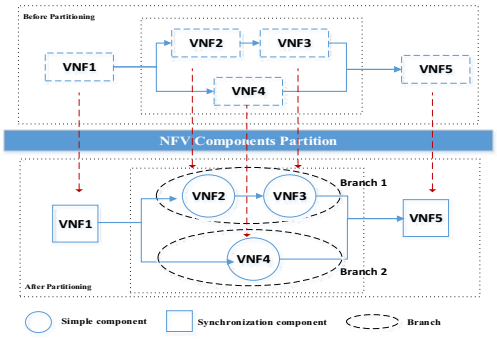


Fig. 5 NFV component partition.

In order to further reduce the total NFV cost, we break the workflow-level deadline into finer-grained sub-deadlines, as illustrated in Table 2. Based on the workflow components partition, corresponding deadlines can be assigned to each partition group. The rationale is that, if each local schedule guarantees that their component execution can be completed within the sub-deadline, then the whole workflow execution will be completed within the overall deadline.

Table 2. Pseudo code of deadline assignment

Define depth as d, each NFV component as n, workflow components set as N, synchronization components set as N', level as L, Deadline as D, sub-deadline as D_i
Deadline_assign()
d = f_depth(n) // d is from 1 to k, k is max depth.
L_d = f_level(n, d) // Dividing NFV components into levels
Group(L_d)
For L_i from L_1 to L_k
For $\forall n \in N$:
Sum (D_i) = D
D_i = Divide(D_i, L_i)
Update (D_i)
For $\forall m, n \in N'$:
$D_m = D_n$
Record (D_m) and Record (D_n)
Update (i)
Until end Level L_k

After deadline assignment, we try to find a local optimal allocation plan for each partition based on its sub-deadline. There are two main factors in our cost model, payment cost and time cost, referred as $f1$ and $f2$, respectively. Our model allows users to specify the balance between the two cost factors. For example, for service requests with real time requirements, the time factor is weighted higher in the cost function. If in an allocation plan, the total time cost exceeds the deadline, then such an allocation plan will not be considered.

$$\text{Cost} = f(f1, f2), \begin{cases} f1, & dollar\ cost \\ f2, & time\ cost \end{cases}$$

$$Cost = \begin{cases} \infty, & Excuted\ Time > Deadline \\ \text{Cost} = f(f1, f2), & otherwise \end{cases}$$

B. Pattern Cases

In the above NFV component partition process, the composition patterns play an important role. Here we discuss three patterns: sequential pattern, parallel pattern, and hybrid pattern.

Sequential pattern is a basic workflow pattern for NFV composition. Each NFV component has its sequential dependence to other components. Take an example in Fig. 5, VNF2 and VNF3 circled by Branch 1 is a sequential pattern; VNF1 is the parent of VNF3; and VNF1 has two children VNF 2 and VNF 3. Since there are multiple tasks, the allocator needs to make a decision on which resource to execute each NFV component after the completion of its parent component. The optimal decision is to minimize the total cost and complete the NFV component within the assigned sub-deadline. Sequential pattern can be solved by modeling the problem as a Markov Decision Process, which has been shown in pervious section. The algorithm is described in Table 3.

Table 3. Algorithm 1

Algorithm 1. MDP NFV components allocation algorithm for cost optimization **Input**: An NFV components graph G **Output**: A component allocation for the NFV components
Initialize s=s_0, U(s_0)=R(s_0)
Repeat
Request(processing time, price and update availability)
Update (s') *and* Update (A)
For $\forall s \in s'$ *compute*:
$U(s) = \min\left\{R\left(s, a, s'\right) + \sum_{j \in S}(1 - \mathrm{p}\left(s \middle\| a, s'\right))\mathrm{U}\left(s'\right)\right\}$
Record (policy):
$\pi(s) = arg\ \min_{a \in A}\left\{R\left(s, a, s'\right) + \sum_{j \in S}(1 - \mathrm{p}\left(s \middle\| a, s'\right))\mathrm{U}\left(s'\right)\right\}$
Record (s) and Record (s')
Update (S)
Until goal state

Parallel pattern is a workflow contains two or more tasks in parallel, for instance, in Fig. 5, VNF2 and VNF4 represent a parallel pattern. We define hybrid pattern as the workflow which contains both sequential and parallel patterns, in Fig. 5, the five components diagram illustrates a hybrid pattern.

For hybrid pattern NFV components, we can partition the hybrid pattern into sequential branches and apply MDP to each branch. The result of the cost minimization solution for each

sequential partition leads to an optimized cost solution for the entire NFV workflow. Therefore, an optimized NFV allocation can be constructed by all local optimal allocations.

If an NFV component graph is hybrid, the partitions are generated and the overall deadline is distributed on every partition. Local optimal will be calculated in each partition. Synchronization components cost should also be added to the local optimization of each branch, in order to generate a global optimal allocation. The pseudo code of the allocation method for hybrid pattern is listed in Table 4.

Table 4. Algorithm 2

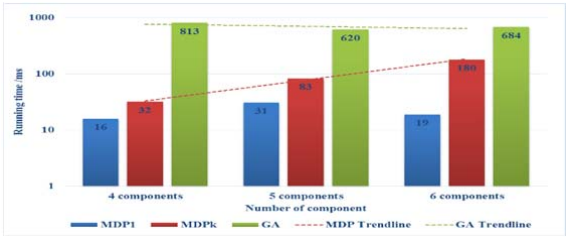
Algorithm 2. MDP NFV components allocation algorithm for cost optimization **Input**: A NFV components graph G **Output**: A component allocation for the NFV components
1. Convert NFV component graph into partition NFV component graph G' 2. Initialize s=s_0, U(s_0)=R(s_0) 3. Repeat 4. Partition(S) 5. Request(processing time, price and update availability) 6. Update (s') *and* Update (A) 7. For $\forall s \in s'$ *compute*: 8. $U(s) = \min\{R(s,a,s') + \sum_{j\in S}(1 - p(s\|a,s'))U(s')\}$ 9. Record (policy): 10. $\pi(s) = arg\min_{a\in A}\{R(s,a,s') + \sum_{j\in S}(1 - p(s\|a,s'))U(s')\}$ 11. Record (s) and Record (s') 12. Update (S) 13. Until goal state

VII. Performance Evaluation

We have designed a collection of experiments to evaluate the performance of our approach. All experiments were conducted on Dell PowerEdge Servers R720 and R810 with RHEL 6.5 operating system. The network bandwidth service is 100Mbps. We used WorkflowSim [26] as the NFV components generator for testing. WorkflowSim is a workflow simulator to assist researchers in evaluating workflow optimization techniques. We used WorkflowSim to simulate different scales of VM and NFV components.

In our evaluation, for MDP, we consider maximizing the expected total reward over a finite horizon. Given an MDP and a horizon H, we compute the optimal finite-horizon policy. In our experiment, we compared two situations: horizon H= K, and H=1. H=K means that we consider k-1 states all the way to the end; and H=1 only considers one state. Here we denote K-horizon MDP as MDPk and 1-horizon MDP as MDP1. We have compared our proposed algorithm MDPk and MDP1 with a common scheduling approach: Genetic Algorithm (GA).

In our experiment, the setting of GA is as follows: Parent selection (rouletteWheelSelection), Population size (20), Generations total (100), Mutation Rate (1/3), Recombination (One point cross over), Mutation Method (Uniform), Random Seed (1,234).



(a) All time included

(b) Time spent in solution stage only

Fig. 6 Running time with different number of NFV Component

The two metrics used to evaluate the allocation approaches are time constraint and execution cost. The former indicates the time cost by the allocation, while the latter indicates how much it costs to accomplish the NFV components.

We designed experiments to explore the performance of our proposed MDP, by varying the conditions, i.e., the number of components, the number of VMs, and different deadlines. In order to analyze the time cost, we separated the total time cost into two stages. One stage is donated as an initialization stage, which records the time cost of initializing all possible states of MDP. The other stage is donated as a solution stage, which records the time cost of finding optimal allocation plan.

Fig. 6 shows our experimental results with fixed number of VMs and varying number of NFV component tasks. We recorded method total running time and solution stage running time in Fig. 6(a) and Fig. 6(b), respectively. While considering the execution time for different methods, the results show that our MDPk method has higher performance when comparing their solution stage running time cost. However, in the initialization stage, the method traverses all the possible states, before calculating all possible allocation solutions. Comparing Fig. 6(a) with Fig. 6(b), it is clear that the time cost for the initialization significantly exceeds that of the solution stage. The time cost for solution serves as a significant factor regarding the dynamic resource allocation process. Considering the evaluation for one node takes constant time, $O(1)$, the running time for MDPk is $O(t^{(v+1)})$, and for MDP1 being $O(t \times v)$, where t stands for the number of NFV component tasks and v denotes the number of VMs.

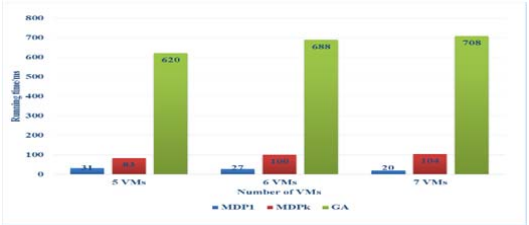
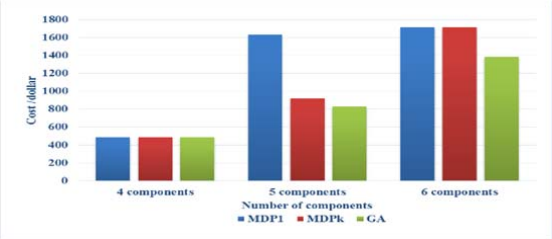


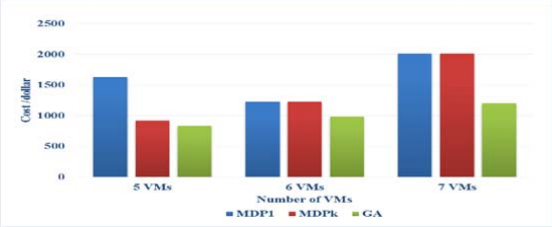
Fig. 7 Running time with different number of VMs(time spent in solution stage only)

Fig. 7 shows the experimental results when setting the fixed number of NFV component tasks and varying VMs numbers to record method running time. Even though GA costs more running time than other methods, the running time for GA remains uncertain, because the GA program keeps running until a stop or interrupting at a specific time spot. At the same time, GA cannot guarantee the global optimal result. If the random seed is not constant, different results are obtained from time to time. If the number of individuals is constant as well as all the parameters (e.g., max generations, crossover rate and mutation rate), GA is able to generate better allocation plan with less time cost, especially for complicated NFV workflows and VM conditions.

If we consider the total running time trend line, MDPk and MDP1 are larger than GA when the components number increases greatly, and most of time is spent in the initialization stage. However, the solution stage of MDPk and MDP1 takes significantly less time than GA. The reason is that MDP tends to do more floating point operations than GA. MDP has expected value calculation whereas GA does not.



(a) Cost with different number of NFV Component



(b) Cost with different number of VMS

Fig. 8 Graph of costs

Fig. 8(a) shows our experimental results with fixed number of VMs and varying number of NFV component tasks. Fig. 8(b) shows our experimental results with fixed number of NFV component tasks and varying number of VMs. As shown in Fig. 8, MDPk could reach the global optimal when varying VMs or NFV component tasks. The result of GA is close to that of MDPk. Theoretically, the K-horizon MDP can always find the best solution because it traverses all the possible states. However, if considering the deadline factor, some of the allocation plans will be filtered out, as the deadline partition is designed to be backward while the method to calculate the overall cost is forward.

We also designed an experiment that set different deadlines to record the total cost, to explore how deadline may influence the result. The results are shown in Fig. 9. While considering the deadline factor, the GA method sometimes generates better allocation plan. MDP1 has the worst performance, as it only traverses the next stage and finds the best node in that stage. In other words, it represents only short-term local optimization. Therefore, whenever there is strict time constraint, MDP1 should not be selected at the beginning. The reason why it cannot balance short term and long term is that at some stages, the cost seems to be lower for some choices and MDP1 cannot foresee the overall cost. When later the deadline is approaching, MDP1 has to choose to spend more money saving time and avoid failure.

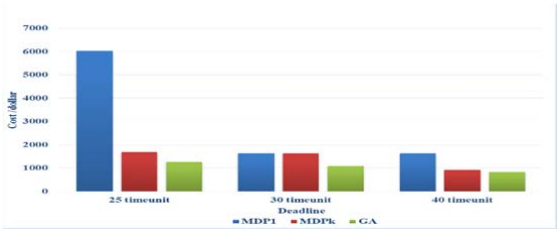


Fig. 9 Cost with different deadlines

In conclusion of our experiment results and analysis, our MDPk could reach the global optimal with relative high time cost. In terms of time cost, the best method for solution is GA. However, GA cannot guarantee to find the best solution. GA is able to find comparatively better solution under strict deadline constraints. MDP1 is able to save time only if there are comparatively special problems to handle.

VIII. Related Work

The related research work can be categorized as four categories: VM placememtn, dynamic scaling, cost modeling, and MDP.

VM placement: The dynamic VM placement has been called with other names as dynamic resource allocation, or dynamic VM management. However, the VM selection has always been part of the decision making [27]. Nelson et al. [28] presented a fast and transparent application migration system to improve the system utilization through load balancing across physical machines. They do not consider migration of networking, storage devices and physical memory usage while migrating virtual machines. Ye et al. [29] compared the live migration efficiency with different resource reservation approaches and proposed corresponding optimization methods. Meng et al. [1] proposed using traffic-aware virtual machine placement to improve network scalability based on traffic analysis.

Compared with those works, our MDP model considers the overall optimization in an NFV dynamic context, which shares some similarity with general dynamic VM placement.

Dynamic Scaling: Previous research on dynamic scaling in the cloud has mainly focused on how to scale including work on autonomic control approaches [30, 31]. Other efforts focus on how to scale in terms of vertical and horizontal scaling [10]. Investigations on supporting VM live migration for load balancing [32] or energy savings via VM consolidation across physical hosts [32, 33].

Compared with those works, our uniqueness is that our allocation focuses on optimization of the real-time dynamic and variation of QoS on network functions virtualization. We mainly consider the effects of execution cost, real-time latency and transition cost.

Cost Modeling: While considering cost modeling in dynamic resource allocation, besides deadline and cost optimization, some research pay attentions to other cost modeling factors, like robustness, budget etc. For example, Poola et al. [34] place robustness as the first priority factor in their cost model of the dynamic resource allocation. Their method aims to provide robust and fault-tolerant scheduling.

Compared with their works, we consider the dollar mount and time cost as main factors in our cost model. For the commercial cloud computing environment, the resource reliability is relatively high. Pursuing robustness seems not the priority for service users. For example, Poola's work is more suitable for certain type of cloud environment.

MDP: Adapting MDP into different problems is a main research focus of MDP. Similarly to our work that adapts MDP for NFV resource management, there are some other related works using MDP. For example, some researchers apply MDP to model web service composition. For example, Doshi et al. [35] use MDP to model workflow composition, focusing on the dynamic aspect of workflows. Gao et al. [36] present a method for dynamic web service composition based on MDP. It is defined on the base of QoS description and addresses the issue of selecting web services for the purpose of their composition.

As we explained in earlier sections, applying MDP to NFV resource management poses unique challenges. In addition to MDP, we apply machine learning algorithms to predict future resource reliability.

IX. Conclusions

The dynamic optimal resource allocation of NFV components is a critical research topic. In this paper, we have presented a novel method that combines the Markov Decision Process and Bayesian learning approach to dynamically allocate cloud computing resource for NFV components. While MDP helps to dynamically allocate NFV components to cloud resources, applying machine learning method on historical data to predict future resource reliability helps to enhance the performance of NFV-oriented cloud resource management. The experimental results show that our proposed method outperforms other greedy methods in overall cost.

We plan to further our research work in three directions. First, we plan to study the impact of cost models on different resources and to analyze the method performance among multiple competitive service providers. Second, we plan to consider NFV component dependencies between multi tenants to build a more comprehensive model. Third, we plan to study other optimization algorithms such as the Ant colony optimization algorithm in combination of machine learning techniques to further optimize the NFV-oriented resource allocation problems.

X. Reference

[1] N.M. Chowdhury and R. Boutaba, "A survey of network virtualization," Computer Networks, 54(5), pp. 862-876, 2010.

[2] A. Basta, W. Kellerer, M. Hoffmann, H.J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in Proceedings of the 4th ACM Workshop on All Things Cellular: Operations, Applications, & Challenges, pp. 33-38, 2014.

[3] M. Ciosi, "Network functions virtualization," Technical report, ETSI, Darmstadt, Germany, 2012.

[4] W. Wang, B. Li, and B. Liang, "Towards optimal capacity segmentation with hybrid cloud pricing," in Proceedings of IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), pp. 425-434, 2012.

[5] L. Jiao, J. Li, T. Xu and X. Fu, "Cost optimization for online social networks on geo-distributed clouds," in Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP), pp. 1-10, 2012.

[6] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley, "Volley: automated data placement for geo-distributed cloud services," Networked Systems Design & Implementation (NSDI), pp. 17-32. 2010.

[7] P.T. Endo, A.V. de Almeida Palhares, N.N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, and J.E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges," IEEE Network, 25(4), pp. 42-46, 2011.

[8] W. Lloyd, S. Pallickara, O. David, M. Arabi and K. Rojas, "Dynamic scaling for service oriented applications: implications of virtual machine placement on IaaS clouds," in Proceedings of 2014 IEEE International Conference on Cloud Engineering (IC2E), pp. 271-276, 2014.

[9] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future Generation Computer Systems, 28(5), pp. 755-768, 2012.

[10] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," Concurrency and Computation: Practice and Experience, 24(13), pp.1397-1420, 2012.

[11] Q. Zhang, M. F. Zhani, Q. Zhu, S. Zhang, R. Boutaba, and J. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in Proceedings of the ACM 9th International Conference on Autonomic Computing, pp. 145-154, 2012.

[12] R. Buyya, A. Beloglazov and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," in Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 6-17, 2010.

[13] Q. Li and Y. Guo, "Optimization of Resource Scheduling in Cloud Computing," in Proceedings of International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 315-320, 2010.

[14] Z. Zheng, R. Wang, H. Zhong and X. Zhang, "An approach for cloud resource scheduling based on parallel genetic algorithm," in Proceedings of IEEE 3rd International Conference on Computer Research and Development (ICCRD), vol. 2, pp. 444-447, 2011.

[15] M. Hauskrecht and H. Fraser, "Planning treatment of ischemic heart disease with partially observable markov decision processes," Artificial Intelligence in Medicine, 18(3), pp. 221-244, 2000.

[16] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, 2014.

[17] C. White III and D. J. White, "Markov decision processes," European Journal of Operational Research, 39(1), pp. 1-16, 1989.

[18] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," Communications of the ACM, 49(1), pp. 101-106, 2006.

[19] Alagoz, H. Hsu, A. J. Schaefer and M. S. Roberts, "Markov decision processes: a tool for sequential decision making under uncertainty," Medical Decision Making, 30(4), pp. 474-483. 2009.

[20] S. Martelloand P. Toth, *Knapsack problems: algorithms and computer implementations*, New York: Wiley, 1990.

[21] N. Vlassis and A. Likas, "A greedy EM algorithm for Gaussian mixture learning," Neural processing letters, 15(1), pp. 77-87, 2002.

[22] T. Ayer, O. Alagoz and N. K. Stout, "OR Forum-A pomdp approach to personalize mammography screening decisions," Operations Research, 60(5), pp. 1019-1034, 2012.

[23] J. Spiegelhalter, A. P. Dawid, S. L. Lauritzen and R. G. Cowell, "Bayesian analysis in expert systems," Statistical Science, pp. 219-247, 1993.

[24] Olajubu, and G. A. Aderounmu, "A Trustworhty Model for Reliable Cloud Service Discovery," International Journal of Computer Applications, 87(16), pp. 23-30, 2014.

[25] J. Yu, R. Buyya, C. K. Tham, "Cost-based scheduling of scientific workflow applications on utility grids," in Proceedings of the IEEE 1st International Conference on e-Science and Grid Computing, pp. 140-147, 2005.

[26] P.W. Chen, E Deelman, "Workflowsim: a toolkit for simulating scientific workflows in distributed environments," in Proceedings of the 8th IEEE International Conference on e-Science, pp. 1-8, 2012.

[27] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba and J. L. Hellerstein, "Dynamic service placement in geographically distributed clouds," IEEE Journal on Selected Areas in Communications, 31(12), pp. 762-772, 2013.

[28] M. Nelson, B. H. Lim and G. Hutchins, "Fast transparent migration for virtual machines," in Proceedings of USENIX Annual Technical Conference, General Track, pp. 391-394, 2005.

[29] K. Ye, X. Jiang, D. Huang, J. Chen and B. Wang, "Live migration of multiple virtual machines with resource reservation in cloud computing environments," in Proceedings of the 4th IEEE International Conference on Cloud Computing (CLOUD), pp. 267-274, 2011.

[30] A. Gandhi, M. Harchol-Balter R. Das and C. Lefurgy, "Optimal power allocation in server farms," ACM SIGMETRICS Performance Evaluation Review, 37(1), pp. 157-168, 2009.

[31] M. Andreolini, S. Casolari, M. Colajanni and M. Messori, "Dynamic load management of virtual machines in cloud architectures," Cloud Computing, 34, pp. 201-214, 2010.

[32] Z. Xiao, W. Song and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," IEEE Transactions on Parallel and Distributed Systems, 24(6), pp. 1107-1117, 2013.

[33] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," Computer Networks, 53(17), pp. 2923-2938, 2009.

[34] P. Doshi, R. Akkiraju, R. and K. Verma, "Dynamic workflow composition using markov decision processes," in Proceedings of IEEE International Conference on Web Services (ICWS), pp. 576-582, 2004.

[35] D. Poola, S. K. Garg, R. Buyya, Y. Yang and K. Ramamohanarao, "Robust scheduling of scientific workflows with deadline and budget constraints in clouds," in Proceedings of IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), pp. 858-865, 2014.

[36] A. Gao, D. Yang, S. Tang and M. Zhang, "Web service composition using markov decision processes," Advances in Web-age Information Management, pp. 308-319, 2005.