NUMA-Aware Virtual Machine Placement: New MMMK Model and Column Generation-Based Decomposition Approach

Xunhang Sun, *Graduate Student Member, IEEE*, Xiaoyu Cao, *Member, IEEE*, Qiaozhu Zhai, *Member, IEEE*, Haisheng Tan, *Senior Member, IEEE*, Jianchen Hu, Lei Zhu, Li Su, Wenli Zhou, Feng Gao, *Member, IEEE*, and Xiaohong Guan, *Life Fellow, IEEE*

***Abstract*— The efficiency and profitability of cloud data centers are significantly influenced by virtual machine (VM) placement. However, the Non-Uniform Memory Access (NUMA), which has been practically applied to reduce the memory bandwidth competition, is often neglected in the existing research. Actually, the incorporation of NUMA may change the traditional resource allocation mechanism, and demands for a new VM placement model. Hence, considering the multi-NUMA architecture, this paper studies the NUMA-aware VM placement (NAVMP) problem in a cloud computing system, where the resource pool is composed of enormous number of heterogeneous servers with diverse multi-resource remains. The NAVMP problem is analytically formulated as an integer program (IP). Also, for the first time, the incarnations of VM types are introduced to simplify the VM deployment rules originated from complex NUMA architecture. We aim to maximize the VM provision ability (VPA) of the resource pool, and thus propose a novel Value Function to describe servers' VPA. The resulting formulation, which is a new variant of the multiple-choice multiple multi-dimensional knapsack (MMMK) problem, is of significant computational challenges. So we customize a decomposition approach based on Column Generation (CG) to support the offline optimization. Numerical experiments on a practical dataset demonstrate the validity and scalability of the customized CG-based approach. Our approach outperforms a professional IP solver, i.e., Cbc, and a popular meta-heuristic algorithm, i.e., genetic algorithm (GA), and can efficiently address large-scale NAVMP instances with ten thousands of VM demands and servers.**

***Note to Practitioners*—This paper proposes a novel IP model for NAVMP. To cope with the complicated deployment logic associated with the complex multi-NUMA architecture of modern multi-core systems, we present an NAVMP formulation from the perspective of incarnations of VM types. Different from the traditional VM placement problem that aims to minimize the number of activated servers, i.e., the vector bin packing (VBP)-based model, we adopt the objective that maximizes the VPA of a resource pool for further improving the resource utilization. The resulting formulation is an MMMK problem, which is computational very challenging for a practical scale resource pool. Hence, to mitigate the computation burden, we design and implement a CG-based decomposition approach to support the offline optimization for NAVMP. Parallelization scheme and non-trivial heuristic strategies are applied to promote the computation efficiency. According to our numerical experiments, the proposed decomposition approach demonstrates a much superior solution capacity to the Cbc solver and GA. In particular, to achieve a comparable solution precision with Cbc, the computing time can be reduced by orders of magnitude. Also the CG-based approach outperforms GA in both the solution quality and computation time for large-scale instances. Besides, compared to the VBP model, our MMMK-based NAVMP model has improved the VPA up to 44.39%. Practically, the proposed offline approach can be leveraged to guide online VM allocation decisions, and perform efficient results evaluation.**

***Index Terms*— Cloud computing, virtual machine placement, non-uniform memory access, multiple-choice multiple multi-dimensional knapsack problem, column generation.**

Manuscript received 21 November 2023; accepted 22 February 2024. Date of publication 29 February 2024; date of current version 31 January 2025. This article was recommended for publication by Associate Editor X. Zuo and Editor J. Li upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFA1004600; in part by the National Natural Science Foundation of China under Grant 62192752, Grant 62103321, Grant 62103319, and Grant 62373294; and in part by a grant from Huawei Cloud Huawei Technologies Company Ltd. *(Corresponding author: Xiaoyu Cao.)*

Xunhang Sun and Jianchen Hu are with the School of Automation Science and Engineering and the Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: xhsun@sei.xjtu.edu.cn; horace89@xjtu.edu.cn).

Xiaoyu Cao and Qiaozhu Zhai are with the School of Automation Science and Engineering and the Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Smart Integrated Energy Department, Sichuan Digital Economy Industry Development Research Institute, Chengdu 610037, China (e-mail: cxykeven2019@xjtu.edu.cn; qzzhai@sei.xjtu.edu.cn).

Haisheng Tan is with the LINKE Laboratory and the CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China (USTC), Hefei 230027, China (e-mail: hstan@ustc.edu.cn).

Lei Zhu is with the Computing Service Product Department, Huawei Cloud Computing Technologies Company, Xi'an 710076, China (e-mail: zhulei41@huawei.com).

Li Su and Wenli Zhou are with the Algorithm Innovation Laboratory, Huawei Cloud Computing Technologies Company, Xi'an 710076, China (e-mail: leigh.su@huawei.com; zhouwenli@huawei.com).

Feng Gao is with the School of Automation Science and Engineering and the State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: fgao@sei.xjtu.edu.cn).

Xiaohong Guan is with the School of Automation Science and Engineering and the Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: xhguan@xjtu.edu.cn).

Digital Object Identifier 10.1109/TASE.2024.3370392

NOMENCLATURE

Abbreviation

CG	Column generation.
IaaS	Infrastructure-as-a-Service.
IP/LP	Integer/linear program.

MMMK	Multiple-choice multiple multi-dimensional knapsack.
NAVMP	NUMA-aware VM placement.
NUMA	Non-uniform memory access.
VBP	Vector bin packing.
VM	Virtual machine.
VPA	VM provision ability.

Sets and Indices

γ	Index of resources.
$\mathcal{P}_s$	Subset of NUMA-aware placement patterns of server type s.
Ω_s	Set of server type s.
τ	Index of CG algorithm iterations.
i	Index of flavors.
k	Index of servers.
q	Index of incarnations.
s	Index of server types.
t	Index of NUMA-aware placement patterns.

Parameters

$\boldsymbol{\xi}_s^t$	NUMA-aware placement pattern t of server type s.
ω_i	Worth of flavor i.
$\overline{\Gamma}/\Gamma$	Number of NUMA resource types (in NUMA-oblivious form/NUMA-aware extensive form).
$\xi_{is}^{q,t}$	Number of incarnation q of flavor i in NUMA-aware placement pattern t of server type s.
K	Number of servers.
N	Number of flavors.
n_i	Demand of flavor i.
Q_i	Number of incarnations of flavor i.
$r_i^q(\gamma)$	Resource γ request of incarnation q of flavor i.
$R_k(\gamma)/R_s(\gamma)$	Residual capacity of resource γ of server k/server type s.
S	Number of server types.
T_s	Number of NUMA-aware placement patterns of server type s.
u_s	Number of servers belonging to server type s.

Decision Variables

λ_s^t	Number of NUMA-aware placement pattern t of server type s (integer variable).
x_{ik}^q	Number of incarnation q of flavor i deployed to server k (integer variable).

I. Introduction

CLOUD computing provides "pay-as-you-use" services to users, enabling them to enjoy elastic and high-quality computing power on demand, similar to the access of water and electricity in everyday life. With the rapid development of Infrastructure-as-a-Service (IaaS) cloud, there is a huge surge in the demands for cloud resources. By virtue of the virtualization technology, an IaaS platform provides resources in the form of *Virtual Machines (VMs)*, which are "slices" over physical servers [1]. With the incoming demands of cloud users, some VMs will be launched on the servers to provide computing services. This process is essentially the selection of suitable servers for VM provision. The resource allocation on the cloud can be generally described as the VM placement problem [2], [3], where a number of VMs (related to users' requests) need to be deployed on specific servers in the resource pool. An appropriate VM placement scheme could render a tremendous impact on the resource consolidation efficiency and economic profitability. Actually, even a 1% increase in resource utilization can save millions of dollars in costs for large cloud service providers [4].

Nowadays, the *Non-Uniform Memory Access (NUMA)* plays an increasing important role in modern multicore systems [5], [6], [7], [8]. With a higher overall system throughput, NUMA outperforms the traditional symmetric multi-processor (SMP) mode by alleviating the memory bandwidth competition [9]. And for cloud providers, the NUMA mode is more economical than the famous massively parallel processing (MPP) system [10]. Albeit the VM placement problem has been well studied, the existing works generally neglect the NUMA architecture. For such an NUMA-oblivious instance, however, the VM placement scheme could be negatively affected by the high latency to access the remote memory. So, to achieve a greater resource management efficiency, as well as full benefits of the NUMA architecture, the *NUMA-aware VM placement (NAVMP)* should be considered and carefully addressed [8]. Indeed, as indicated by the test results in [7] and [9], comparing to the canonical NUMA-oblivious one, the NUMA-aware resource allocation usually brings positive impacts on demanding jobs. Nevertheless, NUMA may significantly change the mechanism of VM placement. Specifically, the VM placement needs to be implemented not only pertaining to which server, but further with a decision on which NUMA nodes of the server. As a result, the complex deployment rules originated from NUMA (to be elaborate in Section III-B) could introduce a resource management formulation with various complicated NUMA-related constraints, leading to extra computational challenges [6]. Therefore, it demands for a more compact description of NUMA-related deployment rules, e.g., without complicated or redundant logical constraints.

Moreover, the NUMA-based computing infrastructure could largely increase the difficulty of optimal decisions on VM placement. Although, in practice, the VM placement is an online procedure, its *offline optimization* (i.e., the complete information of VMs and servers is known a priori) is also of great significance. On the one hand, to pursue a fast solution (in milliseconds or seconds), the application of online algorithm may inevitably lose the optimality. In this regard, the offline optimization is necessary to yield the global optimal solution for a fair evaluation of the online strategy. On the other hand, the results of offline optimization may have the potential to generate more valuable information, e.g., placement patterns [3], so as to guide the online resource allocation.

Additionally, the offline optimization of VM placement can be integrated in the decision framework of cloud capacity management (e.g., the capacity expansion of the resource pool), which would further enhance the efficiency of resource management. Nevertheless, the NAVMP problem is essentially an integer program (IP) with NP-hardness. Thus, even with more desirable formulation of NUMA-related placement constraints, the offline NAVMP problem could still be intractable when the problem scale becomes very large. Hence, we have to design an efficient solution algorithm to handle the computational challenges.

In this paper, we study the VM placement problem under an NUMA-aware consideration. With the help of incarnations of VM types, a novel NAVMP model is formulated to cope with the complex NUMA-related deployment rules. Each incarnation implies a NUMA selection, and it is predefined in advance so that our formulation does not need to incorporate the complex logic of NUMA selection. Also, different from the traditional VM placement model, which is generally considered as a vector bin packing (VBP) problem with a target to minimize the number of activated servers (i.e., occupied by running VMs), the proposed problem aims to maximize the VM provision ability (VPA) of the resource pool. The resulting formulation is actually a vector-oriented variant of the Multiple-Choice Multiple Multi-Dimensional Knapsack (MMMK) problem, which is NP-hard and with exponential computation complexity. To decrease the computational burden, we tailor a decomposition approach based on Column Generation (CG), which demonstrates a strong solution capacity for the NAVMP problem. The major contributions of our study are summarized as:

1) A novel MMMK-based formulation is proposed to support NAVMP, where the incarnations of VM types are introduced to cope with the multi-NUMA architecture. Different from the widely adopted VBP-based model, which aims to minimize the activated servers, our objective is to maximize VPA of the resource pool subject to the request satisfaction and capacity limitation constraints. Furthermore, we develop a Value Function to quantitatively evaluate a server's VPA. To the best of our knowledge, this is the first time to adopt this MMMK-type model in VM placement.
2) A CG-based decomposition approach is developed to overcome the computational challenges associated with the offline optimization for the proposed NAVMP problem. In particular, a reformulation based on NUMA-aware placement patterns is presented. Also, a CG algorithm, coupling with heuristic strategies for initialization and feasible solution construction, is customized and implemented under parallel computation settings to yield a high-quality solution within acceptable time budget.
3) A series of numerical experiments based on a practical dataset from Huawei Cloud are conducted to validate our NAVMP formulation and the superiority of the customized CG-based approach.

The remainder of this paper is organized as follows. A literature review is presented in Section II. Section III proposes the NAVMP model. Section IV develops the CG-based optimization approach. An illustrative example is given in Section V. Then, we conduct extensive numerical experiments in Section VI. Finally, conclusions are drawn and discussed in Section VII.

II. Related Work

To date, the VM placement problem [2], [3], [11], [12], [13], [14], [15], and its variants, e.g., the consideration of network services [16], [17], [18], geo-distributed cloud [1], [19], VM migration [20], and energy consumption [21], [22], [23], [24], [25], have been extensively studied in the current literature. However, these studies mainly concern the VM placement without NUMA. The specific research on NAVMP could be quite limited. Wu et al. [7] presented a holistic NUMA-aware scheduling policy combining both the machine- and cluster-level NUMA-aware optimizations. Cheng et al. [9] introduced a "Best NUMA Node" based VM scheduling algorithm, which attempted to improve the memory access locality while maintaining system's load balance, and implemented it in a user-level scheduler. Cheng et al. [26] proposed a traffic-aware VM deployment scheme on NUMA systems, and adopted the hardware performance monitoring technique to describe VM memory behaviors. Hu et al. [6] proposed a VM consolidation model considering both the NUMA and non-NUMA resources, and a hybrid heuristic Grey-Wolf algorithm was presented to overcome the computational challenges. This model was formulated towards how to explicitly assign VMs on certain NUMA nodes, and the resulting NUMA-related complicated logic undermines the structure of traditional VM placement model. Sheng et al. [27] studied multi-NUMA VM scheduling through reinforcement learning. Since the state and action spaces increase exponentially with the number of servers, the proposed algorithm could be inapplicable to handle large-scale NAVMP problems.

In the existing studies, the VM placement is generally treated as a VBP problem [2], [3], [11], which aims to minimize the number of activated servers thus reducing the operation and maintenance (O&M) expenses of cloud data centers. Considering the energy cost accounts for around 40% of the O&M expenses [19], a lot of researches has integrated the effect of energy consumption in VM placement as one of the multiple objectives [21], [22], [23], [24], [25]. While in practice, we find the VM placement strategy following this target still inevitably leads to large resource fragmentation. To further improve the efficiency of resource utilization, the maximization of VPA for resource pools could be a more appropriate goal for the resource allocation. In this regard, the FIT-principle driven algorithms (e.g., BestFit [28] and FirstFit [29]) specifically designed for VBP are no longer applicable. By virtue of the limitation of the VM type number, Shi et al. [3] proposed a VM placement model based on placement patterns, then presented a delayed CG method to seek for optimal solutions. Although their model, which remained as a VBP problem, only considered homogeneous servers and was NUMA-oblivious, we believe it can be extended to the NUMA-aware case with the VPA maximizing objective and heterogeneous servers.

Note that our NAVMP formulation is essentially a variant of the MMMK problem. To the best of our knowledge, the current studies rarely concerned and dealt with a similar type of this problem. Indeed, the multiple-choice multi-dimensional knapsack problem has been widely studied (e.g., [30], [31]), but only considering a single-knapsack instance. Besides, there exists another stream of research focusing on the multiple multi-dimensional knapsack problem (e.g., [32], [33]), yet they omit the multiple-choice characteristics. In summary, the MMMK problem can be considered as a combination of these two types of problems, while their solution strategies can provide a guidance to effectively handle the MMMK problem.

CG is popular for combinatorial optimization problems with separate structures. There is a large literature using CG to cope with practical problems, such as parallel machine scheduling [34], home healthcare routing and scheduling [35], and so on [36], [37], [38]. Recently, some papers have made efforts to combine CG with meta-heuristic algorithms to improve the computational efficiency [39], [40], [41]. Also, CG has been proven to be an efficient solution approach for the knapsack problem [30], [42], which provides promising support for applying CG to grapple with the MMMK-based NAVMP problem.

III. NAVMP Model

The procedures of VM placement are illustrated in Fig. 1. We consider a cloud computing system, where several VM types (i.e., *flavors*) are predefined as commodities. Also, a large number of servers in the resource pool are provided to satisfy the users' resource requests, e.g., CPU, memory, network bandwidth and storage. Each flavor is treated as a combination of various resources with specified amounts. Users can make flavor procurement according to their demands and capital budgets, and send a corresponding resource request to the cloud platform. Then, the VMs will be assigned to servers by a cloud scheduler to execute programs. The responsibility of the central scheduler is to collect resource information, and decide which server to deploy the VMs on. This paper focuses on the VM placement between two successive capacity expansions, which derives an assumption that the available servers are always sufficient.

Given the above preliminaries, we represent the VMs and servers as multi-dimensional vectors, and develop an incarnation-oriented model for the multi-NUMA system. Also, to explicitly express the objective function of NAVMP, a value function is built to measure the VPA of servers. Accordingly, an integrated formulation of NAVMP is proposed that maximizes future VPA of the entire resource pool.

A. VM and Server

Both the VM and the server are modeled as a multi-dimensional vector, each dimension of which corresponds to a type of resource. As shown in Fig. 2, for a server, the total resource requests of the assigned VMs cannot exceed the server's resource capacities in each dimension. The servers provided in the resource pool are heterogeneous due to both various server products and existing different VM placement

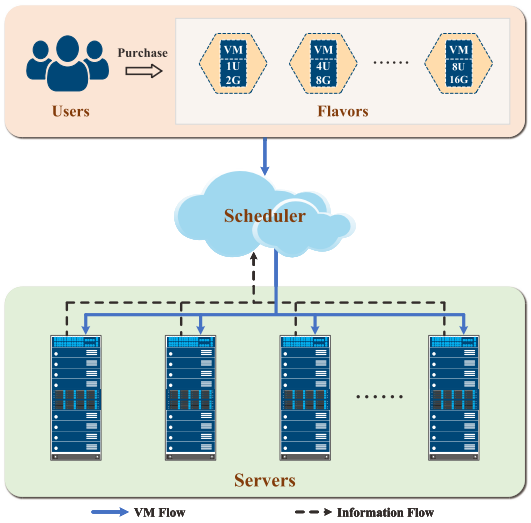


Fig. 1. VM placement in a cloud platform.

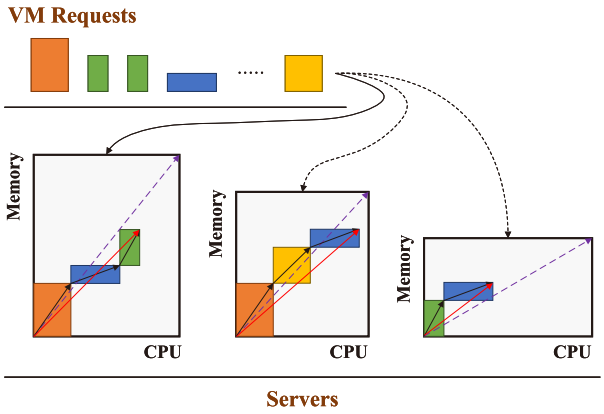


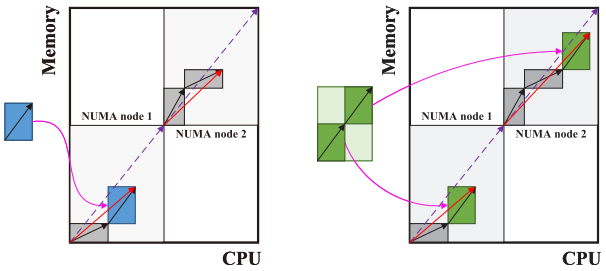
Fig. 2. Model of VM placement process from the perspective of multi-dimensional vectors (NUMA-oblivious, two kinds of resources including CPU and memory are considered). VMs are depicted as black solid vectors and servers as purple dashed vectors. The sum vectors in red represent the total resource requests of VMs.

states. According to residual resource capacities, we can divide servers into various *server types*.

B. NUMA System

In NUMA systems, all servers possess multiple NUMA nodes. All the resources are divided into NUMA resources (e.g., CPU, memory) and non-NUMA resources (e.g., disk, network, storage) [6]. For simplicity, we only consider NUMA resources in this paper and denote the number of NUMA resource types as $\overline{\Gamma}$.

Note that the VM placement mechanism will change because of the multi-NUMA architecture [8], [27]. When a small-scale VM comes, the scheduler will select one NUMA node to deploy it on. While for a relatively larger VM, it may need to be assigned across multiple NUMA nodes of *the same server*. More concretely, suppose in a σ-NUMA system, all servers have σ NUMA nodes, and ι-NUMA flavors ($\iota \leq \sigma$) are predefined to be *equally* created across ι NUMA nodes of a server. Generally, σ and ι are the power of two (e.g., 2^0, 2^1 and 2^n). In Fig. 3, we give an example of a double-NUMA system to illustrate NAVMP.



(a) Single-NUMA VM. (b) Double-NUMA VM.

Fig. 3. Illustration of NAVMP in a double-NUMA system. The purple dashed vector with a shading area represents a NUMA node of a server.

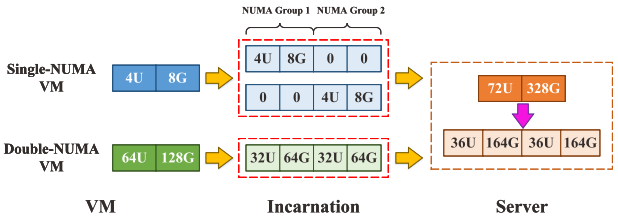


Fig. 4. Illustration of the incarnation-based NAVMP model. In this example, $\overline{\Gamma} = 2$, $\Gamma = 4$, and $\sigma = 2$.

For a σ-NUMA system, we extend the server to a $\Gamma(= \sigma \cdot \overline{\Gamma})$-dimensional vector. Every $\overline{\Gamma}$ dimensions form a NUMA group, representing the resource capacities of a NUMA node. Following the idea of multiple-choice modeling in [30] and [43], we introduce the concept of *incarnation* to describe a flavor's NUMA-aware deployment scheme. Each flavor has a set of incarnations, where each incarnation represents a specific NUMA selection. Similar to the server, an incarnation is also modeled as a Γ-dimensional vector with σ NUMA groups. For an ι-NUMA flavor, ι groups of each incarnation denotes the resource request for the corresponding selected NUMA node, and the remaining dimensions are all zeros. For example, a double-NUMA system can be defined as in Fig. 4. Considering a server configured with 72 CPU cores and 328G memory (denoted as (72U,328G)), its NUMA-aware extension is (36U,164G,36U,164G). Single-NUMA VM (4U,8G) has two incarnations (4U,8G,0,0) and (0,0,4U,8G), corresponding to whether a VM is assigned to NUMA node 1/2 of a server, respectively. And double-NUMA VM (64U,128G) only has one incarnation (32U,64G,32U,64G), indicating that it will be assigned across both NUMA nodes of a server. Rather than deciding on which NUMA node to place a VM directly, *from our incarnation-based NUMA modeling perspective, the cloud schedule will select one incarnation of each VM, and then decide which server to deploy it on.* Besides, for a σ-NUMA system, the number of an ι-NUMA flavor's incarnations is

$$\binom{\sigma}{\iota} = \frac{\sigma!}{\iota! \cdot (\sigma - \iota)!} \tag{1}$$

Remark 1: By introducing the concept of incarnation, our model can be easily extended to contain both NUMA and non-NUMA resources. For instance, we can suppose that the server's total capacities of a non-NUMA resource are equally distributed into each NUMA node. Correspondingly, as to a flavor, the request of this non-NUMA resource can be divided equally into every (zero and non-zero) NUMA group of an incarnation, while the number of incarnations will not change.

C. VM Provision Ability

We introduce the measurement of *resource value* by a *value function* to quantitatively represent each server's VM provision ability (VPA).

Assumption 1: The same type of resources is homogeneous across different servers.

Assumption 1 indicates that a specific resource, e.g., CPU or memory, provided by different servers will offer users the same service. Hence, servers should share the same value function, which can be defined as follows:

Definition 1: (Value Function) The VPA of a server can be measured through the value function $f(\ell(1), \cdots, \ell(\Gamma))$, where $\ell(\gamma)$ represents the remaining capacities of resource γ. The value function should satisfy three properties:

1) *Suppose $\mathcal{D}_d(d = 1, \cdots, \sigma)$ is the set of NUMA group d. $\forall d = 1, \cdots, \sigma$, if $\exists \gamma_d \in \mathcal{D}_d$ such that $\ell(\gamma_1) = \cdots = \ell(\gamma_\sigma) = 0$, there exists $f(\ell(1), \cdots, \ell(\Gamma)) = 0$.*
2) *$f(\ell(1), \cdots, \ell(\Gamma))$ is monotonically increasing in each dimension.*
3) *Let $(\alpha(1), \cdots, \alpha(\Gamma))(0 \le \alpha(\gamma) \le \ell(\gamma))$ be the resources requested by a batch of VMs, and the corresponding worth of these VMs is $g(\alpha(1), \cdots, \alpha(\Gamma))$. Then, we have*

$$f(\ell(1), \cdots, \ell(\Gamma)) \ge g(\alpha(1), \cdots, \alpha(\Gamma)) + f(\ell(1) - \alpha(1), \cdots, \ell(\Gamma) - \alpha(\Gamma)).$$

The first two properties define the boundary condition and the monotonicity of the value function, respectively. The third one associates the servers' resource value with VMs' worth, indicating that the resource value of a server not only increases with an enlarging capacity, but also highly depends on the worth of the VM group.

According to the aforementioned properties, one of the viable forms of the value function is given as:

$$\max g(\ell(1), \cdots, \ell(\Gamma)) \tag{2}$$

which is to maximize the worth of the VM group under the resource request $(\ell(1), \cdots, \ell(\Gamma))$ by choosing the optimal VM combination. It is easy to verify that $\max g(\ell(1), \cdots, \ell(\Gamma))$ satisfies the boundary condition and monotonicity. Besides, for each group of $(\alpha(1), \cdots, \alpha(\Gamma))$, we have

$$\begin{aligned} &\max g(\ell(1), \cdots, \ell(\Gamma)) \\ &\ge \max g(\alpha(1), \cdots, \alpha(\Gamma)) + \max g(\ell(1) - \alpha(1), \\ &\quad \cdots, \ell(\Gamma) - \alpha(\Gamma)) \\ &\ge g(\alpha(1), \cdots, \alpha(\Gamma)) + \max g(\ell(1) - \alpha(1), \cdots, \ell(\Gamma) - \alpha(\Gamma)), \end{aligned}$$

which validates the third property. Then, by introducing the worth of flavor i (as denoted by ω_i), the value function can be calculated by solving the following IP:

$$\max g(\ell(1), \cdots, \ell(\Gamma)) = \max_{x_i^q} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} \omega_i x_i^q \tag{3}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \sum_{q=1}^{Q_i} r_i^q(\gamma) x_i^q \le \ell(\gamma), \quad \gamma = 1, \cdots, \Gamma \tag{4}$$

$$x_{i\cdot}^{q} \in \mathbb{N}, \quad i = 1, \cdots, N; \ q = 1, \ldots, Q_i \tag{5}$$

where, with a little notation abuse, $x_{i\cdot}^{q}$ denotes the number of incarnation q of flavor i assigned to a server. Note that the worth of flavors should be predefined by the cloud provider based on their resource capacities, importance, and expected levels, etc.

D. NAVMP Formulation

We propose an analytical formulation of NAVMP, the purpose of which is to maximize future VPA of the resource pool. Based on the measurement of VPA, in other words, it would maximize the total remaining resource value of the servers after deploying a batch of VMs. According to the value function in (3)-(5), and considering that the total worth of the VM batch is a constant, the NAVMP problem (**I-NAVMP**) can be expressed as the following incarnation-oriented formulation.

$$\mathbf{I-NAVMP}:$$

$$\psi := \max_{x_{ik}^{q}} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} \omega_i x_{ik}^{q} \tag{6}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{q=1}^{Q_i} x_{ik}^{q} \geq n_i, \quad i = 1, \cdots, N \tag{7}$$

$$\sum_{i=1}^{N} \sum_{q=1}^{Q_i} r_i^{q}(\gamma) x_{ik}^{q} \leq R_k(\gamma), \quad k = 1, \cdots, K; \ \gamma = 1, \cdots, \Gamma \tag{8}$$

$$x_{ik}^{q} \in \mathbb{N}, \quad i = 1, \cdots, N; \quad q = 1, \cdots, Q_i; \ k = 1, \cdots, K \tag{9}$$

where x_{ik}^{q} denotes the number of incarnation q of flavor i deployed on server k. These variables are constrained by (7) to satisfy the users' VM requests. Also, the restrictions of server resources are imposed in (8). And, we make the following feasibility assumption:

Assumption 2: The resource capacities of the servers are sufficient, i.e., ***I-NAVMP*** *is always feasible.*

The above assumption can be guaranteed by choosing suitable capacity expansion decisions for the resource pool, which is not the main concern of this paper.

Remark 2: In ***I-NAVMP****, there are two parts of VMs. The first part is "Demand VMs", which satisfies users' resource requests n_i. The VMs in the second part are used to measure the resource pool's future VPA, so we called them "Imaginary VMs". Using y_{ik}^{q}/z_{ik}^{q} to represent the number of incarnation q of flavor i of demand/imaginary VMs deployed to server k, we can obtain another formulation of NAVMP:*

$$\max_{y_{ik}^{q}, z_{ik}^{q}} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} \omega_i z_{ik}^{q} \tag{10}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{q=1}^{Q_i} y_{ik}^{q} = n_i, \quad i = 1, \cdots, N \tag{11}$$

$$\sum_{i=1}^{N} \sum_{q=1}^{Q_i} r_i^{q}(\gamma)(y_{ik}^{q} + z_{ik}^{q}) \leq R_k(\gamma), \quad k = 1, \cdots, K; \ \gamma = 1, \cdots, \Gamma \tag{12}$$

$$y_{ik}^{q} \in \mathbb{N}, \ z_{ik}^{q} \in \mathbb{N} \quad i = 1, \cdots, N; \quad q = 1, \cdots, Q_i; \ k = 1, \cdots, K \tag{13}$$

Compared with this formulation, ***I-NAVMP*** *adds a constant term $\sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} \omega_i y_{ik}^{q}$ to the objective function, and merges y_{ik}^{q} and z_{ik}^{q} to x_{ik}^{q} for simplicy.* ***I-NAVMP*** *ensures all decision variables appear in the objective function and constraints simultaneously, which facilitates the design and execution of solution algorithms. As an aside, in* ***I-NAVMP****, the placement scheme for demand VMs of flavor i can be obtained by randomly selecting n_i VMs from optimal $\hat{x}_{ik}^{q}$.*

We mention that **I-NAVMP** is a type of the MMMK problem. For the classical MMMK problem, there are several knapsacks (servers) with different multi-dimensional resource configurations, and each item (VM in the NUMA-aware form) will be selected in one of a set of incarnations. The goal is to select incarnations of items, and to assign them to the given knapsacks, so that the overall profit can be maximized. As to **I-NAVMP**, a new developed MMMK variant, it possesses an infinite number of items, and requires a lower limit (n_i) for the number of items need to be packed in each type. Additionally, following the typical characteristics of the knapsack problem, the proposed **I-NAVMP** is NP-hard.

Theorem 1: ***I-NAVMP*** *is NP-hard.*

Proof: By setting $n_i = 0$ for all $i = 1, \cdots, n$, constraints (7) will be relaxed. Thus, **I-NAVMP** can be reduced to K individual multi-dimensional integer knapsack problems, which are well-known NP-hard [44]. Obviously, this reduction can be completed in polynomial time. Therefore, **I-NAVMP** is also an NP-hard problem. □

Remark 3: i) Compared with the existing NAVMP models (e.g., in [6]), our ***I-NAVMP*** *formulation connotatively contains the complex NUMA-aware VM deployment rules by using the incarnation sets. Therefore, the NUMA-related constraints of* ***I-NAVMP*** *have a more concise form, which is favorable for us to design efficient algorithms.*

ii) ***I-NAVMP*** *is formulated from the perspective of flavors' incarnations rather than individual VMs as in [6], [29], and [45], thus resulting in a pure IP instead of a 0-1 program. In this situation, the scale of* ***I-NAVMP*** *will not increase with the number of VMs since the incarnation number is a constant. This strategy may help alleviate the computation burden of* ***I-NAVMP*** *with a large number of VMs.*

IV. CG-Based Offline Optimization Approach

According to Theorem 1, **I-NAVMP** is generally with exponential computation complexity. In practical instances, when the problem scale becomes large, it could be very difficult to derive the optimal solution within acceptable amount of time. To handle the computational challenge in the offline optimization, we tailor an efficient decompositon approach based on Column Generation.

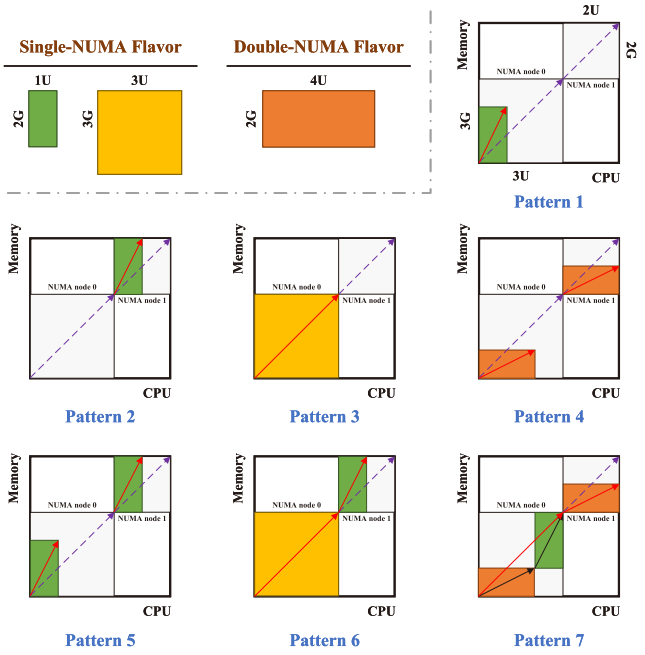


Fig. 5. An example for NUMA-aware placement patterns. Consider a double-NUMA system, including single-NUMA flavors as (1U,2G) and (3U,3G), as well as double-NUMA flavor as (4U,2G). The server type (5U,5G) is comprised of two NUMA nodes (3U,3G) and (2U,2G). There exist seven patterns in total, and each placement scheme of a batch of VMs can be yielded by a combination of these patterns.

A. Pattern-Based Reformulation

Although there might be a tremendous number of VMs and servers, we observe that the number of flavors (and thus incarnations) as well as the number of server types are both constants. Due to a limited variety of flavors, for a given server type, all the schemes that the flavors can be placed are also finite. So, they can be predefined in advance. Based on this intuitive observation, we first propose *NUMA-aware placement patterns* for various heterogeneous server types.

Definition 2: (NUMA-Aware Placement Pattern) For server type s, a NUMA-aware placement pattern is defined as $\boldsymbol{\xi}_s^t = (\xi_{1s}^{1,t}, \cdots, \xi_{1s}^{Q_1,t}, \cdots, \xi_{Ns}^{1,t}, \cdots, \xi_{Ns}^{Q_N,t})^T$ *for* $t = 1, \cdots, T_s$, *such that*

$$\sum_{i=1}^{N}\sum_{q=1}^{Q_i} r_i^q(\gamma)\xi_{is}^{q,t} \le R_s(\gamma), \quad \gamma = 1, \cdots, \Gamma \tag{14}$$

$$\xi_{is}^{q,t} \in \mathbb{N}, \quad i = 1, \cdots, N;\ q = 1, \cdots, Q_i \tag{15}$$

The patterns for server type s correspond to the feasible solutions of problem (14)-(15) one-to-one. Fig. 5 presents an example, explaining the patterns for a given server type. In brief, a pattern represents a VM placement scheme, indicating which flavors and how many VMs of these flavors are placed together on specific NUMA nodes of a server type.

Remark 4: There are quite some comcepts in this paper. To improve the readibility, we list the relationship among "flavor", "incarnation" and "NUMA-aware placement pattern".

- *"Flavor" represents a type of VMs with the same amount of various resources.*
- *"Incarnation" is the NUMA-aware extension of flavors. It subtly implies the VM deployment scheme on NUMA nodes by increasing vector dimensions.*
- *"NUMA-aware placement pattern" delegates a VM placement scheme of a certain server type considering the NUMA architecture. It is a combination of various incarnations in varying numbers.*

The placement scheme of a batch of VMs can be obtained by a combination of these patterns. By employing NUMA-aware placement patterns, **I-NAVMP** can be equivalently reformulated as a pattern-based problem (**P-NAVMP**):

P − NAVMP :

$$\phi := \max_{\lambda_s^t} \sum_{s=1}^{S}\sum_{i=1}^{N}\sum_{t=1}^{T_s}\sum_{q=1}^{Q_i} \omega_i \lambda_s^t \xi_{is}^{q,t} \tag{16}$$

$$\text{s.t.} \sum_{s=1}^{S}\sum_{t=1}^{T_s}\sum_{q=1}^{Q_i} \lambda_s^t \xi_{is}^{q,t} \ge n_i, \quad i = 1, \cdots, N \tag{17}$$

$$\sum_{t=1}^{T_s} \lambda_s^t = u_s, \quad s = 1, \cdots, S \tag{18}$$

$$\lambda_s^t \in \mathbb{N}, \quad s = 1, \cdots, S;\ t = 1, \cdots, T_s \tag{19}$$

where the objective (16) is also to maximize the resource value. Constraints (17) describe the VM request satisfaction from the perspective of pattern combinations. Besides, constraints (18) restrict the pattern usage of each server type in terms of the actual server possession.

P-NAVMP is a set-covering problem. The number of placement patterns (and thus the decisoin variables) grows exponentially with the increase of the problem scale. Therefore, it is impractical to enumerate all patterns for a large-scale **P-NAVMP**. Actually, this type of problem is well suited to be addressed by CG [34], [46], the main idea of which is to provide a solution mechanism only considering a portion of patterns, and iteratively add new promising patterns to augment the solution quality. The iterative procedure will terminate once it obtains the optimal or a high-quality feasible solution. Based on the classical framework of CG, we customize a primal-dual decomposition approach to support the efficient computation of **P-NAVMP**.

B. CG Decomposition

The CG algorithm is powerful to deal with the linear program (LP) with a substantial number of columns (corresponding to the placement patterns in **P-NAVMP**). It decomposes an LP into a restricted master problem and a (or several) pricing subproblem(s) to gain the optimal solution.

We consider the LP relaxation of **P-NAVMP** to form an LP master problem (**LMP**) as below (this relaxation will be elaborated in Section IV-E), and then introduce CG to address the structure of a mass of columns.

LMP :

$$\phi^{\text{LP}} := \max_{\lambda_s^t} \sum_{s=1}^{S}\sum_{i=1}^{N}\sum_{t=1}^{T_s}\sum_{q=1}^{Q_i} \omega_i \lambda_s^t \xi_{is}^{q,t} \tag{20}$$

$$\text{s.t.} \quad (17) - (18) \tag{21}$$

$$\lambda_s^t \ge 0, \quad s = 1, \cdots, S;\ t = 1, \cdots, T_s \tag{22}$$

The iterative procedures of CG begin with the computing of a restricted master problem (**RMP**) using part of the NUMA-aware placement patterns:

RMP :

$$\max_{\lambda_s^t} \sum_{s=1}^{S}\sum_{i=1}^{N}\sum_{t\in\mathcal{P}_s^\tau}\sum_{q=1}^{Q_i} \omega_i \lambda_s^t \xi_{is}^{q,t} \tag{23}$$

$$\text{s.t.} \quad \sum_{s=1}^{S}\sum_{t\in\mathcal{P}_s^\tau}\sum_{q=1}^{Q_i} \lambda_s^t \xi_{is}^{q,t} \geq n_i \rightarrow (\pi_i),\ i=1,\cdots,N \tag{24}$$

$$\sum_{t\in\mathcal{P}_s^\tau} \lambda_s^t = u_s \rightarrow (\zeta_s),\ s=1,\cdots,S \tag{25}$$

$$\lambda_s^t \geq 0, \quad s=1,\cdots,S;\ t\in\mathcal{P}_s^\tau \tag{26}$$

where $\pi_i \geq 0 (i=1,\cdots,N)$ and $\zeta_s (s=1,\cdots,S)$ are the Lagrangian multipliers corresponding to constraints (24) and (25) respectively. $\mathcal{P}_s^\tau$ denotes the subset of placement patterns for server type s in iteration τ. For the patterns that are not in $\mathcal{P}_s^\tau$, $s=1,\cdots,S$, we can treat the corresponding $\lambda_s^t = 0$. Hence, **RMP** actually provides a feasible solution to **LMP**, and the objective value of which gives a lower bound (LB) to **LMP**.

Collecting the dual information $\hat{\pi}_i (i=1,\cdots,N)$ and $\hat{\zeta}_s (s=1,\cdots,S)$ from **RMP**, then, S pricing subproblems are solved in sequence to generate new promising placement patterns. Each subprobelm corresponds to a server type, and the s-th subproblem ($\mathbf{SP}_s$) is defined as

$\mathbf{SP}_s$:

$$\min_{x_{ik}^q,\phi_s} \phi_s \tag{27}$$

$$\text{s.t. } \phi_s = \hat{\zeta}_s - \sum_{i=1}^{N}\sum_{q=1}^{Q_i} x_{is}^q(\hat{\pi}_i + \omega_i) \tag{28}$$

$$\sum_{i=1}^{N}\sum_{q=1}^{Q_i} r_i^q(\gamma) x_{is}^q \leq R_s(\gamma), \quad \gamma=1,\cdots,\Gamma \tag{29}$$

$$x_{is}^q \in \mathbb{N}, \quad i=1,\cdots,N;\ q=1,\cdots,Q_i \tag{30}$$

$\mathbf{SP}_s$ is a Γ-dimensional integer knapsack problem [44] with a quite small scale. In particular, it only has $\sum_{i=1}^{N} Q_i$ integer variables in total, and the number will not change. By solving all the subproblems, an upper bound (UB) can be gained as below.

$$UB = \sum_{s=1}^{S} u_s(\hat{\zeta}_s - \hat{\phi}_s) - \sum_{i=1}^{N} n_i \hat{\pi}_i \tag{31}$$

Note that **RMP** and the subproblems will be iteratively solved until the approximate gap ($\frac{UB-LB}{LB}$) is less than a predefined precision level ε (e.g., 10^{-2} or 10^{-3}). Otherwise, if $\exists s = 1,\cdots,S$ such that $\hat{\phi}_s < 0$, we render $(\hat{x}_{1s}^1,\cdots,\hat{x}_{1s}^{Q_1},\cdots,\hat{x}_{Ns}^1,\cdots,\hat{x}_{Ns}^{Q_N})^T$ to be a new promising NUMA-aware placement pattern, gathering which we will update $\mathcal{P}_s^\tau \rightarrow \mathcal{P}_s^{\tau+1}$.

C. Initialization and Paralleled Iterative Procedures

The selection of initial pattern may evidently affect the feasibility and efficiency of our CG algorithm. So, before proceeding to the detailed procedures of CG customization, we develop an initialization strategy for the placement pattern subsets. For each $s=1,\cdots,S$, $\mathcal{P}_s^1$ is initialized with $\sum_{i=1}^{N} Q_i$ elements, and p-th element $\boldsymbol{\xi}_s^p$ is a $\sum_{i=1}^{N} Q_i$-dimensional vector with all components zero except for the p-component

$$\boldsymbol{\xi}_s^p = (0,\cdots,0,\xi_{J_s}^{W,p},0,\cdots,0)^T, \tag{32}$$

where J satisfies $\sum_{i=1}^{J-1} Q_i < p \leq \sum_{i=1}^{J} Q_i$, as well as $W = p - \sum_{i=1}^{J-1} Q_i$. And $\xi_{J_s}^{W,p}$ is defined as

$$\xi_{J_s}^{W,p} := \arg\max_{x_{is}^W} \left\{\omega_i x_{is}^W \,|\, r_i^W(\gamma) x_{is}^W \leq R_s(\gamma),\ \gamma=1,\cdots,\Gamma;\ x_{is}^W \in \mathbb{N}\right\} \tag{33}$$

Based on the development of **RMP**, the subproblems and the initialization scheme, the complete steps of our customized CG algorithm are outlined in Fig. 6. Specifically, the subproblems ($\mathbf{SP}_s$) corresponding to the server types are independent with each other. Therefore, the parallelization can be applied to improve the performance of CG, as exhibited in the dashed red box of Fig. 6.

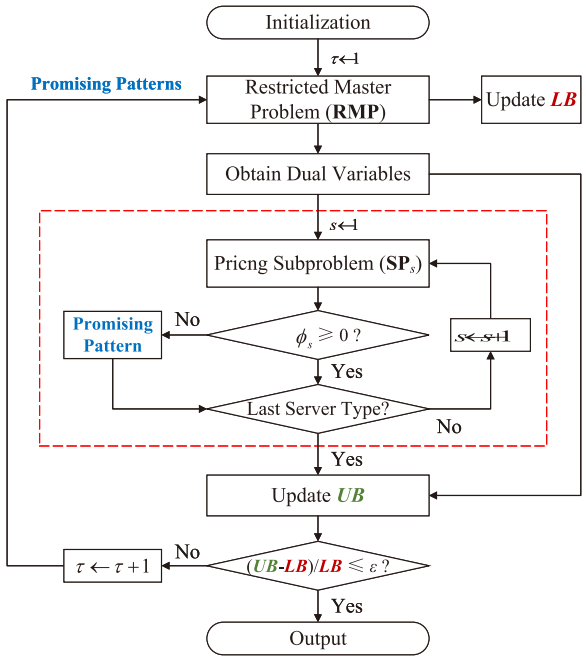


Fig. 6. Flow chart of the customized CG algorithm.

D. Heuristic for Feasible Solution

Using the CG algorithm, we can gain the (continuous) optimal solution to **LMP**, which, however, could be infeasible to **P-NAVMP** (and thus **I-NAVMP**) due to the LP relaxation. Therefore, a heuristic strategy is designed to recover a feasible (integer) solution to the original **I-NAVMP** problem.

The basic idea is given as below. Suppose $\tilde{\tau}$ is the final iteration number of CG. Firstly, we calculate $\lfloor \hat{\lambda}_s^t \rceil$ for each $s=1,\cdots,S, t=1,\cdots,|\mathcal{P}_s^{\tilde{\tau}}|$, where $\lfloor\cdot\rceil$ denotes the Gauss function and satisfies $\lfloor z \rceil = \jmath$, s.t. $\jmath \leq z < \jmath+1, z\in\mathbb{R}, \jmath\in\mathbb{Z}$.

Using the integer part of $\hat{\lambda}_s^t$, the placement scheme of VMs based on patterns can be determined. Then, the unsatisfied request number of flavor i $(i = 1, \cdots, N)$ can be gained by

$$\overline{n}_i = \max\left\{0, n_i - \sum_{s=1}^{S}\sum_{t\in\mathcal{P}_s^{\bar{\tau}}}\sum_{q=1}^{Q_i}[\hat{\lambda}_s^t]\xi_{is}^{q,t}\right\} \tag{34}$$

and the remaining server number of each type s $(s = 1, \cdots, S)$ corresponding to the fraction part of λ_s^t can be obtained by

$$\overline{u}_s = u_s - \sum_{t\in\mathcal{P}_s^{\bar{\tau}}}[\hat{\lambda}_s^t] \tag{35}$$

Thus, the total number of the remaining servers are

$$\overline{K} = \sum_{s=1}^{S}\overline{u}_s \tag{36}$$

and their corresponding resource capacities are

$$\overline{R}_k(\gamma) = R_s(\gamma), \quad \gamma = 1, \cdots, \Gamma;\ k \in \Omega_s;\ k = 1, \cdots, \overline{K};\ s = 1, \cdots, S \tag{37}$$

Finally, by gathering the information of the unsatisfied flavors and the remaining servers, we update the parameters of **I-NAVMP** through the following process:

$$n_i \leftarrow \overline{n}_i, \quad i = 1, \cdots, N \tag{38}$$

$$K \leftarrow \overline{K} \tag{39}$$

$$R_k(\gamma) \leftarrow \overline{R}_k(\gamma), \quad \gamma = 1, \cdots, \Gamma;\ k = 1, \cdots, K \tag{40}$$

and solve the updated **I-NAVMP** directly to derive the corresponding integer placement solution. By combining the integer solution of **P-NAVMP** and the placement results of the updated **I-NAVMP**, we will recover a feasible solution of the original IP formulation.

Compared to the original **I-NAVMP** problem, the updated **I-NAVMP** has much less number of servers ($\overline{K}$) and unsatisfied flavors ($\overline{n}_i$). Usually, the updated **I-NAVMP** can be solved directly through a professional IP solver. But there may exist some special instances that the scale of updated **I-NAVMP** is still large. An intuitive heuristic follows that we can reformulate the updated **I-NAVMP** using placement patterns and resolve the resulting updated **P-NAVMP** using CG again to further reduce the problem scale. In practice, we can develop an additional *outer loop* algorithm for problem scale reduction. Also, before updating the parameters of **I-NAVMP**, we can use the criterion as below to check whether to solve an updated **I-NAVMP** or enter the outer loop

$$K \le \Delta \tag{41}$$

where Δ is a predefined threshold. If (41) satisfies, we will break the outer loop and solve updated **I-NAVMP** directly.

In summary, Fig. 7 depicts the whole flow chart of our CG-based offline optimization approach. The dashed red box indicates the outer-loop algorithm.

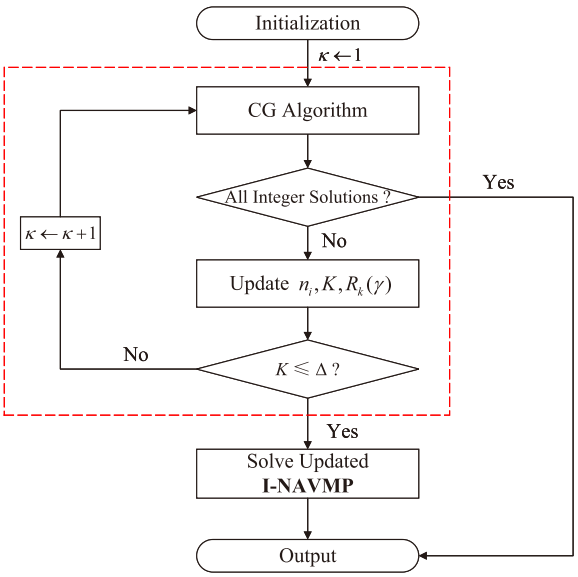


Fig. 7. Flow chart of our offline optimization approach.

E. Relationship With Lagrangian Dual

Consider the Lagrangian relaxation [47] of **I-NAVMP**, which is derived by dualizing the request satisfaction constraints (7) using nonnegative multipliers $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_N)^T$. The corresponding Lagrangian relaxation is given by

$$L(\boldsymbol{\mu}) = \max_{x_{ik}^q}\sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{q=1}^{Q_i}\omega_i x_{ik}^q + \sum_{i=1}^{N}\mu_i\Big(\sum_{k=1}^{K}\sum_{q=1}^{Q_i}x_{ik}^q - n_i\Big) \tag{42}$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\sum_{q=1}^{Q_i}r_i^q(\gamma)x_{ik}^q \le R_k(\gamma), \quad k = 1, \cdots, K;\ \gamma = 1, \cdots, \Gamma \tag{43}$$

$$x_{ik}^q \in \mathbb{N}, \quad i = 1, \cdots, N; \quad q = 1, \cdots, Q_i;\ k = 1, \cdots, K \tag{44}$$

$$\mu_i \ge 0, \quad i = 1, \cdots, N \tag{45}$$

and its dual problem is

$$\psi^{\mathrm{LD}} := \inf_{\boldsymbol{\mu}\ge 0} L(\boldsymbol{\mu}) \tag{46}$$

Proposition 1: $\psi^{\mathrm{LD}} = \phi^{\mathrm{LP}}$.

Proof: According to the server types, $L(\boldsymbol{\mu})$ can be decomposed to

$$L(\boldsymbol{\mu}) = \sum_{s=1}^{S}u_s L_s(\boldsymbol{\mu}) - \sum_{i=1}^{N}\mu_i n_i \tag{47}$$

where

$$L_s(\boldsymbol{\mu}) = \max_{x_{is}^q}\sum_{i=1}^{N}\sum_{q=1}^{Q_i}(\omega_i + \mu_i)x_{is}^q \tag{48}$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\sum_{q=1}^{Q_i}r_i^q(\gamma)x_{is}^q \le R_s(\gamma), \quad \gamma = 1, \cdots, \Gamma \tag{49}$$

$$x_{is}^{q} \in \mathbb{N}, \quad i = 1, \cdots, N; \ q = 1, \cdots, Q_i \tag{50}$$

$$\mu_i \geq 0, \quad i = 1, \cdots, N \tag{51}$$

Note that for the servers in the same server type, they possess the same resource capacity $R_s(\gamma)$, thus resulting in the same optimal x_{ik}^{q}. Hence, with a little notation abuse, we use x_{is}^{q} denoting all x_{ik}^{q} with the same server type s.

Let $\mathcal{X}_s$ be the feasible region defined by (49)-(50), and $\boldsymbol{\xi}_s^t = (\xi_{1s}^{1,t}, \cdots, \xi_{1s}^{Q_1,t}, \cdots, \xi_{Ns}^{1,t} \cdots, \xi_{Ns}^{Q_N,t})^T$ for all $t \in \{1, \cdots, T_s\}$ represent all the feasible points in $\mathcal{X}_s$. Then, we have

$$L_s(\boldsymbol{\mu}) = \max_{t=1,\cdots,T_s} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} (\omega_i + \mu_i) \xi_{is}^{q,t} \tag{52}$$

Thus, the dual problem can be expressed a

$$\inf_{\boldsymbol{\mu} \geq 0} L(\boldsymbol{\mu}) \tag{53}$$

$$= \min_{\boldsymbol{\mu} \geq 0} \sum_{s=1}^{S} u_s [\max_{t=1,\cdots,T_s} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} (\omega_i + \mu_i) \xi_{is}^{q,t}] - \sum_{i=1}^{N} \mu_i n_i \tag{54}$$

$$= \min_{\boldsymbol{\mu}, \boldsymbol{\eta}} \sum_{s=1}^{S} u_s \eta_s - \sum_{i=1}^{N} \mu_i n_i \tag{55}$$

$$\text{s.t. } \eta_s \geq \sum_{i=1}^{N} \sum_{q=1}^{Q_i} (\omega_i + \mu_i) \xi_{is}^{q,t} \rightarrow (\lambda_s^t)$$

$$t = 1, \cdots, T_s; \ s = 1, \cdots, S \tag{56}$$

$$\mu_i \geq 0, \quad i = 1, \cdots, N, \tag{57}$$

where $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_S)$ is the auxiliary variable, and $\lambda_s^t \geq 0$ is the multiplier of constraints (56). Note that problem (55)-(57) is an LP, exhibiting strong duality. Its dual problem is

$$\max_{\lambda_s^t} \sum_{s=1}^{S} \sum_{i=1}^{N} \sum_{t=1}^{T_s} \sum_{q=1}^{Q_i} \omega_i \lambda_s^t \xi_{is}^{q,t} \tag{58}$$

$$\text{s.t.} \sum_{s=1}^{S} \sum_{t=1}^{T_s} \sum_{q=1}^{Q_i} \lambda_s^t \xi_{is}^{q,t} \geq n_i, \quad i = 1, \cdots, N \tag{59}$$

$$\sum_{t=1}^{T_s} \lambda_s^t = u_s, \quad s = 1, \cdots, S \tag{60}$$

$$\lambda_s^t \geq 0 \quad t = 1, \cdots, T_s; \ s = 1, \cdots, S, \tag{61}$$

which is equivalent to **LMP** (20)-(22), and $\psi^{\text{LD}} = \phi^{\text{LP}}$ holds. This completes the proof. □

Following Proposition 1, we actually obtain the objective value of a Lagrangian dual problem of **I-NAVMP** by solving an LP relaxation of **P-NAVMP**. The next proposition indicates the tightness of this relaxation.

Proposition 2: $\psi^{\text{LP}} \geq \phi^{\text{LP}} \geq \psi$.

Proof: Due to Proposition 1, we only have to prove $\psi^{\text{LP}} \geq \psi^{\text{LD}} \geq \psi$.

Denote $\boldsymbol{x} = (\boldsymbol{x}_1^T, \cdots, \boldsymbol{x}_N^T)^T$, and for each $i = 1, \cdots, N$, $\boldsymbol{x}_i = (x_{i1}^1, \cdots, x_{i1}^{Q_i}, \cdots, x_{iK}^1, \cdots x_{iK}^{Q_i})^T$. It is well-known that the objective value of the dual problem in the IP setting is equal to a special LP relaxation of the primal problem [47].

TABLE I

FLAVOR INFORMATION FOR ILLUSTRATIVE EXAMPLE

Flavor	NUMA Type	CPU (U)	Memory (GB)	α_i	Worth*	Demands
I	single	1	1	1	14	2
II	single	1	2	1	22	3
III	single	2	2	1	29	3
IV	single	2	4	2	89	4
V	double	4	8	4	358	1

* Flavors' worth in the table is fine-tuned based on (65). Take an instance for explanation. With the same α, the worth of flavor III should be larger than two times the worth of flavor I. This is because if it is possible to satisfy the resource request for one VM with (2U,2G) configuration, then it must be possible to satisfy two VMs with (1U,1G) as well.

In particular, we have

$$\psi^{\text{LD}} = \max_{x_{ik}^q} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{q=1}^{Q_i} \omega_i x_{ik}^{q} \tag{62}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \text{conv}(\overset{N}{\underset{i=1}{\times}} \{\boldsymbol{x}_i \in \mathbb{N}^{KQ_i \times 1} | \mathbf{1}^T \boldsymbol{x}_i \geq n_i\}) \tag{63}$$

$$\sum_{i=1}^{N} \sum_{q=1}^{Q_i} r_i^q(\gamma) x_{ik}^{q} \leq R_k(\gamma),$$

$$k = 1, \cdots, K; \ \gamma = 1, \cdots, \Gamma \tag{64}$$

where conv(·) represents the convex hull of a feasible set. Because $\text{conv}(\overset{N}{\underset{i=1}{\times}} \{\boldsymbol{x}_i \in \mathbb{N}^{KQ_i \times 1} | \mathbf{1}^T \boldsymbol{x}_i \geq n_i\}) \subseteq \overset{N}{\underset{i=1}{\times}} \{\boldsymbol{x}_i \in \mathbb{R}^{KQ_i \times 1} | \mathbf{1}^T \boldsymbol{x}_i \geq n_i\}$, $\psi^{\text{LP}} \geq \psi^{\text{LD}}$ holds.

As to $\psi^{\text{LD}} \geq \psi$, it obviously holds because ψ^{LD} is the objective value of an **I-NAVMP**'s relaxation problem. This completes the proof. □

*Remark 5: Proposition 2 indicates that **LMP** provides a better relaxation bound to **I-NAVMP** from the Lagrangian dual than directly applying its LP relaxation. A tighter relaxation possesses more valuable information, which is important to construct a high-quality feasible solution. Also, the tightness of relaxation bounds is necessary for the feasible solution evaluation. As an aside, the results in [47] demonstrates that the bounds obtained by Lagrangian relaxation are extremely sharp for many specific combinatorial optimization problems (including the knapsack problem), which provides the numerical basis for our approach to deal with the MMMK problem.*

V. ILLUSTRATIVE EXAMPLE

An example is presented to demonstrate the effectiveness of the proposed NAVMP model. We consider a double-NUMA system including two NUMA resource types, i.e., CPU and memory ($\Gamma = 4$). The information of these flavors is given in TABLE I. Also, the worth of flavor i can be defined as:

$$Worth_i = 10\alpha_i \cdot \sqrt{(r_i^{\text{CPU}})^2 + (r_i^{\text{Mem}})^2}, \tag{65}$$

where $r_i^{\text{CPU}}/r_i^{\text{Mem}}$ is the CPU/memory request of flavor i (in the NUMA-oblivious form). $\alpha_i \in \{\cdots, -1, 1, 2, \cdots\}$ is the importance degree of flavors, which is pre-specified by the cloud service provider. Six servers are considered with (4U,6G,4U,6G) configuration (in the NUMA-aware extensive form). Three of them are unoccupied, while the rest ones have residual capacities of (3U,5G,3U,5G).

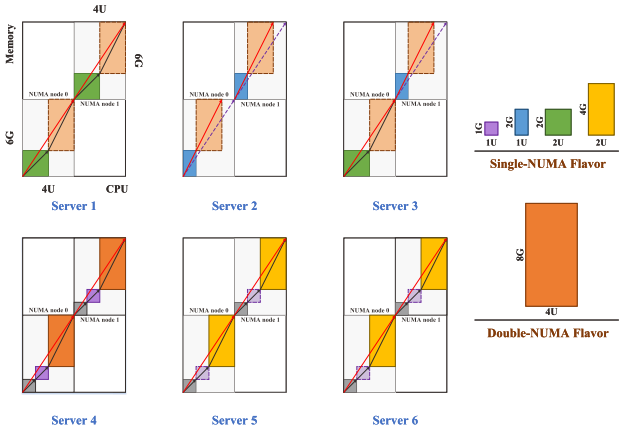


Fig. 8. NAVMP scheme of the example. Demand and imaginary VMs are represented by solid and dashed rectangulars, respectively. The dark grey rectangulars represents the resources that have been assigned for running VMs.

TABLE II
FLAVOR INFORMATION FROM PRACTICAL DATASET

Flavor	NUMA Type	CPU (U)	Memory (GB)	α	Worth*	Demand Distribution
A	single	1	1	1	10	6.63%
B	single	1	2	1	21	42.36%
C	single	2	4	1	44	0.92%
D	single	4	8	1	89	7.31%
E	single	8	16	-1	-178	1.76%
F	single	1	4	1	41	3.87%
G	single	2	8	1	83	0.80%
H	single	4	16	1	167	1.98%
I	single	8	32	-1	-330	0.65%
J	single	6	4	1	72	1.08%
K	single	12	8	1	145	2.31%
L	single	24	16	2	577	1.13%
M	single	36	24	1	436	0.37%
N	single	48	32	3	1731	1.44%
O	single	72	48	1	873	0.13%
P	single	96	64	3	3463	15.63%
Q	double	144	96	1	1747	10.42%
R	double	192	128	4	9230	1.21%

* The same as TABLE I, flavors' worth is fine-tuned based on (65).

Accordingly, the optimal solution of NAVMP can be attained as in Fig. 8. It can be observed that the decision on VM placement tends to fill up the servers, following the objective of VPA. Particularly, 2, 3, 3, 4 and 1 VMs of Flavor I, II, III, IV and V are randomly selected to be the demand VMs, respectively. Others are treated as imaginary VMs, and the corresponding resources are reserved for the future VM requests. In this case, the imaginary VMs are in Flavor I and V. Flavor V is the dominated flavor, i.e., if a VM in Flavor V can be assigned on a server, the other flavors can also be placed on that server. Note that the strategy of selecting demand VMs is rather flexible. When the server resources are sufficient, the VMs on the activated servers can be selected as demand VMs preferentially to take count of the traditional target (e.g., minimizing the number of activated servers) simultaneously. These observations indicate the flexibility and validity of our VM placement strategy, which well prepares for potential users' requests by yielding a rational reserve plan.

VI. NUMERICAL STUDIES

A. Description

This section conducts numerical experiments to verify the effctiveness of our NAVMP model and CG-based approach. These experiments are based on a practice dataset from Huawei Cloud, which belongs to the double-NUMA system. Two types of NUMA resources, namely CPU and memory, are taken into consideration. TABLE II shows the flavor information, and gives the flavor distribution of the 20000 demand VMs in the experiments. It can be seen that the distribution of different flavors is extremely uneven. For instance, Flavor B has a 42.36% share, while Flavor O is only 0.13%. Also, we choose the server with (276U,324G) configuration, whose NUMA-aware extension form is (138U,162G,138U,162G), and the total number of servers is 30000. The initial states of the servers results in different server types. We pack the demand VMs and servers into several instances for testing.

All the tests are performed on a desktop computer with Intel Core i7-9700T 2.00GHz processor and 16GB main memory. The open-source COIN-OR Branch-and-Cut (Cbc) solver [48] is applied to solve both IPs and LPs. CG terminates if the criterion is satisfied ($\varepsilon = 0.01\%$) or the iteration limit (30) is reached, and we set $\Delta = 10$.

B. Computational Results

All computational results using CG are presented in TABLE III. The solutions obtained by using the popular genetic algorithm (GA) [49], [50], and directly applying the Cbc solver (CBC) are given as the benchmark. We integrate 100 individuals for GA implementation, and include a uniform crossover operator [51] to improve its performance. The GA is terminated when the objective becomes stable. In TABLE III, Column 1-3 indicate the number of demand VMs (M), servers (K) and server types (S), respectively. For simplicity, let $\mathscr{T}_{(M,K)}$ denote the test instance with M demand VMs and K servers. Column 4 lists the evaluation bound (EB) from CG, which is the UB of the first outer loop. Column 5-8 exhibit the feasible bound (FB), optimality gap (Gap), number of outer loops (κ), and execution time (Time) by adopting CG. Column 9-11 show the objective value (Obj), optimality gap (Gap) and solution time (Time) of GA. Column 12-14 give the corresponding computational performance data of CBC, where "Obj", "Gap" and "Time" have the same meaning of GA's. For each algorithm, Gap is calculated by

$$Gap = \frac{EB - Value}{EB} \times 100\%, \tag{66}$$

where $Value$ represents FB or Obj derived by CG, GA or CBC. Additionally, the time limit is set to 7200 seconds (2 hours). Note that the results in TABLE III refer to average performances over five independent instances.

Together with the comparative studies presented in Table III, the following observations and insights are derived:

1) Our CG-based approach provides similar solutions as CBC, but outperforms CBC in the computation time. For all test instances, CG has the ability to construct high-quality solutions (Gap $\leq$ 0.01%) in reasonable amount of time, demonstrating a strong and scalable solution capacity across **I-NAVMP** with different sizes. Particularly, it takes no more than 11 minutes to address some very challenging instances (e.g., $\mathscr{T}_{(20000,30000)}$ with

TABLE III
RESULTS OF COMPUTATIONAL EXPERIMENTS

M	K	S	EB	CG				GA			CBC		
				FB	Gap	κ	Time (s)	Obj	Gap	Time (s)	Obj	Gap	Time (s)
30	30	20	260971	260944	0.01%	1	5.95	224505	13.97%	2.92	260962	$<0.01\%$	28.26
300	80	44	557399.75	557353	$<0.01\%$	1	31.78	480653	13.77%	4.87	557360	$<0.01\%$	257.01
300	1000	196	10983329	10983307	$<0.01\%$	1	57.75	10269766	6.50%	37.15	10983309	$<0.01\%$	1764.51
300	3000	639	32567241	32566520	$<0.01\%$	1	114.81	30907681	5.10%	94.33	32566438	$<0.01\%$	>7200
300	5000	708	54064479	54063125	$<0.01\%$	1	149.05	51492083	4.76%	170.21	NA*	/	>7200
300	10000	799	108269784	108269784	0	1	164.81	103323728	4.57%	352.49	NA	/	>7200
300	20000	865	216163320	216163320	0	1	175.88	206635099	4.41%	646.87	NA	/	>7200
300	30000	998	326235333	326235333	0	1	180.48	311928918	4.39%	963.19	NA	/	>7200
3000	3000	639	31055796.7	31053012	$<0.01\%$	1	282.96	28494378	8.25%	85.25	NA	/	>7200
3000	5000	708	52952330.5	52950250	$<0.01\%$	1	265.88	49046933	7.38%	160.64	NA	/	>7200
3000	10000	799	107425557	107423097	$<0.01\%$	1	167.99	100855370	6.12%	351.07	NA	/	>7200
3000	20000	865	215333865	215333865	0	1	179.18	204060808	5.24%	633.35	NA	/	>7200
3000	30000	998	325411478	325403619	$<0.01\%$	1	200.15	309416946	4.92%	951.48	NA	/	>7200
20000	20000	865	205434446	205421689	$<0.01\%$	1	319.77	188012451	8.48%	623.49	NA	/	>7200
20000	30000	998	316988115	316977455	$<0.01\%$	1	600.81	292268585	7.80%	933.94	NA	/	>7200

* NA means CBC cannot obtain a feasible solution within the time limitation.

1020000 integer variables and 1140018 constraints), which is practically desirable for an offline optimization problem. On the contrary, CBC can only handle small-scale problems (e.g., $\mathcal{T}_{(30,30)}$, $\mathcal{T}_{(300,80)}$, $\mathcal{T}_{(300,1000)}$) due to the NP-hard nature. Even for the instances with a moderate size, it fails to yield a feasible solution within the time restriction. For example, considering $\mathcal{T}_{(30,30)}$, $\mathcal{T}_{(300,80)}$ and $\mathcal{T}_{(300,1000)}$, CG exports comparable solution to CBC, but in less time. Especially for $\mathcal{T}_{(300,1000)}$, CG solves 30.55 times faster than CBC. As to instances range from $\mathcal{T}_{(300,5000)}$ to $\mathcal{T}_{(20000,30000)}$, CG can derive the 0.01% gap solution within 600.81s, while CBC even cannot find a feasible solution in 7200s.

2) We notice that CG dominates GA in the solution quality, and is faster for solving large-scale problems. It can be seen that CG generates high-quality solutions with a very small gap within 0.01% for all the instances, while the optimality gap of GA's solutions ranges from 4.39% to 13.97%. This may be due to the inherent flaw of meta-heuristic algorithms, which could easily trap GA into a local optimal solution. This drawback makes GA unsuitable for the offline optimization where the goal is to search for the global optimality. As to the solution time, even though GA could spend less time obtaining a feasible solution for some small- and middle-scale problems (i.e., $\mathcal{T}_{(30,30)}$, $\mathcal{T}_{(300,80)}$, $\mathcal{T}_{(300,1000)}$, $\mathcal{T}_{(300,3000)}$, $\mathcal{T}_{(3000,3000)}$, and $\mathcal{T}_{(3000,5000)}$), for the other larger-sclae instances, the convergence speed of CG can be 1.14-5.33 times faster.
3) We also perform the parametric analysis on the proposed CG approach. Generally, for a fixed M, the solution time increases with K, indicating that the solution efficiency is highly correlated with the problem size. For instance, it takes 57.75s, 114.81s and 1149.05s for $\mathcal{T}_{(300,1000)}$, $\mathcal{T}_{(300,3000)}$ and $\mathcal{T}_{(300,5000)}$, respectively. Among the instances in TABLE III, $\mathcal{T}_{(3000,3000)}$ and $\mathcal{T}_{(3000,5000)}$ are special cases. Their slow convergence might be caused by other factors. Besides, with the same K, CG spends longer time solving the instances with larger M (and thus n_i). Taking $\mathcal{T}_{(300,30000)}$, $\mathcal{T}_{(3000,30000)}$ and $\mathcal{T}_{(20000,30000)}$ for example, they need 180.48s, 200.15s and 600.81s to converge, respectively. Hence, the efficiency of CG is significantly affected by the problem parameters. Additionally, we would like to point out that **LMP** provides an extremely sharp EB for **I-NAVMP**. The gap between the optimal value (FB) and EB for all instances are no more than 0.01%. Also, the number of outer loops is equal to 1 for all tests, meaning that the optimal decisions to **LMP** are integer in the majority.

TABLE IV
TIME BREAKDOWN OF CG-BASED APPROACH

M	K	MP	SP	IP
30	30	25.07%	69.70%	5.23%
300	80	13.41%	66.56%	20.03%
300	1000	22.09%	77.91%	0*
300	3000	28.10%	71.50%	0.40%
300	5000	22.82%	76.88%	0.30%
300	10000	20.76%	79.24%	0
300	20000	26.86%	73.14%	0
300	30000	35.25%	64.75%	0
3000	3000	14.37%	77.50%	8.13%
3000	5000	15.48%	84.37%	0.15%
3000	10000	22.67%	77.08%	0.25%
3000	20000	22.11%	77.89%	0
3000	30000	25.32%	74.68%	0
20000	20000	18.12%	81.13%	0.75%
20000	30000	11.62%	88.24%	0.14%
Average		21.6%	76.04%	2.36%

* "0" in Column "IP" means the optimal solution of **LMP** are all integer. In that case, there is no need to solve the updated **I-NAVMP**.

In summary, the CG approach provides a powerful and scalable computation tool for the **I-NAVMP** problem.

TABLE IV gives the detailed time breakdown of the proposed offline algorithm. Columns "MP", "SP" and "IP" list the proportion of time spent on solving **RMP**, a series of **SP**$_s$, and

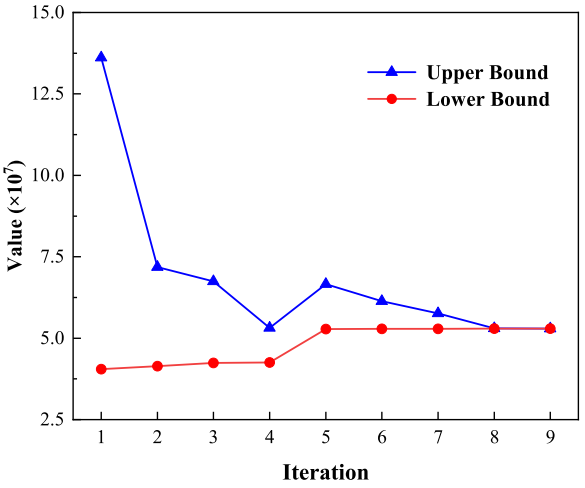


Fig. 9. Convergence behavior of CG for $\mathscr{T}_{(3000,5000)}$.

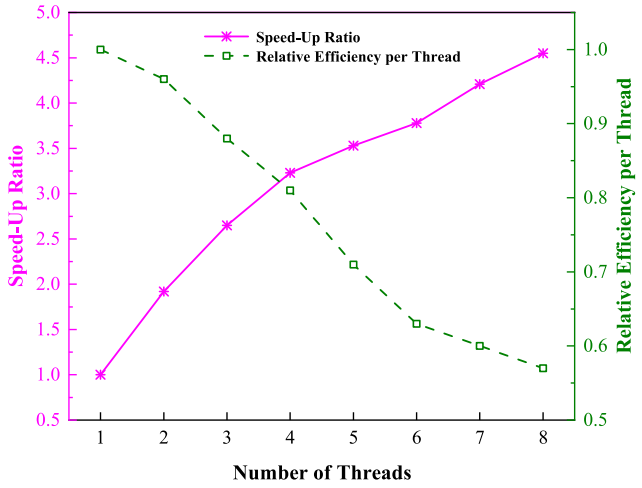


Fig. 10. Parallelization SUR and REPT with $\mathscr{T}_{(3000,30000)}$.

updated **I-NAVMP**, respectively. From the table, it can been observed that the major computation burden lies on the pricing subproblems. The time percentage of solving subproblems ranges from 64% to 89%, regardless of the number of servers. Except for $\mathscr{T}_{(300,80)}$, most of the remaining time in each test is consumed by solving the restrictive master problem. Besides, Column "IP" shows that half of the instances do not need to solve the updated **I-NAVMP**, which once again illustrates the tightness of the Lagrangian relaxation.

Fig. 9 depicts the typical convergence behavior of CG in one outer loop. It is observed that LB grows iteratively, because CG is actually a way to perform the Simplex method, which continuously improves the current solution of an LP untill achieving the global optimality. Also, as indicated by the change of UB and LB, our CG algorithm shows a desirable convergence performance for solving the NAVMP problem.

C. Parallelization Analysis

Let σ be the number of threads used in CG experiments, and $\varphi(\sigma)$ denote the related solution time. The *speed-up ratio (SUR)* is defined by

$$SUR = \frac{\varphi(1)}{\varphi(\sigma)}, \tag{67}$$

and the *relative efficiency per thread (REPT)* is given by

$$REPT = \frac{\varphi(1)}{\sigma\varphi(\sigma)}. \tag{68}$$

Fig. 10 exhibits the average SUR and REPT regarding the thread number for instance $\mathscr{T}_{(3000,30000)}$. As one might expect, SUR increases with the number of threads. The solution time for the instance using 8 threads is nearly 21.97% of that using 1. Yet, note that SUR is not proportional to the thread number due to the potential decrease of REPT. For example, REPT for the 8-thread case is 0.57 of the 1-thread case.

D. Comparative Studies With VBP-Based Model

TABLE V compares the VPA of the proposed MMMK-based model with traditional VBP-based one for NAVMP (please refer to the Appendix for more details). The scales of test instances (M, K) are listed in Column 1-2. Column 3

TABLE V
COMPARISON BETWEEN MMMK AND VBP MODEL

M	K	RRR	VPA_{MMMK}	VPA_{VBP}	VIR
30	8	67.52%	22210	15382	44.39%
30	10	51.87%	45745	36307	25.99%
30	100	5.05%	917402	892411	2.80%
30	1000	0.45%	11029967	11004004	0.24%
30	10000	0.05%	108304895	108286264	0.02%
300	64	75.03%	110216	96062	14.73%
300	100	44.24%	521320	479555	8.71%
300	1000	3.92%	10722672	10554253	1.60%
300	10000	0.40%	107951252	107885531	0.06%
3000	607	67.76%	1794609	1696259	5.80%
3000	10000	4.06%	104503796	102795531	1.66%
3000	20000	2.03%	212124706	210874082	0.59%

presents the resource request rate (RRR) of the demand VMs, which can be defined as

$$RRR = \max\left\{\frac{\sum_{i=1}^{N} r_i^{\text{CPU}} \cdot n_i}{\sum_{k=1}^{K} R_k^{\text{CPU}}}, \frac{\sum_{i=1}^{N} r_i^{\text{Mem}} \cdot n_i}{\sum_{k=1}^{K} R_k^{\text{Mem}}}\right\} \times 100\%, \tag{69}$$

where $r_i^{\text{CPU}}/r_i^{\text{Mem}}$ is the CPU/memory request of flavor i in the NUMA-oblivious form, and $R_k^{\text{CPU}}/R_k^{\text{Mem}}$ is the residual CPU/memory capacity of server k in the NUMA-oblivious form. Clearly, a larger RRR indicates higher proportion of demand VMs in a VM placement scheme. Column 4-5 present the VPA of the MMMK model (VPA_{MMMK}) and the VBP model (VPA_{VBP}), respectively. Correspondingly, we provide the VPA improvement rate (VIR) by applying the MMMK model, which is defined as

$$VIR = \frac{VPA_{\text{MMMK}} - VPA_{\text{VBP}}}{VPA_{\text{VBP}}} \times 100\%. \tag{70}$$

As shown in TABLE V, for all of the test instances, our MMMK model demonstrates a larger VPA than the VBP model. The VIR varies from 0.02% to 44.39%. For a fixed M, VIR generally decreases with the growing of K. For example, the VIRs of $\mathscr{T}_{(30,10)}$, $\mathscr{T}_{(30,100)}$ and $\mathscr{T}_{(30,1000)}$ are 25.99%, 2.80% and 0.24%, respectively. A possible reason is that for the same batch of demand VMs, more servers corresponds to smaller RRR. When RRR is relatively small, most clouding resources

is provided for imaginary VMs, so that the placement scheme of the demand VMs does not have a dominant effect on VPA. In such instances, the VPA metrics of MMMK and the traditional VBP model are relatively close. In other words, the proposed MMMK model could achieve a significant improvement of VPA especially for the extreme cases when available resources are scarce (i.e., RRR is very large), which further demonstrates the effectiveness of our formulation.

VII. Conclusion and Discussion

In this paper, we studied the classic VM placement problem in the multi-NUMA environment. An MMMK-based IP model was proposed to support NAVMP, which utilized incarnations to deal with the complex NUMA-aware VM placement rules, and adopted the objective that maximizes VPA of the resource pool. To alleviate the computational burden, a CG-based offline optimization approach was customized and implemented. Numerical experiments on a practical dataset from Huawei Cloud confirmed the validity of the NAVMP model and demonstrated the superiority of our CG-based approach. Some key observations and insights are presented as follows:

- *The MMMK-based model* copes well with the NAVMP problem. Compared to the traditional VBP-based model, it could increase VPA of the cloud resource pool by up to 44.39% among the test instances. Particularly, the proposed MMMK model could achieve a significant improvement of VPA for the extreme cases when available resources were scarce.
- *The CG-based decomposition approach* provides a powerful and scalacble tool to tackle the real-world NAVMP problem. Our experiments indicated that it could address the largest instance with 20000 VM demands and 30000 servers (involving 1020000 integer variables and 1140018 constraints) in 11 minutes, resulting in an optimality gap within 0.01%. Also, CG outperformed a professional IP solver (i.e., Cbc) as well as a prevalent meta-heuristic approach (i.e., GA) in both the solution quality and computation time, especially for large-scale cases.
- *The parallelization strategy* significantly augments the solution efficiency of the independent subproblems, which take up the major part of the CG process. For a certain instance with 3000 VM demands and 30000 servers, the solution speed with 8 threads was 4.55 times faster than the single-thread execution. But, the speed-up ratio was not proportional to the thread number. There exists a trade-off between solution efficiencies and thread acquisition costs.

However, the current CG-based approach is incompetent for the real-time resource allocation. How to design an efficient online algorithm for NAVMP based on the valuable information from the offline optimization needs a separate study. Also, we will conduct multi-objective optimization for NAVMP integrating energy consumption reduction and resource efficiency improvement. Furthermore, NAVMP can provide important decision boundaries for the capacity management of the cloud resource pool. How to model the capacity expansion coupling with NAVMP is also in our future work.

Appendix
VBP Model for NAVMP

The traditional VM placement model aims to minimize the number of activated servers. Following the incarnation-oriented model, NAVMP can be formulated as a multiple-choice VBP problem [43] as below:

$$\min_{x_{ik}^q, b_k} \sum_{k=1}^{K} b_k \tag{71}$$

$$\text{s.t.} \sum_{k=1}^{K} \sum_{q=1}^{Q_i} x_{ik}^q = n_i, \quad i = 1, \cdots, N \tag{72}$$

$$\sum_{i=1}^{N} \sum_{q=1}^{Q_i} r_i^q(\gamma) x_{ik}^q \le R_k(\gamma) b_k,$$
$$k = 1, \cdots, K; \ \gamma = 1, \cdots, \Gamma \tag{73}$$

$$x_{ik}^q \in \mathbb{N}, \quad i = 1, \cdots, N;$$
$$q = 1, \cdots, Q_i; \ k = 1, \cdots, K \tag{74}$$

$$b_k \in \{0, 1\}, \quad k = 1, \cdots, K \tag{75}$$

where b_k $(k = 1, \cdots, K)$ is a binary variable and represents the server state, which is "1" if server k is activated. Note that for server k which has been occupied by some running VMs, $b_k = 1$ exists.

Acknowledgment

The authors would like to thank Dr. Yuzhou Zhou, Yuexian Zhang, Yuchen Dong, Fangzhu Ming, and Haoran Zheng with Xi'an Jiaotong University for their selfless helps and valuable suggestions.

References

[1] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, and A. Y. Zomaya, "A multi-objective optimization scheme for job scheduling in sustainable cloud data centers," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 172–186, Jan. 2022.

[2] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *J. Netw. Comput. Appl.*, vol. 66, pp. 106–127, May 2016.

[3] J. Shi, J. Luo, F. Dong, J. Jin, and J. Shen, "Fast multi-resource allocation with patterns in large scale cloud data center," *J. Comput. Sci.*, vol. 26, pp. 389–401, May 2018.

[4] (2022). *ICPC 2022 Online Challenge Powered By HUAWEI*. [Online]. Available: https://codeforces.com/blog/entry/105097

[5] J. Rao, K. Wang, X. Zhou, and C.-Z. Xu, "Optimizing virtual machine scheduling in NUMA multicore systems," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2013, pp. 306–317.

[6] K. Hu, W. Lin, T. Huang, K. Li, and L. Ma, "Virtual machine consolidation for NUMA systems: A hybrid heuristic grey wolf approach," in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 569–576.

[7] R. Wu, X. Zhang, X. Kong, Y. Chen, R. Jnagal, and R. Hagmann, "Evaluation of NUMA-aware scheduling in warehouse-scale clusters," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 475–477.

[8] C. Lameter, "NUMA (Non-uniform memory Access): An overview: NUMA becomes more common because memory controllers get close to execution units on microprocessors," *Queue*, vol. 11, no. 7, pp. 40–51, Jul. 2013, doi: 10.1145/2508834.2513149.

[9] Y. Cheng, W. Chen, X. Chen, B. Xu, and S. Zhang, "A user-level NUMA-aware scheduler for optimizing virtual machine performance," in *Proc. Int. Workshop Adv. Parallel Process. Technol.* Cham, Switzerland: Springer, 2013, pp. 32–46.

[10] K. E. Batcher, "Design of a massively parallel processor," *IEEE Trans. Comput.*, vol. C-29, no. 9, pp. 836–840, Sep. 1980.

[11] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, "VM placement strategies for cloud scenarios," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 852–859.

[12] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1230–1242, Dec. 2013.

[13] M. Gaggero and L. Caviglione, "Model predictive control for energy-efficient, quality-aware, and secure virtual machine placement," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 420–432, Jan. 2019.

[14] P. Zhang, M. Zhou, and X. Wang, "An intelligent optimization method for optimal virtual machine allocation in cloud data centers," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1725–1735, Oct. 2020.

[15] W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, "A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 153–161, Jan. 2015.

[16] F.-H. Tseng, Y.-M. Jheng, L.-D. Chou, H.-C. Chao, and V. C. M. Leung, "Link-aware virtual machine placement for cloud services based on service-oriented architecture," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 989–1002, Oct. 2020.

[17] O. Biran et al., "A stable network-aware VM placement for cloud systems," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2012, pp. 498–506.

[18] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[19] A. Khosravi, L. L. H. Andrew, and R. Buyya, "Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 183–196, Apr. 2017.

[20] V. Kherbache, É. Madelaine, and F. Hermenier, "Scheduling live migration of virtual machines," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 282–296, Jan. 2020.

[21] B. Hu, Z. Cao, and M. Zhou, "Scheduling real-time parallel applications in cloud to minimize energy consumption," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 662–674, Jan. 2022.

[22] S. K. Mishra et al., "Energy-efficient VM-placement in cloud data center," *Sustain. Comput., Informat. Syst.*, vol. 20, pp. 48–55, Dec. 2018.

[23] H. Zhao et al., "Power-aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1385–1400, Jun. 2018.

[24] T. Abbasi-Khazaei and M. H. Rezvani, "Energy-aware and carbon-efficient VM placement optimization in cloud datacenters using evolutionary computing methods," *Soft Comput.*, vol. 26, no. 18, pp. 9287–9322, Sep. 2022.

[25] M. Xu and R. Buyya, "Managing renewable energy and carbon footprint in multi-cloud computing environments," *J. Parallel Distrib. Comput.*, vol. 135, pp. 191–202, Jan. 2020.

[26] Y. Cheng, W. Chen, Z. Wang, and X. Yu, "Performance-monitoring-based traffic-aware virtual machine deployment on NUMA systems," *IEEE Syst. J.*, vol. 11, no. 2, pp. 973–982, Jun. 2017.

[27] J. Sheng et al., "Learning to schedule multi-NUMA virtual machines via reinforcement learning," *Pattern Recognit.*, vol. 121, Jan. 2022, Art. no. 108254.

[28] M. Gabay and S. Zaourar, "Vector bin packing with heterogeneous bins: Application to the machine reassignment problem," *Ann. Oper. Res.*, vol. 242, no. 1, pp. 161–194, Jul. 2016.

[29] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," Res. Microsoft Com, Tech. Rep., 2011. [Online]. Available: https://www.microsoft.com/en-us/research/publication/heuristics-for-vector-bin-packing/

[30] N. Cherfi and M. Hifi, "A column generation method for the multiple-choice multi-dimensional knapsack problem," *Comput. Optim. Appl.*, vol. 46, no. 1, pp. 51–73, May 2010.

[31] L. Lamanna, R. Mansini, and R. Zanotti, "A two-phase kernel search variant for the multidimensional multiple-choice knapsack problem," *Eur. J. Oper. Res.*, vol. 297, no. 1, pp. 53–65, Feb. 2022.

[32] Y. Song, C. Zhang, and Y. Fang, "Multiple multidimensional knapsack problem and its applications in cognitive radio networks," in *Proc. MILCOM IEEE Mil. Commun. Conf.*, Nov. 2008, pp. 1–7.

[33] R. S. Camati, A. Calsavara, and L. Lima Jr., "Solving the virtual machine placement problem as a multiple multidimensional knapsack problem," in *Proc. ICN*, 2014, p. 264.

[34] J.-Y. Ding, S. Song, R. Zhang, R. Chiong, and C. Wu, "Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 1138–1154, Apr. 2016.

[35] S. Riazi, O. Wigström, K. Bengtsson, and B. Lennartson, "A column generation-based gossip algorithm for home healthcare routing and scheduling problems," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 1, pp. 127–137, Jan. 2019.

[36] A. Muhammad, I. Sorkhoh, L. Qu, and C. Assi, "Delay-sensitive multi-source multicast resource optimization in NFV-enabled networks: A column generation approach," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 286–300, Mar. 2021.

[37] O. Ozturk, "A truncated column generation algorithm for the parallel batch scheduling problem to minimize total flow time," *Eur. J. Oper. Res.*, vol. 286, no. 2, pp. 432–443, Oct. 2020.

[38] E. Choi and D.-W. Tcha, "A column generation approach to the heterogeneous fleet vehicle routing problem," *Comput. Oper. Res.*, vol. 34, no. 7, pp. 2080–2095, Jul. 2007.

[39] C. Wang, C. Guo, and X. Zuo, "Solving multi-depot electric vehicle scheduling problem by column generation and genetic algorithm," *Appl. Soft Comput.*, vol. 112, Nov. 2021, Art. no. 107774.

[40] M. Dunbar, S. Belieres, N. Shukla, M. Amirghasemi, P. Perez, and N. Mishra, "A genetic column generation algorithm for sustainable spare part delivery: Application to the Sydney DropPoint network," *Ann. Oper. Res.*, vol. 290, nos. 1–2, pp. 923–941, Jul. 2020.

[41] E. Figielska, "A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources," *Comput. Ind. Eng.*, vol. 56, no. 1, pp. 142–151, Feb. 2009.

[42] J. J. H. Forrest, J. Kalagnanam, and L. Ladanyi, "A column-generation approach to the multiple knapsack problem with color constraints," *INFORMS J. Comput.*, vol. 18, no. 1, pp. 129–134, Feb. 2006.

[43] B. Patt-Shamir and D. Rawitz, "Vector bin packing with multiple-choice," *Discrete Appl. Math.*, vol. 160, nos. 10–11, pp. 1591–1600, Jul. 2012.

[44] H. Kellerer, U. Pferschy, D. Pisinger, and H. G. Kellerer, *Knapsack Problems*. Berlin, Germany: Springer, 2004. [Online]. Available: https://link.springer.com/book/10.1007/978-3-540-24777-7

[45] X. Dai, J. M. Wang, and B. Bensaou, "Energy-efficient virtual machines scheduling in multi-tenant data centers," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 210–221, Apr. 2016.

[46] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Oper. Res.*, vol. 53, no. 6, pp. 1007–1023, 2005.

[47] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Manage. Sci.*, vol. 50, no. 12, pp. 1861–1871, Dec. 2004.

[48] *COIN-OR Branch-and-Cut Solver*. Accessed: Oct. 15, 2022. [Online]. Available: https://github.com/coin-or/Cbc

[49] S. M. Mirmohseni, C. Tang, and A. Javadpour, "FPSO-GA: A fuzzy metaheuristic load balancing algorithm to reduce energy consumption in cloud networks," *Wireless Pers. Commun.*, vol. 127, no. 4, pp. 2799–2821, Dec. 2022.

[50] Y. Zhou, Q. Zhai, W. Yuan, and J. Wu, "Capacity expansion planning for wind power and energy storage considering hourly robust transmission constrained unit commitment," *Appl. Energy*, vol. 302, Nov. 2021, Art. no. 117570.

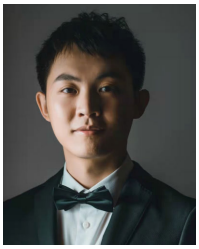
[51] X.-B. Hu and E. Di Paolo, "An efficient genetic algorithm with uniform crossover for air traffic control," *Comput. Oper. Res.*, vol. 36, no. 1, pp. 245–259, Jan. 2009.

Xunhang Sun (Graduate Student Member, IEEE) received the B.Eng. degree in electrical engineering and the B.Mgt. degree in business administration from Xi'an Jiaotong University, Xi'an, China, in 2020, where he is currently pursuing the Ph.D. degree in systems engineering with the School of Automation Science and Engineering. His research interests include planning and scheduling of networked systems, stochastic programs, and large-scale decomposition algorithms, coupled with applications in smart grids, cloud computing, and cyber-physical systems.

Xiaoyu Cao (Member, IEEE) received the Ph.D. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2019. He is currently a Professor with the Systems Engineering Institute, Xi'an Jiaotong University. He is also with the Smart Integrated Energy Department, Sichuan Digital Economy Industry Development Research Institute, Chengdu, China, as the Director. His research interests include power systems planning, scheduling and resilience analysis, cloud computing, and stochastic mixed-integer programming with applications in cyber-physical energy systems. He is the Chair of Technical Committee 9.3 (Control for Smart Cities) at IFAC (2023–2026).

Qiaozhu Zhai (Member, IEEE) received the B.S. and M.S. degrees in applied mathematics and the Ph.D. degree in systems engineering from Xi'an Jiaotong University, Xi'an, China, in 1993, 1996, and 2005, respectively. He is currently a Professor with the Systems Engineering Institute, Xi'an Jiaotong University. His research interests include optimization of large-scale systems and integrated resource bidding and scheduling in the electric power market.

Haisheng Tan (Senior Member, IEEE) received the B.E. degree (Hons.) in software engineering and the B.S. degree (Hons.) in management from the University of Science and Technology of China (USTC), and the Ph.D. degree in computer science from The University of Hong Kong (HKU). He is currently a Professor with USTC. His research interests lie primarily in networking algorithm design and system implementation, where he has published over 100 papers in prestigious journals and conferences, including IEEE/ACM TRANSACTIONS ON NETWORKING, NSDI, INFOCOM, and EruoSys. He recently received the Best Paper Award in WASA'19, CWSN'20, PDCAT'20, and ICPADS'21.



Jianchen Hu was born in Xi'an, China. He received the B.S. degree from Northwest University, Xi'an, in 2011, the M.S. degree from Arizona State University, Tempe, AZ, USA, in 2013, and the Ph.D. degree from Xi'an Jiaotong University, Xi'an, in 2019.

He is currently an Associate Professor with the School of Automation Science and Engineering, Xi'an Jiaotong University. His research interests include predictive control, building energy optimization, and cloud resources allocation.

Lei Zhu received the Ph.D. degree from the School of Computer Science and Technology, Northwestern Polytechnical University, China, in 2017. Currently, he is a Chief Engineer with the Alkaid Laboratory—Huawei Cloud, Huawei Technologies Company Ltd. He is also working on virtual machine placement capacity planning and cloud resource management. His research interests include scheduling and fault tolerance for high-performance computing.

Li Su received the B.S. degree in software engineering and the M.S. degree in computer architecture from Xi'an Jiaotong University, Xi'an, China, in 2008 and 2011, respectively. He is currently a Principal Engineer with the Algorithm Innovation Laboratory, Huawei Cloud, Huawei Technologies Company Ltd. His expertise areas include cloud resources allocation and deep learning.

Wenli Zhou received the Ph.D. degree in applied mathematics from Nankai University in 2010. Once, he was with IBM Research and Microsoft Bing Search. He is currently a Principal Engineer with the Algorithm Innovation Laboratory, Huawei Cloud, Huawei Technologies Company Ltd. His expertise areas include combinatorial optimization and information retrieval. He is working on AI for system related topics like virtual machine placement and cloud resource management.

Feng Gao (Member, IEEE) received the B.S. degree in automatic control and the M.S. and Ph.D. degrees in systems engineering from Xi'an Jiaotong University, Xi'an, China, in 1988, 1991, and 1996, respectively. He is currently a Professor with the Systems Engineering Institute, Xi'an Jiaotong University. From February 2000 to June 2001, he visited Harvard University, Cambridge, MA, USA, as a Post-Doctoral Researcher. His current research interests include optimization, scheduling, and prediction in power systems and cloud computing.

Xiaohong Guan (Life Fellow, IEEE) received the B.S. and M.S. degrees in control engineering from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical and systems engineering from the University of Connecticut, Storrs, CT, USA, in 1993.

From 1993 to 1995, he was a Senior Consulting Engineer with Pacific Gas and Electric. From 1999 to 2000, he visited the Division of Engineering and Applied Science, Harvard University. From 1985 to 1988 and then since 1995, he was with Xi'an Jiaotong University, Xian, China, and has been the Cheung Kong Professor in systems engineering since 1999. From 1999 to 2009, he was the Director of the State Key Laboratory for Manufacturing Systems. From 2008 to 2018, he was the Dean of the School of Electronic and Information Engineering. Since 2019, he has been the Dean of the Faculty of Electronic and Information Engineering. Since 2001, he has also been with the Center for Intelligent and Networked Systems, Tsinghua University, where he was the Head of the Department of Automation from 2003 to 2008. His research interests include economics and security of networked systems, including power and energy systems, manufacturing systems, and cyber-physical systems.

Dr. Guan is a member of the Chinese Academy of Science.