Future Generation Computer Systems, accepted on 27 Dec 2019

Exact algorithms for energy-efficient virtual machine placement in data center

Chen Wei[1], Zhi-Hua Hu[1*], You-Gan Wang[2]

[1] Logistics Research Center, Shanghai Maritime University, Shanghai 200135, China

[2] School of Mathematical Sciences, The University of Queensland of Technology, Brisbane, Queensland 4001, Australia

Abstract

Virtual machine placement (VMP) and power management are important topics in the development of cloud computing and data center. The assignment of virtual machine to physical machine impacts the energy consumption, the makespan and the idle time of physical machines. In this paper, we formulate the problem as a three-dimension bin-packing optimization to minimize the energy cost of working machines and idle machines. By considering the CPU and memory requirements from virtual machine, the assignment is constrained under the capacities of physical machine. Inspired by the best-fit decreasing algorithm, four variants of this exact algorithm are developed to address the multiple-objective problem under multiple-capacity constraints. The effectiveness of the proposed algorithms is demonstrated by experimental results on small-, medium- and large-scale instances profiled from data centers. The results indicate that the algorithms assigning virtual machines to the physical machines of best-fit hosting time is competitive in instances with loose capacity constraints, and the energy-efficiency best-fit algorithm produces efficient assignments when a makespan limit is required on the physical machines. The algorithm combining the fit rules has a linear computing time with respect to the numbers of physical and virtual machines, and a stable performance that obtains gaps of results lower than 5.8% compared to an on-the-shelf mixed-integer linear program solver.

Keywords

Virtual machine placement; data center; computational service supply chain; bin packing problem; first-fit algorithm

1. Introduction

Cloud computing has activated a new supply chain, namely the computational service supply chain, between suppliers of computational resources and customers with computational demands. In the context of cloud computing industry, the computational services are materialized as infrastructure, platforms and software applications. Servicers adopt a computing framework of virtual machines (VMs) and physical machines (PMs) based on virtualization technology to provide customers' demands [1]. A VM is a computing unit of deploying customer's computing requests in PMs for satisfying the constraints of customized computational resources, including the demands of computing time, CPUs and memory sizes. Then these user-defined VMs are assigned to PMs under the constraints of CPUs and memory sizes [2]. Many organizations or companies have established their cloud computing facilities, e.g., big data centers acting as computational service providers to produce attractive benefits, such as reliability, quality of service and robustness [3].

The computational resources are centrally and specially controlled and managed, while the computational services can be accessed friendly by distributed customers with lower costs because the resources can be scheduled and utilized efficiently. To reduce the investment and maintenance costs, some big cloud data centers emerge and even the market of computational service supply chains has been created. In the context of cost reduction, a fine placement of VMs to PMs is important for improving the computational service quality. In the perspective of computational service providers, active placement of VMs to PMs impacts their profits, energy and operating costs.

Energy management in cloud data centers is a crucial issue since it directly constitutes the operational costs and environmental impacts [4, 5]. Serious energy issues including carbon dioxide consumption and system reliability will rise due to electricity power consumption. The emergence of cloud data centers makes significant impacts on the information technology industry over the past years and enables the business models of computational service supply chains, which become industrial standards of using and maintaining computational resources and capabilities. Thus, many big cloud data centers make the energy management even more challenging. In data centers, PMs are managed under groups in the purpose of facilitating power supply, and a PM must stay idle in power-on state when the member PM in its group is working. From the perspective of energy consumption, PMs dominate about 60% of the total energy consumption in a data center [6]. The energy consumption of a PM varies depending on its computational capacity and the computational requirement of the assigned VMs. Idle PM still require approximately 70% of its power consumption in full speed state with tasks [7]. Therefore, an optimal management of the PMs is key to efficiently controlling the power consumption and reducing the number of idle PMs. By optimizing the VMP schedules, the VMs can be assigned to energy-efficient PMs while minimizing the superfluous consumption of idle PMs.

The VMP problem can be formulated to bin-packing models. The energy concerns may introduce a non-linear component to the optimization objectives of the models and make the solution methods challenging. Besides, the quality of service is also a general objective, which is usually represented by completion time of the VMs. Therefore, the VMP problems can be formulated as multi-objective optimization models. In placement models, CPUs and memory of computing resources are general demands of VMs and they also represent resource limits provided by PMs, which are mathematically formulated as constraints. Evenly the simplest one-dimensional bin-packing models present the nature of computational complexity – it is a NP hard problem. To decompose the complexity, various heuristics and intelligent algorithms are developed for the VMP problem and its variants (see Table 1 in Section 2). Three streams of algorithms are involved: heuristics based on exact scheduling rules, intelligent algorithms based on solution encoding and decoding schemes, and model-based algorithms for mathematical programs.

In this study, the VMP considering energy consumption is formulated as a mixed-integer linear program (MILP), whose single-objective model can be solved efficiently by on-the-shelf MILP solvers, e.g., Cplex and Gurobi. The energy cost is a function of assignments of VMs to PMs and the number of active PMs. The model is further investigated mathematically, which contributes to develop the heuristic solution method. Medium- and large-scale instances are generated from profiles originally shared by the Google Data Center [8] and solved efficiently.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of the VMP problem, the bin-packing problem and solution algorithms. In Section 3, the features of the energy-efficient VMP problem are depicted and described. Section 4 formulates the VMP and the model is analyzed. In Section 5, exact algorithms based on best-fit the first scheduling rule and greedy strategy are presented. The computational results on benchmark problems are given in Section 6. We conclude in Section 7.

2. Related studies

2.1. Virtual machine placement

Virtualization technology enables computational service providers to package various computing requests into VMs and implement these VMs in physical machines, such as computer servers or clusters, in the purpose of sharing the relative enormous computational resources among the tasks and reducing the computational costs. Several reviews have been carried out on the optimization of VMPs in the perspective of cost saving, energy saving, task workflow efficiency, service quality and resource transmission [7, 9-11]. In the pioneering review [9], cost optimization approaches for scientific workflow scheduling in cloud and grid computing were reviewed. Eight aspects of cost optimization are discussed, including optimization methods and criteria. In this classification system, our study built multi-objective model, and developed mathematical program and heuristics algorithms. In the reviewed papers, only one paper used mathematical program as a solution method. The review also asserted that meta-heuristic methods are efficient with some compromise on the execution time, while heuristics are popular and effective although their optimality generally can not be guaranteed and proved. In this study, we try to make contributions to VMP and especially the modeling and solution methods.

In the Scopus database, using "virtual machine placement" as the keyword in querying English journal papers, 147 papers are found (April 2, 2019). Most mathematical programs developed in the studies are used to describe the objectives and constraints, and generally not solved. Mazumdar and Pranzo [12] solved the MILP by Cplex, while arbitrary time limits are set for medium- and large-scale instances in their study. Besides, various meta-heuristics and intelligent algorithms are developed. Although these algorithms are flexible to cope with the complex objectives and constraints, and can improve the solutions continuously, they cannot guarantee the solution optimality. First-fit heuristics are most popular algorithms used to solve the problems exactly and efficiently. So, it is usually used in the comparison studies of most papers.

Totally 25 studies on VMP problems are reviewed in Table 1 by summarizing the distinct features, the model types and the employed algorithms. Distinct features are coupled in different studies, mainly including energy and resource utilization degrees. Additionally, the general VMP problems are also studied by considering load balance, machine consolidation and migration of VMs. Among these studies, some just describe the problems and then developed algorithms to solve simulated instances; other studies developed MILPs or non-linear MILPs (NMILPs). Generally, the complicate energy functions result in making the models non-linear ones. Most studies mainly use non-linear expressions to elucidate the complicate constraints, and these NMILPs were not directly solved or the solution methods were not developed based on these models. As for the algorithms developed in these studies, three types are involved. First, various exact heuristics are developed, e.g., round robin (RR), greedy, first-fit decreasing (FFD) and best-fit decreasing (BFD); these algorithms are generally used for comparison studies where intelligent algorithms are developed. However, few studies focused on these algorithms and their improvements, although they are most efficient in computing times. Second, although ant colony optimization (ACO), cuckoo search optimization (CSO) and genetic algorithm (GA) are primarily used, we can conclude that any intelligent algorithms can be tested for the VMP problems. These algorithms can gradually improve the solutions while their computing performances are not competitive and their solutions are not stable. Third, a few studies begin to consider model-based heuristics. However, few studies developed such algorithms. In these studies, Mazumdar and Pranzo [12] developed formal and solvable MILPs, and the results of solving the models are compared with BFD heuristics that are extensively improved from basic ones.

Table 1. Studies on the VMP problems

No.	Study	Features	Model	Algorithm
1	An, Shekhar, Caglar, Gokhale and Sastry [13]	1) Develop a framework for VM replica deployment; 2) Number of PMs is minimized	MILP*	-
2	Tavana, Shahdi-Pashaki, Teymourian, Santos-Arteaga and Komaki [14]	1) Assignments among PMs, VMs, tasks and subtasks; 2) Task migration among VMs 3) Minimize the migration, energy and penalty cost	NMILP*	CSO → GA, FF and RR
3	Gutierrez-Garcia and Ramirez-Nafarrate [15]	1) Balance workloads of PMs by VM migration 2) Minimize the migrations	NMILP*	RR, Greedy, FFD, BFD
4	Gao, Guan, Qi, Hou and Liu [16]	1) Mapping VMs to PMs; 2) Minimize the power consumption and the resource wastage	NMILP*	ACO→ GA
5	Riahi and Krichen [17]	1) Minimize the PMs and total resource wastage 2) Load balance is considered in algorithms	NMILP*	MOGA
6	Baalamurugan and Vijay Bhanu [18]	Minimize the power consumption and the resource wastage	NMILP*	Krill herd MOP→GA, ACO and FFD
7	Alharbi, Tian, Tang, Zhang, Peng and Fei [19]	Minimize the total energy consumed by VMs and the PMs' idle time	Descriptive	ACO→FFD
8	Liu, Zhan and Zhang [20]	1) Consider the energy save and VM migration; 2) Formulate the dynamic VM consolidation and VMP.	NMILP*	ACO
9	Malekloo, Kara and El Barachi [21]	1) VMs placement and consolidation by migration 2) Minimizing resource wastage, communication energy cost and energy consumption	NMILP*	ACO
10	Stolyar [22]	1) real-time assignment of VMs to PMs; 2) minimize the increment of the objective function caused by each new assignment	Descriptive	Greedy
11	Kim, Jeon, Lee and Yang [23]	Parallelize migrations to reduce the time required for VM relocation	Descriptive	Simulation
12	Zhu, Zhuang and Zhang [24]	1) Consider three dimensions (CPU, memory and bandwidth) 2) Minimize energy savings	Descriptive	Allocation strategies heuristics
13	Li, Yan, Yu and Yu [25]	1) VMs' migration and placement criterion are presented.	Descriptive	Hybrid Bayesian network
14	Li, Li, Yuan, Chen and Jiang [26]	Minimize the energy consumption and maximize the resource utilization	MILP*	CRO→CSO, GA, FFD and BFD
15	Satpathy, Addya, Turuk, Majhi and Sahoo [27]	1) A queueing structure is used to manage and schedule VMs. 2) Minimize the resources wastage and power consumption 3) Migration strategies (serial, parallel, improved serial) were tested.	Descriptive	Crow search
16	Sait, Bala and El-Maleh [28]	Power consumption and Resource wastage are minimized	NMILP*	CSO→GA and FFD
17	Ding, Gu, Luo, Chang, Rugwiro, Li and Wen [29]	1) Establish the energy consumption and resource loss model; 2) multi-objective: system security and efficiency	NMILP*	Discrete firefly algorithm
18	Hallawi, Mehnen and He [30]	Minimize PMs and resource wastage	MILP*	GA→FF and FFD
19	Mazumdar and Pranzo [12]	1) mapping incoming and failing VMs; 2) Reducing a total number of VM migrations; 3) Consolidate running server workloads.	MILP	Compared with BFD
20	Vasudevan, Tian, Tang, Kozan and Zhang [31]	1) A profile-based dynamic energy management framework is presented; 2) The study estimates application finishing times and addresses real-time issues in application resource provisioning; 3) Optimize energy efficiency and resource utilization	Descriptive	Repairing GA
21	Zhao, Liu, Wang, Zhang and Zuo [32]	1) Private cloud is considered; 2) heterogeneous and multidimensional VMP; 3) Minimize the datacenter scale and decrease the upfront cost of PMs.	NMILP*	BaB
22	Gupta and Amgoth [33]	Minimize the power consumption by reducing active PMs and minimize the unbalanced utilization of resources among the active PMs.	MILP*	Resource-aware heuristics
23	Canali and Lancellotti [34]	Using behavioral similarities to develop the solution algorithm.	ILP*	Class-Based placement technique
24	Zheng, Li, Li, Shah, Zhang, Tian, Chao and Li [35]	1) Incremental placement and consolidated placement of VMs; 2) Minimize power consumption, resource wastage, server loads, inter-VM and storage network traffic.	NMILP*	biogeography-based optimization

25	Terra-Neves, Lynce and Manquinho [36]	Minimize the energy consumption, resource wastage and migration costs.	MILP*	Constraint-based MOO	

*the model is not solved and is mainly a descriptive model. GA=genetic algorithm, ACO=Ant Colony Optimization, FFD=Fit-First Decreasing, MOO=Multi-Objective Optimization, BaB=branch-and-bound algorithm, CSO=Cuckoo Search Optimization, BFD=Best-Fit Decreasing, RR=round robin, CRO=Chemical reaction optimization, →=[compared with].

2.2. Bin-packing problem and algorithms

The baseline model of bin-packing problem packs the items into bins under the minimization of used bins. So, the decision is to assign the items to bins. In the most typical and simplest cases, the items and bins are all of one dimension. Therefore, the model is constructed as a set-covering model. The bin-packing problem has many applications and variants, including the VMP problem. Like the algorithm classes discussed for the VMP problem in Section 2.1, exact heuristics, intelligent algorithms and model-based solution algorithms are typically used in the studies in Table 2.

Table 2. Studies on the bin-packing problem and solution algorithms

No.	Study	Feature	Model	Algorithm	Application
1	Johnson [37]	Compare the efficiency of fit rules	Descriptive	First fit, best fit, linear-time approximation	Project Schedule
2	Fernandez de la Vega and Lueker [38]	Prove the computation time of solving bin packing	MILP	Reduction	Algorithm design
3	Fekete and Schepers [39]	Provide an asymptotic worst-case performance of 3/4 for a bound	MILP	Elementary bounds, dual feasible functions	generating fast lower bounds
4	Valério De Carvalho [40]	-	MILP, LP	Review	Vehicle routing problem
5	Fleszar and Hindi [41]	Meta-Heuristics	MILP	Variable neighborhood search	Material wastage
6	Alvim, Ribeiro, Glover and Aloise [42]	Hybrid improvement procedure	MILP	Lower bounding strategies, dual min-max solution, load redistribution	Multiprocessor scheduling problem
7	Byholm and Porres [43]	Fragmentable items	MILP	Approximation, metaheuristic algorithms	Shared resources
8	Abdel-Basset, Manogaran, Abdel-Fatah and Mirjalili [44]	Nature inspired heuristics	MILP	Whale optimization algorithm	Bin packing problem case study
9	Ritt and Costa [45]	consider the precedence constraints and the station limits in assembly line	MILP	Linear relaxation	Assembly line
10	Wang, Han, Dósa and Tuza [46]	Game, interest matrix	MILP, game theory	Nash Equilibrium	Item grouping

2.3. Contributions

The VMP is a NP-hard problem due to its complicated capacity constraints and the numerous candidate assignments of VMs to PMs. The energy consumption of idle and active PMs makes the VMP further complicate: the energy consumption of idle PMs is dynamically determined by the resultant makespan of the placement scheduling; the energy consumption of active PMs is non-linear to the ratios of PMs' resources occupied by the VMs. The basic FFD and BFD algorithms lose their effectiveness since the criteria (e.t, energy, execution time, CPUs, memory and other computing resources) of fitting are various in the assignments of VMs to PMs. Although most pioneering studies had used FFD and/or BFD algorithms as the baselines, we can not find the exact implementations of them in the papers in Table 1. The optimality degrees are different when using different criterion of fitting in the assignment process.

To address the research gaps as elucidated above, the following endeavors are made to contribute to the literature. First, the idle time and non-linear energy consumption function of PMs are formulated in the proposed VMP model. The objective is to minimize the energy consumption of the idle and active PMs, and the makespan of the computing tasks of VMs. Second, a MILP is developed to formulate the energy-efficient VMP. The proposed MILP is solvable for schedules with hundreds of PMs and thousands of VMs. Third, we improve the FFD and BFD algorithms by integrating the fitting criteria to solve the VMP. The solution algorithms are mathematically analyzed. Fourth, the proposed algorithms are analyzed by using experimental comparisons. The results indicate that the improved algorithms are efficient and the proposed MILP model has advantage when a makespan upper boundary is predetermined.

3. Energy-efficient virtual machine placement

The assignments of VMs to PMs in VMP dominate the energy consumption in data centers. When a data center receives a computational request, the request is deployed in a VM with specific configuration of computational resources (typically, execution time, CPUs and memory size). Then, the VM is assigned to a PM and executed on the PM. The energy consumption of the executions is determined by the VP and PM simultaneously. Additionally, the PMs are organized in groups such as computer server or cluster and the PMs group is powered off until all the VMs are completed; frequent shutdown and startup of a PM will damage the system and incur serious energy consumption; however, even idle PM that does not host a VM will consume much energy. To provide efficient computing solutions to customers, the computational service providers usually consider the makespan (completion time) of the computing tasks as service quality, and energy consumption as service cost when they develop the schedules.

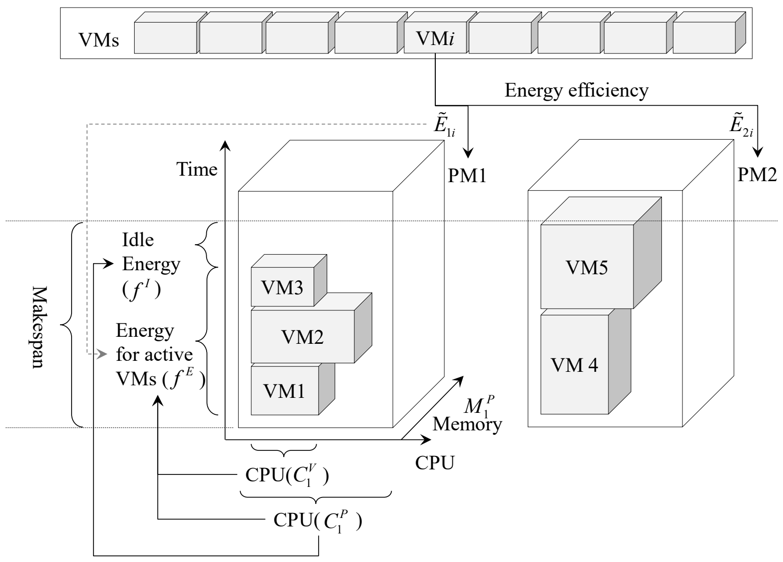

Figure 1. Virtual machine placement diagram

Figure 1 depicts the placement constraints of VMs (V) to PMs (P), and two quality of service (QoS) objectives: the makespan and the energy consumption. In the VMP schedule, a set of VMs are hosted by a set of PMs, constrained by the CPU and memory. In a feasible placement of a VM v to a PM p, the CPU demand of the VM (C_v^V) must be under the CPU capacity provided by the PM (C_p^P), e.t., $C_v^V \le C_p^P$; similarly, the VM's memory demand (M_v^V) should meet the PM's capacity (M_p^P), $M_v^V \le M_p^P$. A PM can host a sequence of VMs. Considering CPU, memory and time, the placements of VMs to PMs can be visualized as a three-dimension bin-packing problem, while a PM can only host a VM at a time interval (when the time is discretized into a set of time intervals), and a VM computes for T_v^V time intervals. Active PM and VM consume energy in two levels. The PM p will consume energy in the range $[E_p^-, E_p^+]$, which depends on the energy efficient ($\tilde{E}_{pv}$) of the hosted VM v.

The features of the VMP problem are summarized as follows:

1) The placements of VMs to PMs are constrained by the PMs' CPUs and memory capacities, $C_v^V \leq C_p^P$ and $M_v^V \leq M_p^P$ for all feasible placement x_{pv}, which is binary decision variable and equals one when v is hosted by p.

2) The placement of a VM to a PM is preemptive, which means that a PM can only host one VM at any time, $\sum_v x_{pv} \leq 1$ at any time t for a PM p.

3) If a PM in the placements starts, it must be on before all VMs finish. The earliest time of shutting down the PM is the makespan of the placements, equally $\sum_v T_v^V x_{pv}$.

4) Two kinds of energy consumption are considered: the working energy of PM for executing VMs, denoted by f^E; the energy when a PM is active and does not host a VM, entitled as idle energy and denoted by f^I. The total energy consumption ($f = f^E + f^I$) is minimized in the model.

5) The working energy f^E depends on the energy efficiency ($\tilde{E}_{pv}$) and the VM's execution time (T_v^V).

6) The energy efficiency ($\tilde{E}_{pv}$) of a VM (v) to a PM (p) is formulated as (1), where $[E_p^-, E_p^+]$ are the minimum and maximum energy for one hour. Equation (1) is a typical function of energy consumption reported in [3].

7) The idle energy of a PM (p) is determined by t, t_p and E_p^-, namely the idle time and the minimum energy consumption per hour.

$$\tilde{E}_{pv} = E_p^- + \left(E_p^+ - E_p^-\right) \cdot e^{-\frac{C_v^V}{C_p^P}} \tag{1}$$

4. Formulation

4.1. Notations

To facilitate the modeling process and increase the notations' readability, the sets, parameters (known data) are denoted by capital letters while the variables and indices are lower-case letters; the superscripts of notations are used to describe the meanings by using capitals while the subscripts are member indices of sets.

The notations used in the paper are summarized as follows. Sets V_p and P_v are used to represent the feasible placements, which help reduce the number of CPU and memory constraints. T_p^P is the pre-occupied time of the PM p, which contributes to build a rolling-horizon placement schedule. Based on x_{pv}, the binary variable y_p represents that the PM is activated or not. Four time-related notations, T, t, t_p and t_p^I denote the time limit of the placement, the makespan, the active time and idle time of PM p.

Set	
P	A set of physical machines, indexed by p.
V	A set of VMs, indexed by v.
Data	
C_p^P	CPU capacity (MIPS) of PM p.
C_v^V	CPU requirement (MIPS) of VM v.
M_p^P	Memory capacity (GB) of PM p.
M_v^V	Memory capacity (GB) required by VM v.
V_p	The set of VMs that can be hosted by PM p, $V_p = \{v \mid C_v^V \leq C_p^P, M_v^V \leq M_p^P, v \in V\}$.
P_v	The set of PMs that can host VM v, $P_v = \{p \mid C_v^V \leq C_p^P, M_v^V \leq M_p^P, p \in P\}$.
T_v^V	Execution time (hour) of VM v.
T_p^P	Pre-occupied time of the PM p.
E_p^+	Maximum energy consumption (kWh) of PM p.
E_p^-	Minimum energy consumption (kWh) of PM p.

E_{pv}	Energy consumed by PM p for placing VM v on it: $E_{pv} = \tilde{E}_{pv} \cdot T_v^V$.
T	Given makespan limit.
α	Makespan scale that is used to adjust the makespan limit.
Variable	
x_{pv}	$x_{pv} \in \{0,1\}$. x_{pv} is 1, if VM v is assigned to PM p; otherwise, 0.
y_p	$y_p \in \{0,1\}$. y_p is 1, if PM p is power on and used; otherwise, 0.
t	Makespan of the VMs' executions.
t_p	Time of PM p that is power on and hosting VMs.
t_p^I	Idle time of PM p, when the PM is not occupied by any VM.
f^E	Total energy consumed by the VMs' placements.
f^I	Total energy consumed by idle times of the PMs.

4.2. Models

A multi-objective optimization model [M1] is developed under the minimization of energy cost and makespan, followed by an energy minimization model [M2] constrained by the makespan upper bound. The energy consumption and its two parts, working energy and idle energy, are computed by (3)-(5). The total makespan (t) is an upper bound of the makespans (t_p) of all PMs, as constrained by (6) and (7). A VM must be placed on a PM and the PMs with VMs are active ones consuming energy, as constrained by (8) and (9). The idle energy consumption of a PM is determined by the idle time, which is computed by (10). The placement variable (x_{pv}) and the PM usage variable (y_p) are binary integers, as denoted by (11).

$$[\text{M1}]\ \min(f, t) \tag{2}$$

Where:

$$f = f^E + f^I \tag{3}$$

$$f^E = \sum_{p;v \in V_p} x_{pv} E_{pv} \tag{4}$$

$$f^I = \sum_p t_p^I E_p^- \tag{5}$$

$$t \geq t_p, \forall p \tag{6}$$

$$t_p = \sum_{v \in V_p} T_v^V x_{pi} + T_p^P, \forall p \tag{7}$$

Subject to:

$$\sum_{p \in P_v} x_{pv} = 1, \forall v \tag{8}$$

$$\sum_{v \in V_p} x_{pv} \leq y_p \cdot M^1, \forall p, M^1 = |V| \tag{9}$$

$$t_p^I \geq (t - t_p) + (y_p - 1) \cdot M^2, \forall p; M^2 = \sum_v T_v^V \tag{10}$$

$$x_{pv}, y_p \in \{0,1\}, t_p \geq 0, t \geq 1, \forall p, v \tag{11}$$

Based on [M1], considering the QoS objective of the VMP, the makespan upper bound is directly formulated as a hard constraint (12). The makespan limit T is calculated by equation (13) and controlled by a makespan scale denoted by α.

$$[\text{M2}]\ \min\{f | t \leq T, (3) - (11)\} \tag{12}$$

$$T = \alpha \cdot \sum_{v \in V} T_v^V / |P| \tag{13}$$

4.3. Analyzing the models

In [M1] and [M2], the possible placements are determined by the combinations of x_{pv}.

Considering (8) and (9), the models are set-covering and one-dimensional bin-packing ones. So, they are NP hard problems and generally difficult to solve. In this section, the models are investigated mathematically, as baseline principles for developing exact algorithms.

Without loss of generality, let $\mathcal{L} = \{v_1, \dots, v_n\}$ be the list of VMs sorted in non-increasing order of execution time, i.e., $T_{v_i}^V \geq T_{v_{i+1}}^V$, $i \in \mathcal{L}$. Let T_i denote the total execution time of a subset of VMs; $K = \{1,2,\cdots\}$ denotes a set of all these subsets; Let $\mathcal{T} = \{T_1, T_2, \dots\}$, which is sorted in non-decreasing order, i.e., $T_i \leq T_{i+1}$. Additionally, in a feasible solution of the VMP, let p_1 and p_2 denote two PMs, $V_{p_1} \supset V_{p_2}$, and the hosting time of them are $\beta_1 \cdot T$ and $\beta_2 \cdot T$, respectively, β_1, $\beta_2 \in (0,1)$.

In the following, besides the nature of the models, the bounds of the makespan, number of active PMs, idle time and energy efficiency are investigated.

Proposition 1. The studied VMP problem is NP-hard.
Proof. The problem is a generalization of the one-dimensional bin packing problem (BPP), where a set of VMs need to be assigned to PMs with limited makespan T and assignment constraints, characterized by memory and CPU capacities. An optimal solution is to pack all the VMs into PMs with the minimum energy consumption. The studied problem is reduced to BPP when T is given in advance and the capacities of different PMs are identical. Therefore, the problem is NP-hard since the BPP is such. □

Proposition 2. In the optimal solution, the makespan T is in $\mathcal{T}$, i.e., $T \in \mathcal{T}$.
Proof. Consider a solution with makespan $T' \notin \mathcal{T}$, without loss of generality, T' satisfies $T_i < T' < T_{i+1}$, $T_i, T_{i+1} \in \mathcal{T}$. According to $t_p \in \mathcal{T}$ and hosting time definition $t_p \leq T'$, we have $t_p \leq T_i$, $\forall p \in P$. By decreasing T' to T_i, the solution is unchanged and the objective is reduced since saving the idle times of PMs. Therefore, T_i is the makespan of the solution. The proposition is proved. □

Proposition 3. For identical PMs, the lower bound of PM number is $\lceil \sum_{v\in V} T_v^V / T \rceil$.
Proof. By relaxing the VMs to fractional items, the minimum number of PMs is $\lceil \sum_{v\in V} T_v^V / T \rceil$. Therefore, the lower bound of PM number is $\lceil \sum_{v\in V} T_v^V \backslash T \rceil$. □

Proposition 4. For heterogeneous PMs, the number of PMs is $|V|$ if $\bigcap_{p\in P'} V_p = \emptyset$, $p' = \{\{i,j\}: i,j \in P, i \neq j\}$.
Proof. $\bigcap_{p\in P} V_p = \emptyset$ indicates that any two VMs cannot be assigned to one identical PM. Therefore, the number of PMs equals to the number of VMs. □

Proposition 5. Given the makespan T and used PMs number $|P|$, any feasible solution has a total idle time $t^I = T \cdot |P| - \sum_{v\in V} T_v^V$.
Proof. The total hosting time of PMs is $T \cdot |P|$, and the summation of VM execution times is $\sum_{v\in V} T_v^V$. Therefore, the total idle time is the unused time of PM, i.e., $T \cdot |P| - \sum_{v\in V} T_v^V$. □

Proposition 6. Given a set of identical PMs and the makespan T, the total idle time of the solution is decided by $|P|$.
Proof. According to $t^I = T \cdot |P| - \sum_{v\in V} T_v^V$, T and $\sum_{v\in V} T_v^V$ is given in advance, therefore, the

idle time depends on $|P|$. □

Proposition 7. Given a makespan T, let T^L denote a lower bound of total idle time for a set of identical PMs. Then,

$$T^L = \max\left\{L_1(\alpha) + L_2(\alpha): \alpha \in \{0, T_v, |T - T_v|\}, v \in V, 0 \le \alpha <= \frac{T}{2}\right\}$$

where

$$L_1(\alpha) = \sum_{v \in N_1} (T - T_v^V), N_1 = \{v \in V: T_v^V > T - \alpha\}$$

$$L_2(\alpha) = T \cdot \left\lceil \frac{\sum_{v \in N_2} T_v^V}{T} \right\rceil - \sum_{v \in N_2} T_v^V, N_2 = \{v \in V: T - \alpha \ge T_v^V \ge \alpha\}.$$

Proof. Each VM in N_1 requires one single PM so that each of them has an idle time $T - T_v^V$. As for the VMs in N_2, no one of them can be assigned to the PMs used by VMs in N_1 according to the makespan limit. The minimal number of PMs required for N_2 is $\lceil \sum_{v \in N_2} T_v^V / T \rceil$, when the VMs in N_2 can be exactly grouped into these PMs. So the lower bound of total idle time in N_2 is $T \cdot \lceil \sum_{v \in N_2} T_v^V / T \rceil - \sum_{v \in N_2} T_v^V$. By relaxing V with $N_1 \cup N_2$, the total idle time is obtained by $L_1(\alpha) + L_2(\alpha)$. Therefore, the proposition is proved. □

Proposition 8. In an optimal solution of identical PMs, there exists at most one PM of less than half makespan.
Proof. When there exists two PM of half makespan, these two PMs can be merged into one to reduce the used PM number and the energy consumption. □

Proposition 9. In an optimal solution with $\beta_1 + \beta_2 \le 1$, we have $\sum_{v \in V'_{p_2}} (E_{p_1 v} - E_{p_2 v}) \cdot T_v^V + T \cdot \beta_2 \cdot (E_{p_2}^- - E_{p_1}^-) - T \cdot E_{p_2}^- > 0$.
Proof. Let V'_{p_1} and V'_{p_2} denote the set of VMs that are assigned to p_1 and p_2. The total cost is

$$f = \sum_{p \in \{p_1, p_2\}} \sum_{v \in V'_p} E_{pv} \cdot T_v^V + T \cdot (1 - \beta_1) \cdot E_{p_1}^- + T \cdot (1 - \beta_2) \cdot E_{p_2}^-.$$

After merging the VMs of p_2 into p_1, the total cost is

$$f' = \sum_{v \in V'_{p_1} \cup V'_{p_2}} E_{p_1 v} \cdot T_v^V + T \cdot (1 - \beta_1 - \beta_2) \cdot E_{p_1}^-.$$

If $f' \ge f$, the total cost of using both PMs is lower than the cost of p_1. □

Proposition 10. For a set of PMs with identical E_p^-, the hosting time of the PM with lowest energy efficiency is T.
Proof. Let p_1 denote the lowest energy efficiency PM, i.e., $E_{p_1,v} \le E_{p,v}$, $p \in P$, $v \in V$. p_m is the PM with T. The sets of VMs assigned to p_1 and p_m are denoted by V_{p_1} and V_{p_m}, respectively. Let $t_{p_1} = T'$ and $t_{p_m} = T$, $T' < T$. The working energy consumption is $f^E = \sum_{v \in V_{p_1}} E_{p_1 v} \cdot T_v + \sum_{v \in V_{p_m}} E_{p_m v} \cdot T_v$. By swapping the VMs in V_{p_1} and V_{p_m}, the working energy consumption is $f^{E'} = \sum_{v \in V_{p_1}} E_{p_m v} \cdot T_v + \sum_{v \in V_{p_m}} E_{p_1 v} \cdot T_v$. The changed energy consumption Δf^E

is calculated by

$$\begin{aligned}
\Delta f^E &= f^{E'} - f^E \\
&= \sum_{v \in V_{p_1}} \left(E_{p_m v} - E_{p_1 v}\right) \cdot T_v + \sum_{v \in V_{p_m}} \left(E_{p_1 v} - E_{p_m v}\right) \cdot T_v \\
&= \left(E_{p_m v} - E_{p_1 v}\right) \cdot \left(\sum_{v \in V_{p_1}} T_v - \sum_{v \in V_{p_m}} T_v \right) \\
&= \left(E_{p_m v} - E_{p_1 v}\right) \cdot (T' - T)
\end{aligned}$$

where $E_{p_m v} - E_{p_1 v} > 0$ and $T' - T < 0$. Therefore, $\Delta f^E < 0$ which implies a reduction of energy consumption by adding the hosting time of PM with lowest energy-efficiency. □

Proposition 11. For the set of PMs with identical energy efficiency, the idle time of lowest E_p^- PM is maximized.

Proof. Similar with the previous proof of **Proposition 10**. □

5. Exact solution algorithms

As reviewed in Table 1, the nature-inspired intelligent algorithms are the main stream of algorithms applied to VMP, while most of these algorithms are compared with exact heuristics algorithms, mainly including FFD and BFD. Generally, the FFD and BFD are fast while they are not competitive comparing with the intelligent algorithms in terms of optimality. Even so, few studies elaborate the procedures of FFD and BFD although the makespan and energy cost are generally involved as criteria of choosing VMs for placements. Generally, two kinds of exact algorithms can be developed, namely model-based heuristics and meta-heuristics. In this study, the branch-and-bound (BaB) is taken as the first kind of exact algorithms, which are implemented in on-the-shelf MILP solvers, e.g., Cplex and Gurobi. As for the second kind, three algorithms (the Algorithms 1-3) are developed here using the notations in Section 4.

The Algorithm 1 is a detailed design of BFD heuristic which is popular in the research community of bin-packing problems. We improve the basic BFD by designing sorting rule and fit criteria based on hosting time and energy efficiency, and integrating the fit criteria in the procedure of the algorithms. The algorithm is developed with the following considerations. First, the algorithm selects the best-fit criterion that determines the inserting order of VMs and the priority order of PMs. Second, the sorted VMs are selected sequentially and assigned to the first active PM that satisfies the CPU and memory requirements. Finally, the algorithm selects a serviced PMs more preferentially than an empty PMs for hosting VMs to reduce the number of PMs in usage. If no serviced PM satisfies the CPU and memory requirements of the selected VM, the algorithm assign the VM to the empty PM of the lowest energy efficiency. As a summary, the Algorithm 1 constructs a solution of VMP by inserting VMs sequentially into the schedules of PMs.

Algorithm 1	Best-Fit Decreasing heuristic algorithm (BFD)
Input	$V,\ P,\ T_v^V,\ E_p^+,\ E_p^-,\ E_{pv},\ C_p^P,\ C_v^V,\ M_p^P,\ M_v^V,\ T$
Output	$x_{pv},\ y_p,\ t_p,\ t$
Step 1	Initialize $x_{pv} = 0,\ y_p = 0,\ t_p = 0,\ t = 0,\ p \in P,\ v \in V$
Step 2	Execute [VA] or [VD]: [VA] Sort V into V' by T_v^V in ascending order; [VD] Sort V into V' by T_v^V in decreasing order.

```
Step 3          Initialize a set $\mathcal{P}$ holding the serviced PM, $\mathcal{P} = \emptyset$
Step 4          For $v$ in $V'$
Step 4.1          Set $found = False$
Step 4.2          Execute [PE] or [PT]:
                  [PE] Sort $\mathcal{P}$ into $\mathcal{P}'$ by $\tilde{E}_{pv}$ in ascending order, $p \in \mathcal{P}$.
                  [PT] Sort $\mathcal{P}$ into $\mathcal{P}'$ by $t_p$ in ascending order, $p \in \mathcal{P}$.
Step 4.3          For $p$ in $\mathcal{P}'$
Step 4.3.1          If $t_p + T_v^V \le T$ and $C_v^V \le C_p^P$ and $M_v^V \le M_p^P$
Step 4.3.1.1          Assign $v$ to $p$: $x_{pv} \leftarrow 1,\ y_p \leftarrow 1,\ t_p \leftarrow (t_p + T_v^V)$
Step 4.3.1.2          Set $found = True$
Step 4.3.1.3          Break
                    End if
                  End for
Step 4.4          If $found$ is True
Step 4.4.1          Go to Step 4
                  end If
Step 4.5          Set $\bar{\mathcal{P}} = P \backslash \mathcal{P}$
Step 4.6          [PE] Sort $\bar{\mathcal{P}}$ into $\mathcal{P}'$ by $\tilde{E}_{pv}$ in ascending order, $p \in \bar{\mathcal{P}}$.
Step 4.7          For $p$ in $\bar{\mathcal{P}}'$
Step 4.7.1          If $t_p + T_v^V \le T$ and $C_v^V \le C_p^P$ and $M_v^V \le M_p^P$
Step 4.7.1.1          Assign $v$ to $p$: $x_{pv} \leftarrow 1,\ y_p \leftarrow 1,\ t_p \leftarrow (t_p + T_v^V)$
Step 4.7.1.2          Go to Step 4.
                    End if
                  End for
                End for
Step 5          $t = \max_{p \in P} t_p$
Step 6          Output $x_{pv},\ y_p,\ t_p,\ t$
```

The Algorithm 1 has four variants considering choices in the Steps 2 ([VA] or [VD]) and 4.2 ([PE] or [PT]).

1) VAPE. The algorithm sorts the unassigned VMs in a descending order by execution time and assigns VMs to active PMs with the lowest energy efficiency.

2) VAPT. The algorithm sorts the unassigned VMs in a descending order by execution time, and inspired by Proposition 10, assigns VMs to active PMs with the shortest hosting time.

3) VDPE. The algorithm sorts the unassigned VMs in an ascending order by execution time and assigns VMs to active PMs with the lowest energy efficiency.

4) VDPT. The algorithm sorts the unassigned VMs in a descending order by execution time, and inspired by Proposition 10, assigns VMs to active PMs with the shortest hosting time.

In the following, use the Algorithm 1 [*] to represent the Algorithm 1 using the choice [*]; for simplicity, [*] can also be used to represent the Algorithm 1 [*] when it will not produce confusions.

Proposition 12. The working energy consumption of the solution returned by the Algorithm 1 [PE] is a lower bound of the working energy consumption of [M2] when the makespan is unlimited ($T = +\infty$) and the resultant PMs in the solutions of the Algorithm 1 and [M2] are same.

Proof. The set of PMs in the optimal solutions is denoted as P^*. In the solution, each VM of the set V must be assigned to a PM in P^*. The Algorithm 1 [PE] selects a PM with minimal energy consumption for each VM, resulting in a total working energy consumption $f^{E,PE}$,

$$f^{E,PE} = \sum_{i \in V} \min_{p \in P^*} \{E_{pi}\}.$$

On the other hand, for a VM i, its minimal working energy consumption is denoted by $\min_{p\in P^*}\{E_{pi}\}$, and thus the total working energy consumption (f^E) in the optimal solution is not lower than the total of minimal working energy consumption. So, we have the inequality:

$$f^E \geq \sum_{i\in V} \min_{p\in P^*}\{E_{pi}\} = f^{E,PE}.$$

Therefore, the working energy consumption from the Algorithm 1 [PE] is a lower bound of the working energy consumption of [M2]. □

Proposition 13. The solutions of the Algorithm 1 [PE] and Algorithm 1 [PT] cannot dominate each other when the objective is to minimize $f = f^E + f^I$ or f^I.
Proof. Without loss of generality, we assume that two PMs, p_1 and p_2 satisfying $t_p = t_{p_1} = t_{p_2}$ and they must host two VMs with execution time $t = T_v$, which is denoted by v. Therefore, the sequence of selecting VMs makes no difference on the solution. Given that the PM p_1 is more efficient than the PM p_2, the working energy consumption of VM on p_1 and p_2 are denoted as $\tilde{E}_{p_1,v}$ and $\tilde{E}_{p_2,v}$. The minimum idle energy of p_1 and p_2 are denoted as $E^-_{p_1}$ and $E^-_{p_2}$.
By using [PE], the two VMs are assigned to the efficient PM, namely p_1. The working energy and idle energy consumptions, and the total energy consumption are calculated by

$$f^{E,PE} = 2 \cdot \tilde{E}_{p_1,v} \cdot t, \text{ and } f^{I,PE} = 2 \cdot E^-_{p_2} \cdot t,$$
$$f^{PE} = 2 \cdot \tilde{E}_{p_1,v} \cdot t + 2 \cdot E^-_{p_2} \cdot t.$$

By using [PT], the two VMs are equal assigned to PM p_1 and p_2, so the assignment produces no idle time. The total energy consumption is then calculated by

$$f^{PT} = \tilde{E}_{p_1,v} \cdot t + \tilde{E}_{p_2,v} \cdot t.$$

By comparing the total energy consumptions of using [PE] and [PT], we found that

$$\tilde{E}_{p_1,v} + 2 \cdot E^-_{p_2} \begin{pmatrix} > \\ = \\ < \end{pmatrix} \tilde{E}_{p_2,v} \Longrightarrow f^{PE} \begin{pmatrix} > \\ = \\ < \end{pmatrix} f^{PT}.$$

The result indicates that the dominance of [PE] on [PT] is determined by the efficiency of the used PMs and the idle energy efficiency of the lower efficient PM. The proposition is proved. □

Proposition 14. When [PT] is chosen, [VA] and [VD] cannot dominate each other when the objective is to minimize $f = f^E + f^I$ or f^I.
Proof. Assume that two PMs (p_1 and p_2) have the same energy parameters, E^+_p and E^-_p, and three VMs (v_1, v_2 and v_3) are to be assigned. The execution times of these VMs are denoted as t_1, t_2, t_3, $t_1 < t_2 < t_3$ and $t_1 + t_2 > t_3$. By using [VA], v_1 is first assigned to p_1; v_2 is assigned to p_2 and finally v_3 is assigned to p_1. Thus, the executing times of p_1 and p_2 for the VMs are

$$t_{p_1} = t_1 + t_3, \text{ and } t_{p_2} = t_2.$$

So, the total energy consumption of using [VA] is

$$f^{VA} = (t_1 + t_2 + t_3) \cdot E_{pv} + |t_1 + t_3 - t_2| \cdot E^-_{p_1}.$$

By using [VD], firstly v_3 is assigned to p_1; secondly, v_2 is assigned to p_2; and finally, v_1 is assigned to p_2. Similar with the calculation of energy consumption in using [VA], we obtain

$$f^{VD} = (t_1 + t_2 + t_3) \cdot E_{pv} + |t_3 - t_1 - t_2| \cdot E^-_{p_2}$$

By comparing f^{VA} and f^{VD}, we found that

$$\frac{E_{p_1}^-}{E_{p_2}^-}\begin{pmatrix} > \\ = \\ < \end{pmatrix}\frac{\text{t1+t3−t2}}{\text{t3−t1−t2}} \Rightarrow f^{VA}\begin{pmatrix} > \\ = \\ < \end{pmatrix}f^{VD}.$$

The result indicates that the idle energy consumption is affected by the differences of execution times among VMs. It proves that the strategy of scheduling the VMs with long execution times in advance is not always effective when the PMs' active times are to be balanced. The proposition is proved. □

Proposition 15. When [PE] is chosen, (1) the impacts of [VA] and [VD] on the total energy consumption are same when the makespan is unlimited ($T = +\infty$), and (2) [VA] and [VD] cannot dominate each other when the makespan is limited ($T < +\infty$).

Proof. (1) The proof by contradiction is used here. When [VA] and [VD] are used to schedule the VM i, the PMs p_1 and p_2 are chosen individually. Assume that p_1 is different from p_2, and $E_{p_1,i} \neq E_{p_2,i}$. However, considering that [PE] is chosen in advance, we obtain $E_{p_1,i} = \min_{p\in P}\{E_{pi}\}$ and $E_{p_2,i} = \min_{p\in P}\{E_{pi}\}$, which is conflict with $E_{p_1,i} \neq E_{p_2,i}$. Therefore p_1 and p_2 must be same. Proved. □

(2) Introduce an example first. The execution times of three VMs (v_1, v_2, v_3) are denoted by t_1, t_2, t_3, where $t_1 < t_2 < t_3$, $t_1 + t_2 > t_3$; Given two PMs (p_1, p_2), p_1 is more efficient than p_2; a makespan T is given and $t_1 + t_2 < T$, $T < t_1 + t_3$.

By using [VA], v_1 and v_2 are assigned to p_1 because $t_1 + t_2 < T$, $E_{p_1,v_1} < E_{p_2,v_1}$ and $E_{p_1,v_2} < E_{p_2,v_2}$, then v_3 has to be assigned to p_2 because $T < t_1 + t_3$. The total energy consumption f^{VA} is

$$f^{VA} = E_{p_1,v_1} + E_{p_1,v_2} + E_{p_2,v_3} + |t_1 + t_2 - t_3| \cdot E_{p_2}^-.$$

By using rule VD, v_3 is assigned to p_1, then v_2 and v_1 are assigned to p_2 because $T < t_1 + t_3$. The total energy consumption f^{VD} is

$$f^{VD} = E_{p_1,v_3} + E_{p_2,v_2} + E_{p_2,v_1} + |t_1 + t_2 - t_3| \cdot E_{p_1}^-.$$

To compare f^{VA} and f^{VD}, we relax the difference of $E_{p_1,i}$ and $E_{p_2,i}$, $i \in \{v_1, v_2, v_3\}$. We obtain

$$E_{p_1}^-\begin{pmatrix} < \\ = \\ > \end{pmatrix}E_{p_2}^- \Rightarrow f^{VA}\begin{pmatrix} > \\ = \\ < \end{pmatrix}f^{VD}.$$

The result indicates that [VA] and [VD] can not dominate each other. It proves that the strategy of scheduling the VMs with long execution times in advance is not always effective when the energy-efficient PMs are to be scheduled with higher priorities. The proposition is proved. □

As a summary of Proposition 14 and Proposition 15, the most popular FFD and BFD criteria are not always effective when the working and idle energy consumption, and the makespan are simultaneously optimized.

The Algorithms 2-3 are greedy algorithms while they use different assessment criteria of PM schedules when the VMs are chosen for insertion to these schedules. In the Algorithm 2, the total energy cost including f^E and f^I is tried to be minimized, while the Algorithm 3 mainly minimizes the working energy consumption f^E.

In the Algorithm 2, the following steps are refined. First, the assessment criteria of matching VM to PM is the increment of summation of f^E and f^I, as evaluated through step 2.1.1-2.1.5.

Second, for an unassigned VM, the candidate PMs are sorted by the increment of total energy consumption in an ascending order, as described in step 2.2. Third, the PM that has the lowest energy increment and satisfies the requirement is selected to service the chosen VM, as described in step 2.3.

In the Algorithm 3, the following additional features are implemented. First, the assessment criterion of the PM is the energy efficiency $\tilde{E}_{pv}$. Second, the algorithm sorts the candidate PMs by $\tilde{E}_{pv}$ in an ascending order, as described in step 2.1. Third, the first PM satisfying the requirement in the sorted PMs is assigned to host the selected VM, as described in step 2.2.

```
Algorithm 2    Greedy placement, $G(f)$
Input          $V,\ P,\ T_v^V,\ E_p^+,\ E_p^-,\ E_{pv},\ C_p^P,\ C_v^V,\ M_p^P,\ M_v^V,\ T$
Output         $x_{pv},\ y_p,\ t_p,\ t$
Step 1         Initialize $x_{pv} = 0,\ y_p = 0,\ t_p = 0,\ t = 0,\ f^I = 0,\ f^E = 0,\ p \in P,\ v \in V$
Step 2         For $v$ in $V$
Step 2.1         For $m$ in $P$
                   Calculate the increment of $f^I$ and $f^E$ by assigning $v$ to $m$:
Step 2.1.1         $t'_p \leftarrow t_p, p \in P$
Step 2.1.2         $t'_m \leftarrow t'_m + T_v^V$
Step 2.1.3         $t' = \max_{p \in P} t'_p$
Step 2.1.4         $f_m^I = \sum_{p \in P} E_p^- \cdot (t' - t'_p) - f^I$
Step 2.1.5         $f_m^E = \tilde{E}_{mv}$
                 End for
Step 2.2         Sort $P$ into $P'$ by $(f_p^I + f_p^E)$ in ascending order, $p \in P$
Step 2.3         For $p$ in $P'$
Step 2.3.1         If $t_p + T_v^V \le T$ and $C_v^V \le C_p^P$ and $M_v^V \le M_p^P$
Step 2.3.1.1         Assign $v$ to $p$: $x_{pv} \leftarrow 1,\ y_p \leftarrow 1,\ t_p \leftarrow (t_p + T_v^V)$
Step 2.3.1.2         Break
                   End if
                 End for
Step 2.4         $t = \max_{p \in P} t_p$
Step 2.5         $f^I = \sum_{p \in P} E_p^- \cdot (t - t_p)$
               End for
Step 3         Output $x_{pv},\ y_p,\ t_p,\ t$
```

```
Algorithm 3    Choose most energy-efficient PM from available PMs, $G(f^E)$
Input          $V,\ P,\ T_v^V,\ E_p^+,\ E_p^-,\ E_{pv},\ C_p^P,\ C_v^V,\ M_p^P,\ M_v^V,\ T$
Output         $x_{pv},\ y_p,\ t_p,\ t$
Step 1         Initialize $x_{pv} = 0,\ y_p = 0,\ t_p = 0,\ t = 0,\ p \in P,\ v \in V$
Step 2         For $v$ in $V$
Step 2.1         Sort $P$ into $P'$ by $\tilde{E}_{pv}$ in ascending order, $p \in P$.
Step 2.2         For $p$ in $P'$
Step 2.2.1         If $t_p + T_v^V \le T$ and $C_v^V \le C_p^P$ and $M_v^V \le M_p^P$
Step 2.2.1.1         Assign $v$ to $p$: $x_{pv} \leftarrow 1,\ y_p \leftarrow 1,\ t_p \leftarrow (t_p + T_v^V)$
Step 2.2.1.2         Break
                   End if
                 End for
               End for
```

Step 3	$t = \max_{p \in P} t_p$
Step 4	Output x_{pv}, y_p, t_p, t_{max}

Computational complexity of the exact algorithms is analyzed as follows:

Algorithm 1. We use the bubble sort to rank the VMs in Step 2 and the PMs in Step 4.2 and 4.6. Therefore, in the worst case, the computational complexity of the sorts is $O(|V|^2 + |V| \cdot |P|^2)$. The assignment is completed by comparing the cost of assigning one VM to every PM, thus in the worst case, the computational complexity of the assignment is $O(|V| \cdot |P|)$. So, the overall computational complexity of the Algorithm 1 is $O(|V|^2 + |V| \cdot |P|^2 + |V| \cdot |P|)$.

Algorithm 2. The computational complexity of the incremental energy consumption of PMs hosting the VMs is $O(|V| \cdot |P|)$ in Step 2.1 and the one of sort is $O(|V| \cdot |P|^2)$ in Step 2.2. The assignment of VMs to PMs in Step 2.3 is $O(|V| \cdot |P|)$ in the worst case. Therefore, the overall computational complexity is $O(2 \cdot |V| \cdot |P| + |V| \cdot |P|^2)$.

Algorithm 3. The computational complexity of sort in Step 2.1 is $O(|V| \cdot |P|^2)$. The assignment in Step 2.2 is $O(|V| \cdot |P|)$. Therefore, the overall computational complexity is $O(|V| \cdot |P| + |V| \cdot |P|^2)$.

Considering the above analysis of computational complexity, the three algorithms are polynomial algorithms that can be effectively executed when the problem scales are increasing.

6. Experiments

Experimental instances are solved by the MILP solver (Gurobi 6.0, www.gurobi.com) and the proposed algorithms. The MILP model and algorithms are programmed by using Python and implemented on a desktop computer with dual core 2.4 GHz CPU, 8.0 GB RAM and an operation system of windows 10.

6.1. Data sets and assessments

As studied in Section 4.1, the parameters of the proposed models [M1] and [M2] can be incorporated into a data vector, $\left[C_p^P, C_v^V, M_p^P, M_v^V, V_p, P_v, T_v^V, T_p^P, E_p^+, E_p^-, E_{pv}, T, \alpha\right]$. Based on the data set provided by the Google Data Center [8], the statistical distributions of the first ten components can be configured as Table 3 and 4. Using these settings, the dataset "$PxVyz$" is created so that it contains x PMs and y VMs in scenario z, where $z = l$ represents a loose CPU and memory requirements of VMs, and $z = t$ represents a tight CPU and memory requirements of VMs. Small-, medium- and large-scale instances are generated with x differing from 10 to 500 and y differing from 50 to 5000. The CPU capacity, memory capacity and energy consumption of the PMs are generated according to the data distributions in Table 3, where U represents "uniform distribution". These distributions are summarized with extensions from [19]. Furthermore, in the loose scenario, the requirements of the VMs are generated so that every VM can be deployed in all the PMs, while in the tight scenario, every VM can be deployed in only several certain PMs (less than ten). The distribution of VM requirements is displayed in Table 4.

Table 3. Data distribution revised from data center profiles

$C_p^P \leftarrow U[10,20]$	$C_v^V \leftarrow U[1,8]$
$M_p^P \leftarrow U[20,40]$	$M_v^V \leftarrow U[10,20]$
$T_v^V \leftarrow U[1,12] \cdot 100/3600$	$T_p^P \leftarrow U[0,6] \cdot 100/3600$
$E_p^+ \leftarrow \left(C_p^P \cdot 20 + M_p^P \cdot 3\right) \cdot U[0.8,1.2]$	$E_p^- \leftarrow \left(C_p^P + M_p^P\right) \cdot U[0.9,1.1]$

Table 4. The CPU and memory requirements of VMs in loose and tight scenarios

Loose scenario	$C_v^V \leftarrow U[1,8]$	$M_v^V \leftarrow U[10,20]$
Tight scenario	$C_v^V \leftarrow U[1,15]$	$M_v^V \leftarrow U[10,30]$

Table 5. Experiment settings

No.	Purpose	Datasets	α
1	Demonstrate the Algorithms 1-3 and [M2] in loose scenario.	$P50V100l$	$\alpha = 5$
2	Analyze CPU, memory and PM utilization of the Algorithms 1-3 and [M2] in tight scenario.	$P10V(50/100/200)t$, $P100V(500/800/1000)t$, $P500V(1000/2000/5000)t$	$\alpha = \infty$
3	Compare the Algorithms 1-3 and [M2] in terms of optimality and computing time.	$P10V(50/100/200)t$, $P100V(500/800/1000)t$, $P500V(1000/2000/5000)t$, $P10V(50/100/200)l$, $P100V(500/800/1000)l$, $P500V(1000/2000/5000)l$	$\alpha = \infty$
4	Study the impacts of α on the performances of the Algorithms 1-3 and [M2].	$P50V100l$, $P50V100t$	$\alpha = \begin{Bmatrix} 1.5,2,2.5, \\ 3,3.5, \\ 4,4.5,5 \end{Bmatrix}$

6.2. Experimental settings

In the experiments, the VMP is studied by using the Algorithms 1-3 and [M2]. The experiments are designed to evaluate the quality of solutions and the performances of the algorithms and MILP. The computing time of MILP solver is limited under 3600 seconds. Table 5 lists the experiment settings of experimental purpose, instances, and the corresponding configurations.

6.3. Results and discussion

6.3.1. Loose scenario results

The solution methods (the Algorithm 1-3 and [M2]) are tested on instance $P50V100l$ under a makespan limit of 1.11 hour ($\alpha = 3$). The results are displayed in Table 6. Comparing the minimal (MIN^1), the second-minimal (MIN^2) and the maximal hosting time (MAX) of the resultant PMs, we analyze that every solution contains at most one PM that has a hosting time (MIN^1) of more than half of the makespan, which is consistent with Proposition 8. Among the exact algorithms, VDPE obtains the lowest total energy consumption, using 17 PMs to host all the VMs. Compared with VDPE, VDPT generates a lower idle energy consumption. However, its higher working energy consumption results in a higher total energy consumption. VAPT and VAPE obtain a higher total energy consumption than VDPT and VDPE, and they generate high consumption of both working and idle energy. $G(f^E)$ produces the minimum working energy consumption among all the exact algorithms, while the time difference between MIN^1 and MIN^2 is the largest and it uses 20 PMs, which is three more than the optimal number of used PMs. $G(f)$ has the highest total energy consumption and uses all the PMs to host the VMs, leading to the lowest makespan (0.39 hour) in the algorithms. Solving [M2] obtains the lowest total energy consumption with both lowest working and idle energy consumptions. In the perspective of computing time, VAPT, VAPE, VDPT, VDPE and $G(f^E)$ spend less than one second to generate the solutions. $G(f)$ requires less than 3 seconds to solve the result. However, the MILP solver requires much computing time for solving [M2], it runs to the computing time limit of one hour, which is notably higher than the exact algorithms.

Table 6. Result of [M2] and algorithms on instance $P50V100l$ with $\alpha = 5$

Methods	f	f^E	f^I	PMs	MIN^1 (h)	MIN^2 (h)	MAX (h)	CT (s)
[M2]	2203.39	2198.42	4.97	17	1.02	1.08	1.10	3600.35
VDPE	2254.52	2242.35	12.17	17	0.96	1.09	1.11	0.07
VDPT	2255.39	2243.23	12.16	17	1.07	1.07	1.11	0.08
VAPE	2358.54	2279.57	78.97	19	0.84	0.84	1.08	0.06
VAPT	2358.54	2279.57	78.97	19	0.84	0.84	1.08	0.06
$G(f^E)$	2377.17	2238.53	138.64	20	0.33	0.66	1.08	0.19
$G(f)$	2572.15	2527.07	45.08	50	0.22	0.25	0.39	3.00

Note: $\mathrm{MIN}^1 = \min t_p y_p$; $\mathrm{MIN}^2 = \min\{t_p y_p : t_p y_p > MIN^1\}$; $\mathrm{MAX} = \max t_p y_p$; CT=Computing time.

6.3.2. Tight scenario results

The solutions of instance $P100V500t$ solved by the algorithms and [M2] are examined for analyzing the algorithm performances. The energy consumptions of the results are displayed in Figure 2. $G(f)$ and the Algorithm 1 using [PT] generate low idle energy consumptions as well as the total consumption. And the [VAPT] obtains the minimal total consumption among the exact algorithms. Compared to [VAPT], the Algorithm 1 [PE] and $G(f^E)$ obtain lower working energy consumptions, while they incur significantly high idle energy consumptions.

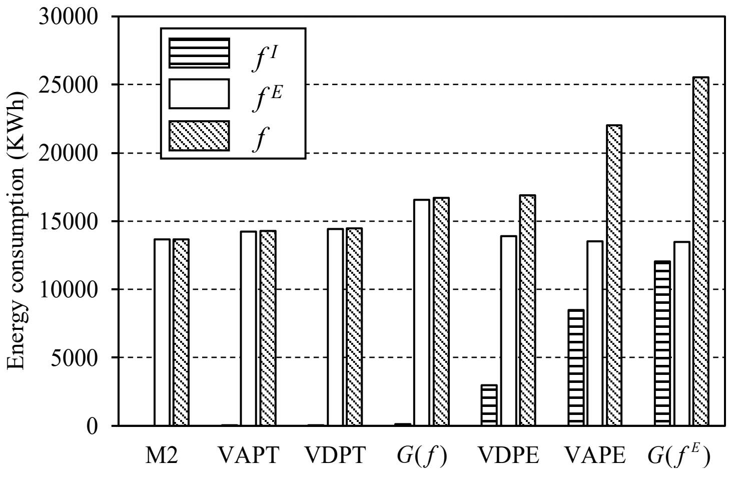

Figure 2. Comparisons of energy consumption obtained by the algorithms on $P100V500t$

The CPU, memory and PM time utilizations of the results from the algorithms are displayed in Figure 3. The CPU utilization is calculated by $\sum_{v\in V}(C_v^V \cdot T_v^V) \,/\, \sum_{p\in P}(C_p^P \cdot t_p)$, and the utilizations of memory and PM are calculated similarly. [VAPT], which outperforms other exact algorithms, gives solution with a higher memory utilization than the solution from MILP, while with less PM time utilization. For [VDPE] and [VAPE] that use [PE] as inserting criterion, their CPU and memory utilizations is higher than the ones of [VAPT], however, their PM time utilization is significantly lower. By considering both the idle and energy consumption, $G(f)$ obtains a higher PM time utilization than $G(f^E)$.

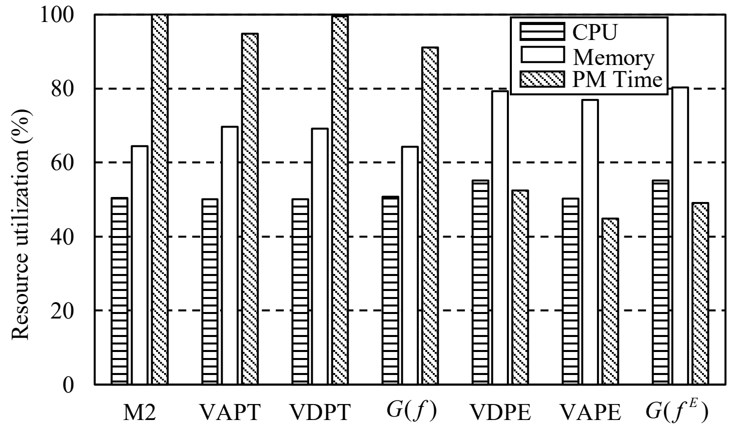

Figure 3. Utilizations in solutions generated from different algorithms on $P100V500t$

6.3.3. Algorithm performances

The performances of solution methods are examined in both tight and loose scenarios. The small-, medium- and large-scale instances are tested when T is set to infinite ($T \leftarrow +\infty$).

Table 7 shows the results in the loose scenario and the gaps of the total energy consumption resulted from the Algorithms 1-3 and [M2]. Solving [M2] can obtain the optima for the instances in such loose scenario. [VDPT] and [VDPE] obtain the same solutions, while [VAPT] and [VAPE] also generate the equal objective values. These same solutions indicate that the inserting criterion of PMs affects little on the total energy consumption. Among the instances, [VDPT] and [VDPE] solve 6 instances with gaps lower than 1.00% compared with the results from [M2], and [VAPT] and [VAPE] solve 5 instances. $G(f^E)$ generates the optimal solutions of the small-scale instances. However, nearly 20% gaps are observed in the medium- and large-scale instances. The gaps of solution from $G(f)$ are higher than 15% for all the instances.

Table 7. Comparison of algorithm and MILP results in loose scenario when T is set to infinite

Instance	VDPT		VDPE		VAPT		VAPE		$G(f^E)$		$G(f)$		[M2]
	f	Gap	f	Gap	f	Gap	f	Gap	f	Gap	f	Gap	CT/s
P10V50	1056.55	**0.00**	1056.55	**0.00**	1056.55	**0.00**	1056.55	**0.00**	1056.55	**0.00**	1235.18	16.91	0.03
P10V100	2384.83	**0.00**	2384.83	**0.00**	2384.83	**0.00**	2384.83	**0.00**	2384.83	**0.00**	2773.85	16.31	0.04
P10V200	5867.26	31.17	5867.26	31.17	4473.05	**0.00**	4473.05	**0.00**	4473.05	**0.00**	5202.27	16.30	0.05
P100V500	9383.67	**0.00**	9383.67	**0.00**	13009.54	38.64	13009.54	38.64	11576.89	23.37	12246.69	30.51	1.65
P100V800	19193.98	30.99	19193.98	30.99	20307.24	38.59	20307.24	38.59	18167.49	**23.99**	19136.61	30.60	3.95
P100V1000	18173.00	**0.00**	18173.00	**0.00**	25242.07	38.90	25242.07	38.90	22435.15	23.45	23694.01	30.38	5.35
P500V1000	17826.67	**0.89**	17826.67	**0.89**	23605.77	33.59	23605.77	33.59	20958.00	18.61	23564.68	33.36	107.46
P500V2000	35916.81	0.89	35916.81	0.89	35819.47	**0.62**	35819.47	**0.62**	42020.26	18.04	47193.15	32.57	127.36
P500V5000	121805.82	35.40	121805.82	35.40	90541.67	**0.65**	90541.67	**0.65**	105456.55	17.23	118852.80	32.12	514.69

Note: $\text{Gap(algorithm)} = (f(\text{algorithm}) - f([\text{M2}]))/f([\text{M2}]) \cdot 100\%$; CT=Computing time.

Table 8. Comparison of algorithm and MILP results in tight scenario when T is set to infinite

Instance	VDPT		VAPT		VDPE		$G(f)$		VAPE		$G(f^E)$		[M2]
	f	Gap	f	Gap	f	Gap	f	Gap	f	Gap	f	Gap	CT/s
P10V50	1360.60	**0.82**	1375.60	1.93	1360.60	0.82	1531.15	13.46	1421.70	5.35	1609.68	19.28	0.19
P10V100	2945.14	**1.68**	3055.16	5.48	3932.68	35.78	3175.61	9.64	3741.91	29.19	4137.39	42.85	0.33
P10V200	6044.25	**1.96**	6079.56	2.55	7803.09	31.63	6523.62	10.04	7612.33	28.41	8326.62	40.46	0.81
P100V500	14451.35	5.88	14272.67	**4.57**	16897.77	23.81	16724.47	22.54	22022.99	61.36	25521.03	86.99	168.46
P100V800	23512.90	5.66	23241.30	**4.44**	27401.37	23.14	27192.90	22.20	36489.00	63.97	41162.56	84.97	216.17
P100V1000	28995.32	5.38	28809.61	**4.71**	33561.56	21.98	33636.82	22.25	45210.19	64.32	51532.47	87.30	763.41
P500V1000	26874.17	**1.83**	27221.42	3.15	33289.22	26.14	33642.08	27.48	44754.68	69.59	76121.21	188.44	3600.00
P500V2000	52587.42	**3.60**	52767.11	3.96	65535.13	29.11	66004.67	30.04	82593.29	62.72	150718.37	196.93	3600.00
P500V5000	129880.54	**2.94**	131348.40	4.11	161299.51	27.85	161746.14	28.20	220349.34	74.65	371405.76	194.38	3600.00

Note: $\text{Gap(algorithm)} = (f(\text{MILP}) - f(\text{MILP}))$

The results of instances in tight scenario are compared in Table 8. In term of the total energy consumption, [VDPT] outperforms other exact algorithms and gets gaps lower than 6.00% compared to [M2]. [VAPT] performs 1.00% higher gap than [VDPT] in most instances, while it gets

1.00% better results than [VDPT] in medium-scale instances. The good performances of these two algorithms are resulted from the criterion that selects the PM with lowest hosting time to host the VMs. And better solution is achieved by prior inserting the VMs with longer execution time. VDPT generates good solutions only in small instance, and its gap rises to higher than 20% when the problem scale increases. The gaps of $G(f)$ become larger when the number of VMs increases. When the number of VMs increases, the gaps of $G(f)$ fluctuate within 5%, which indicates the stability of $G(f)$. The performance of [VAPE] deteriorate with increasing problem scale, and its gap reaches 69.59% which is the highest in variants of the Algorithm 1. $G(f^E)$ performs the worst on the instances with infinite T, since it aims at minimizing the working energy consumption when insert an unassigned VM. $G(f^E)$ is not competitive for solving the problem that its gap exceeds 100% when the problem grows to 500 PMs and 1000 VMs.

The scalability of the algorithms is demonstrated through solving the instances whose problem size (represented by $|V| \cdot |P|$) differs from 500 to 2,500,000 in tight scenario. The computing times of algorithms with respect to the problem size are displayed in Figure 4. The computing time of solving [M2] is lower than 1 seconds for small-scale problem. However, when the problem size is larger than 50000, the computing time increases dramatically. The Algorithms 1-3 show a linear computing time with respect to the problem size. The computing times of variants of the Algorithm 1 are always lower than 2 seconds. They show more scalable than $G(f^E)$ for the differing problem size.

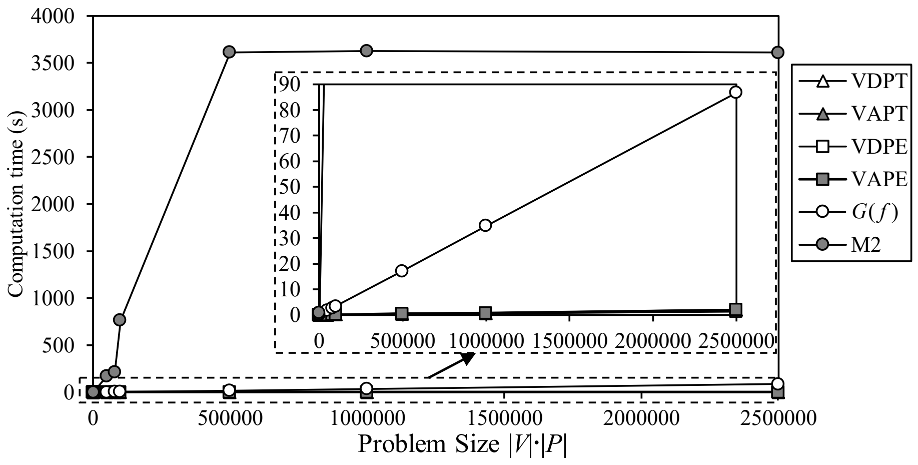

Figure 4. The scalability of the algorithms for differing size problems in tight scenario

6.3.4. Impact of makespan scale

The results of the Algorithms 1-3 and solving [M2] on the tight instance $P50V100l$ are displayed in Table 9. [VDPE] and [VDPT] outperform other exact algorithms in most instances when the makespan scale increases from 1.5 to 5. Especially, they get gaps smaller than 4% when the makespan scale is lower than 5. $G(f^E)$ performs the best when the makespan scale is 5. But compared to [VDPE] and [VDPT] in other instances, $G(f^E)$ performs worse, as well as $G(f)$. [VAPT] and [VAPE] always generate the same results, and the gaps of their solutions trend to decrease with α increasing. In the perspective of computing time, exact algorithms except $G(f)$ solve the problem in less one second. Despite solving [M2] incurs a plenty computing time (more than one hour) required by the MILP solver, the results from [M2] are the lowest in all the solutions.

Table 10 displays the results of the Algorithms 1-3 and [M2] on instance $P50V100t$ with the makespan scale increasing from 1.5 to 5. Although the solver for solving [M2] is terminated at the computing time limit, it still generates feasible solutions for all the instances. [VDPE] and [VDPT] outperform other exact algorithms when α is lower than 3.5, while [VAPT] and [VAPE] perform better when α increases. Notably, [VDPE] always obtains a gap of solution within 2% compared to the best objective values returned by other solution methods. $G(f^E)$ and $G(f)$ performs worse and the gap of $G(f)$ increases when the makespan scale grows larger.

The numbers of active PMs resulted from solving [M2] and the variants of the Algorithm 1 are displayed in Figure 5. [VAPE] and [VAPT] always require the highest number of PMs to host the VMs, while [VDPE] and [VDPT] need less PMs which is close to the result of solving [M2]. The number of PMs decreases when the makespan scale increases, and the number of PMs of most algorithms drops to 11 when the makespan scale reaches 5.

Table 9. Comparison of results on $P50V100l$ under different makespan scale

α	VDPE		VDPT		VAPT		VAPE		$G(f^E)$		$G(f)$		[M2]
	Gap	CT/s	Gap	CT/s	Gap	CT/s	Gap	CT/s	Gap	CT/s	Gap	CT/s	CT/s
1.5	**2.19**	0.14	2.97	0.14	18.54	0.10	18.54	0.10	16.93	0.19	8.45	2.89	3600
2	3.32	0.11	**3.30**	0.11	10.29	0.08	10.29	0.08	9.23	0.19	12.51	2.99	3600
2.5	**2.63**	0.09	2.99	0.09	10.17	0.07	10.17	0.07	10.25	0.19	14.74	2.93	3600
3	**2.32**	0.07	2.36	0.08	7.04	0.06	7.04	0.06	7.89	0.19	16.74	3.00	3600
3.5	3.82	0.06	**3.70**	0.06	5.78	0.05	5.78	0.05	6.53	0.19	17.54	2.93	3600
4	**3.35**	0.06	3.51	0.06	6.21	0.05	6.21	0.05	7.32	0.19	18.79	3.01	1971
4.5	**4.11**	0.05	4.21	0.05	4.58	0.04	4.58	0.04	9.16	0.19	19.24	2.91	3600
5	4.25	0.05	4.26	0.05	3.84	0.04	3.84	0.04	**3.10**	0.20	19.82	3.03	3600

Note: $\mathrm{Gap(algorithm)} = (f(\mathrm{algorithm}) - f(\mathrm{MILP}))/f(\mathrm{MILP}) \cdot 100\%$; CT=Computing time.

Table 10. The comparison of results on $P50V100t$ under different makespan scale

α	VDPE		VDPT		VAPT		VAPE		$G(f^E)$		$G(f)$		[M2]
	Gap	CT/s	Gap	CT/s	Gap	CT/s	Gap	CT/s	Gap	CT/s	Gap	CT/s	CT/s
1.5	**2.65**	0.18	2.69	0.23	14.18	0.14	14.11	0.13	15.73	0.29	9.48	4.23	3600
2	4.20	0.22	**4.15**	0.14	8.44	0.20	7.57	0.14	8.48	0.32	12.23	4.44	3600
2.5	**4.39**	0.11	4.63	0.11	8.97	0.09	9.04	0.09	7.70	0.24	14.01	4.18	3600
3	**5.58**	0.09	5.71	0.10	9.95	0.07	8.74	0.07	8.69	0.24	15.47	4.14	3600
3.5	8.40	0.08	7.27	0.08	**6.63**	0.06	7.11	0.06	10.19	0.25	16.40	4.17	3600
4	7.52	0.07	7.79	0.07	8.83	0.06	**6.05**	0.06	11.65	0.24	17.31	4.17	3600
4.5	7.73	0.07	8.90	0.07	**5.28**	0.05	7.87	0.05	10.37	0.24	17.77	4.17	3600
5	7.29	0.06	9.61	0.06	6.65	0.05	**5.46**	0.05	9.62	0.24	18.29	4.17	3600

Note: $\mathrm{Gap(algorithm)} = (f(\mathrm{algorithm}) - f(\mathrm{MILP}))/f(\mathrm{MILP}) \cdot 100\%$; CT=Computing time.

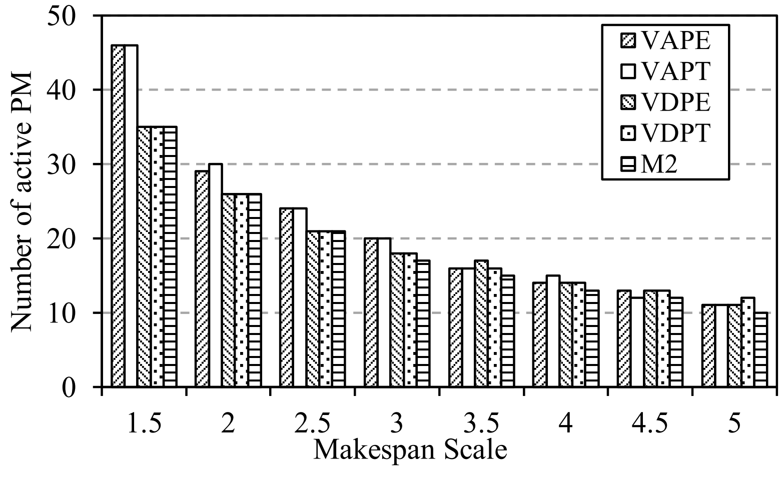

Figure 5. The impact of makespan scale on the number of PMs from different algorithms

The total energy consumptions resulted from the Algorithms 1-2 and [M2] are displayed in Figure 6. The results of [VDPT] and [VAPT] present decreasing trends in the energy consumptions when the makespan scale increases. The results of [VAPE] and $G(f^E)$ present high total consumptions at makespan scale 1.5 and the consumptions drop down when α increases. [VDPE] presents a stable total consumption with less fluctuation compared to other algorithms. The total energy consumption returned by solving [M2] decreases gradually and it holds a gap between other solution methods.

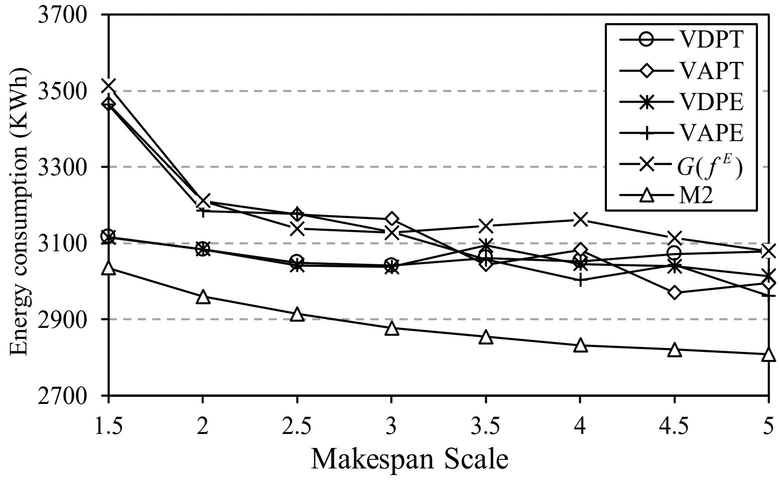

Figure 6. Impact of makespan scale on total energy consumption resulted from algorithms

The idle energy consumption solved by the Algorithm 1 and [M2] are depicted in Figure 7. High idle energy consumptions of [VAPT] and [VAPE] are observed at $\alpha = 1.5$. With the makespan scale increasing, the idle energy consumptions of [VAPT] and [VAPE] trend to decrease, while the ones of [VDPT] and [VDPE] trend to increase. The idle energy consumption returned by solving [M2] becomes stable when the makespan scale grows above 3.5.

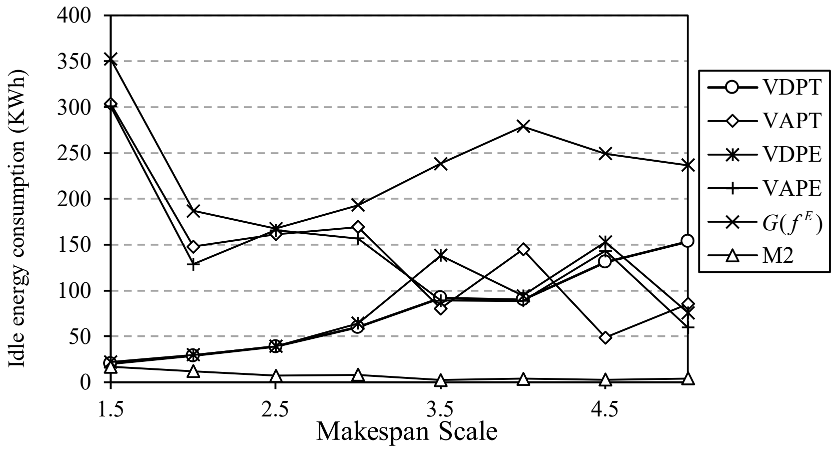

Figure 7. Impact of makespan scale on idle energy consumption resulted from algorithms

6.3.5. Discussion

(1) The results of the instances demonstrated the effectiveness of the Algorithm 1. The proposed propositions help to determine the insert criteria based on energy efficiency of PMs and hosting time of PMs. Near-optimal results are generated by the variants of the Algorithm 1 with small gaps that are lower than 5.8% compared to the ones from solving [M2]. In the perspective of scalability, the Algorithm 1 requires a linear computing time to solve the problem with respect to

the numbers of PMs and VMs.

(2) The performances of the variants of the Algorithm 1 depends on the capacities of the PMs. In the loose scenario, the algorithm preferentially assigning VM with short execution time performs better than the one assigning VM with long execution time; while in the tight scenario, assigning VM with long execution time prior to short execution time helps to reduce the total energy consumption. The variants with better performance show advantages in decreasing the idle energy consumption.

(3) The pre-determined makespan limit affects the total energy consumption and significantly worsens the performance of the Algorithms 2-3 that uses incremental energy consumption as the inserting criterion. The low objective values of [VDPE] indicate that for the makespan limited VMP, the best strategy is to preferentially assign VM with long execution time into PM with best-fit energy efficiency.

(4) Although the Algorithm 1 can solve the problem with small gap compared to the results of solving [M2], there still exist gaps between the variants of the Algorithm 1 using different combinations of strategies [VA], [VD], [PE] and [PT]. [VD] is effective for the problem with makespan limit on PM schedule, while [VA] performs better in loose scenario with unlimited makespan. [PE] incurs additional idle energy consumptions compared to the Algorithm 1 using [PT], however, [PE] saves working energy consumptions so that the total consumption is reduced when the makespan is limited.

(5) Different combinations of [VA], [VD], [PE] and [PT] produce different optimality. The gaps to the results of solving [M2] even range from 0 to 75%. Therefore, different implementations of the FFD and BFD heuristics presents big differences in terms of optimality. The comparisons to the FFD and BFD can not stand when these variances are not implemented and specified.

7. Conclusion

This study devised a MILP for the VMP problem considering the makespan, the energy consumption and the idle energy of active PMs. To solve the energy-efficient model by MILP solvers using Branch and Bound algorithms, in the context of QoS management, the makespan upper bound is formulated as a constraint. The model is then investigated mathematically to reveal the model nature, and the bounds of PMs and energy related to the makespan. Three exact heuristics algorithms are developed based on first-fit and best-fit criteria, and greedy strategies of minimizing the energy consumption. Data sets are generated based on data center profiles. The numerical results indicate that the MILP can be solved within a reasonable time for small-scale instances and be used to compute the ideal lower bounds for the heuristics algorithms; the exact heuristics algorithms are competitive for solving medium- and large-scale instances; especially, the algorithms that preferentially assign VMs with short execution time outperforms other algorithms when the capacity of PMs satisfies most VMs, and criterion of energy consumption outweighs the one of hosting time when the placement is limited under given makespan. These insights can be used for combining fit rules and criteria according to the PM capacities and makespan limits.

Inspired by the reviews as presented in Table 1, this study is a start to apply exact algorithms for VMP problems. In real-world data centers, stable and fast solution algorithms are important to ensure the QoS in the computational service supply chains and save energy. Although the devised models and exact algorithms can solve large-scale instances in this study, they cannot capture the future mega-scale data centers. As for researches on the way, exact heuristics algorithms based on model decompositions and dynamic solution methods based on machine learning algorithms will be developed and tested. Additionally, this study focused on the basic VMP while the solution methods can be extended for more general and practical scenarios. Variants of basic VMP emerge when the data centers become acceptable to various organizations and companies. For example, thread-based VMP may challenge the bin-packing model because the memory may be shared among cores and even threads; the VMs may migrate among PMs to reduce energy consumption and increase service quality. New extended features of the VMP will be studied in the future.

Acknowledgments

We are grateful to the anonymous referees for their helpful comments and questions. This work was partially supported by the National Natural Science Foundation of China (No. 71871136), the Australian Research Council grant (DP160104292) and the Science and Technology Commission of Shanghai Municipality (No. 17DZ2280200). YGW also wishes to thank the support from ARC Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS).

References

[1] M. Masdari, S.S. Nabavi, V. Ahmadi, An overview of virtual machine placement schemes in cloud computing, Journal of Network and Computer Applications, 66 (2016) 106-127.

[2] J. Arjona Aroca, A. Fernández Anta, M.A. Mosteiro, C. Thraves, L. Wang, Power-efficient assignment of virtual machines to physical machines, Future Generation Computer Systems, 54 (2016) 82-94.

[3] D. Versick, I. Waßmann, D. Tavangarian, Power consumption estimation of CPU and peripheral components in virtual machines, ACM SIGAPP Applied Computing Review, 13 (2013) 17-25.

[4] A. Kansal, F. Zhao, J. Liu, N. Kothari, A.A. Bhattacharya, Virtual machine power metering and provisioning, in: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC '10, 2010, pp. 39-50.

[5] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, H. Tenhunen, Using Ant Colony System to Consolidate VMs for Green Cloud Computing, IEEE Transactions on Services Computing, 8 (2015) 187-198.

[6] P. Arroba, J.M. Moya, J.L. Ayala, R. Buyya, Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers, Concurrency Computation, 29 (2017).

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

[8] Google Data Center, https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md (Accessed Oct 28, 2019).

[9] E.N. Alkhanak, S.P. Lee, R. Rezaei, R.M. Parizi, Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues, Journal of Systems and Software, 113 (2016) 1-26.

[10] X. Liu, R. Deng, K.R. Choo, Y. Yang, H. Pang, Privacy-Preserving Outsourced Calculation Toolkit in the Cloud, IEEE Transactions on Dependable and Secure Computing, (2018) 10.1109/TDSC.2018.2816656.

[11] E.N. Alkhanak, S.P. Lee, S.U.R. Khan, Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities, Future Generation Computer Systems, 50 (2015) 3-21.

[12] S. Mazumdar, M. Pranzo, Power efficient server consolidation for Cloud data center, Future

Generation Computer Systems, 70 (2017) 4-16.

[13] K. An, S. Shekhar, F. Caglar, A. Gokhale, S. Sastry, A cloud middleware for assuring performance and high availability of soft real-time applications, Journal of Systems Architecture, 60 (2014) 757-769.

[14] M. Tavana, S. Shahdi-Pashaki, E. Teymourian, F.J. Santos-Arteaga, M. Komaki, A discrete cuckoo optimization algorithm for consolidation in cloud computing, Computers and Industrial Engineering, 115 (2018) 495-511.

[15] J.O. Gutierrez-Garcia, A. Ramirez-Nafarrate, Agent-based load balancing in Cloud data centers, Cluster Computing, 18 (2015) 1041-1062.

[16] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, Journal of Computer and System Sciences, 79 (2013) 1230-1242.

[17] M. Riahi, S. Krichen, A multi-objective decision support framework for virtual machine placement in cloud data centers: a real case study, Journal of Supercomputing, 74 (2018) 2984-3015.

[18] K.M. Baalamurugan, S. Vijay Bhanu, A multi-objective krill herd algorithm for virtual machine placement in cloud computing, Journal of Supercomputing, (2018).

[19] F. Alharbi, Y.C. Tian, M. Tang, W.Z. Zhang, C. Peng, M. Fei, An Ant Colony System for energy-efficient dynamic Virtual Machine Placement in data centers, Expert Systems with Applications, 120 (2019) 228-238.

[20] X.F. Liu, Z.H. Zhan, J. Zhang, An energy aware unified ant colony system for dynamic virtual machine placement in cloud computing, Energies, 10 (2017).

[21] M.H. Malekloo, N. Kara, M. El Barachi, An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments, Sustainable Computing: Informatics and Systems, 17 (2018) 9-24.

[22] A.L. Stolyar, An infinite server system with general packing constraints, Operations Research, 61 (2013) 1200-1217.

[23] C. Kim, C. Jeon, W. Lee, S. Yang, A parallel migration scheme for fast virtual machine relocation on a cloud cluster, Journal of Supercomputing, 71 (2015) 4623-4645.

[24] W. Zhu, Y. Zhuang, L. Zhang, A three-dimensional virtual resource scheduling method for energy saving in cloud computing, Future Generation Computer Systems, 69 (2017) 66-74.

[25] Z. Li, C. Yan, X. Yu, N. Yu, Bayesian network-based Virtual Machines consolidation method, Future Generation Computer Systems, 69 (2017) 75-87.

[26] Z. Li, Y. Li, T. Yuan, S. Chen, S. Jiang, Chemical reaction optimization for virtual machine placement in cloud computing, Applied Intelligence, 49 (2019) 220-232.

[27] A. Satpathy, S.K. Addya, A.K. Turuk, B. Majhi, G. Sahoo, Crow search based virtual machine placement strategy in cloud data centers with live migration, Computers and Electrical Engineering, 69 (2018) 334-350.

[28] S.M. Sait, A. Bala, A.H. El-Maleh, Cuckoo search based resource optimization of datacenters, Applied Intelligence, 44 (2016) 489-506.

[29] W. Ding, C. Gu, F. Luo, Y. Chang, U. Rugwiro, X. Li, G. Wen, DFA-VMP: An efficient and secure virtual machine placement strategy under cloud environment, Peer-to-Peer Networking and Applications, 11 (2018) 318-333.

[30] H. Hallawi, J. Mehnen, H. He, Multi-Capacity Combinatorial Ordering GA in Application to Cloud resources allocation and efficient virtual machines consolidation, Future Generation Computer

Systems, 69 (2017) 1-10.

[31] M. Vasudevan, Y.C. Tian, M. Tang, E. Kozan, W. Zhang, Profile-based dynamic application assignment with a repairing genetic algorithm for greener data centers, Journal of Supercomputing, 73 (2017) 3977-3998.

[32] Y. Zhao, H. Liu, Y. Wang, Z. Zhang, D. Zuo, Reducing the upfront cost of private clouds with clairvoyant virtual machine placement, Journal of Supercomputing, 75 (2019) 340-369.

[33] M.K. Gupta, T. Amgoth, Resource-aware virtual machine placement algorithm for IaaS cloud, Journal of Supercomputing, 74 (2018) 122-140.

[34] C. Canali, R. Lancellotti, Scalable and automatic virtual machines placement based on behavioral similarities, Computing, 99 (2017) 575-595.

[35] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.M. Chao, J. Li, Virtual machine consolidated placement based on multi-objective biogeography-based optimization, Future Generation Computer Systems, 54 (2016) 95-122.

[36] M. Terra-Neves, I. Lynce, V. Manquinho, Virtual machine consolidation using constraint-based multi-objective optimization, Journal of Heuristics, (2018).

[37] D.S. Johnson, Fast algorithms for bin packing, Journal of Computer and System Sciences, 8 (1974) 272-314.

[38] W. Fernandez de la Vega, G.S. Lueker, Bin packing can be solved within 1 + ε in linear time, Combinatorica, 1 (1981) 349-355.

[39] S.P. Fekete, J. Schepers, New classes of fast lower bounds for bin packing problems, Mathematical Programming, Series B, 91 (2001) 11-31.

[40] J.M. Valério De Carvalho, LP models for bin packing and cutting stock problems, European Journal of Operational Research, 141 (2002) 253-273.

[41] K. Fleszar, K.S. Hindi, New heuristics for one-dimensional bin-packing, Computers and Operations Research, 29 (2002) 821-839.

[42] A.C.F. Alvim, C.C. Ribeiro, F. Glover, D.J. Aloise, A hybrid improvement heuristic for the one-dimensional bin packing problem, Journal of Heuristics, 10 (2004) 205-229.

[43] B. Byholm, I. Porres, Fast algorithms for fragmentable items bin packing, Journal of Heuristics, 24 (2018) 697-723.

[44] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, S. Mirjalili, An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems, Personal and Ubiquitous Computing, 22 (2018) 1117-1132.

[45] M. Ritt, A.M. Costa, Improved integer programming models for simple assembly line balancing and related problems, International Transactions in Operational Research, 25 (2018) 1345-1359.

[46] Z. Wang, X. Han, G. Dósa, Z. Tuza, A General Bin Packing Game: Interest Taken into Account, Algorithmica, 80 (2018) 1534-1555.