Computers & Industrial Engineering 115 (2018) 495–511

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A discrete cuckoo optimization algorithm for consolidation in cloud computing

Madjid Tavana[a,b,*], Saleh Shahdi-Pashaki[c], Ehsan Teymourian[d], Francisco J. Santos-Arteaga[e], Mohammad Komaki[f]

[a] Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, USA
[b] Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, D-33098 Paderborn, Germany
[c] Department of Industrial Engineering, Faculty of Engineering, Shomal University, Amol, Iran
[d] Department of Management Science and Information Systems, Rutgers University, New Brunswick and Newark, USA
[e] Faculty of Economics and Management, Free University of Bolzano, Bolzano, Italy
[f] Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, USA

ARTICLE INFO

Keywords:
Cloud computing
Group technology
Mathematical model
Virtual machine
Cuckoo optimization algorithm

ABSTRACT

Consolidation problems in cloud computing (CC) encompass server consolidation, virtual machine (VM) consolidation, and task consolidation. These problems have become increasingly challenging for resource allocation in distributed systems. Group technology (GT) has been effectively used to manage resource allocation problems by reducing manufacturing costs and increasing system productivity. We propose a discrete cuckoo optimization algorithm (DCOA) based on GT for consolidation in CC. The proposed model is designed to control manufacturing costs (i.e., energy, penalty, VM creating, and task migration). The DCOA developed in this study contains several new adjustments that allow it to solve large-sized discrete problems, including a grouping strategy based on the Jaccard similarity coefficient, as well as modified egg laying and immigration processes. A numerical example is used to demonstrate the applicability of the proposed model and exhibit the efficacy of the DCOA. The results illustrate the quality superiority of the DCOA over the first fit (FF) and round robin (RR) algorithms, and the efficiency and effectiveness superiority of the DCOA over the genetic algorithm (GA).

1. Introduction

Cloud computing (CC) is a new computational technology arising from the advance of multiple technologies, such as grid computing, parallel computing, and distributed computing. The main difference between CC and other technologies is that it provides virtualized resources to users. A cloud service provider (SP) offers computing resources as virtual machines (VMs) via the Internet. Users utilize these resources based on their demand and pay their costs according to a pay-as-you-go model. In general, the main goal of CC is to provide multiple services, though it focuses on offering three major ones. The layers of CC include: (1) infrastructure as a service, (2) platform as a service, and (3) software as a service (Buyya, Broberg, & Goscinski, 2010; Selvarani & Sadhasivam, 2010). Fig. 1 describes these main layers.

One of the main goals of CC is optimizing the computational costs of its beneficiaries (e.g., SPs and users). At the same time, users want to minimize some direct costs (i.e., hardware and software) by transferring the computational tasks to the CC environment. Moreover, SPs seek to maximize their profits by reducing the costs of resource utilization. Therefore, effective approaches to manage hardware are essential to control these costs.

1.1. Consolidation problems

In order to manage the available resources properly, CC applies a key technology called virtualization, which focuses on consolidating the main components of CC, i.e., server, VM and task (Ferreto, Netto, Calheiros, & De Rose, 2011). This is called the consolidation problem and is divided into three sub-problems; *server consolidation*, which refers to the process of combining the workloads of several different servers on a set of target servers (Speitkamp & Bichler, 2010; Zhang, Liu, Liu, Zheng, & Zhang, 2015), *VM consolidation*, which refers to the consolidation of VMs into a single physical server (Hsu, Slagter, Chen, & Chung, 2014; Luo, Li, & Chen, 2014), and *task consolidation*, which

* Corresponding author at: Business Systems and Analytics Department, Distinguished Chair of Business Analytics, La Salle University, Philadelphia, PA 19141, USA
E-mail addresses: tavana@lasalle.edu (M. Tavana), s.shahdi@ustmb.ac.ir (S. Shahdi-Pashaki), ehsan.teymourian@rutgers.edu (E. Teymourian), fsantosarteaga@unibz.it (F.J. Santos-Arteaga), gxk152@case.edu (M. Komaki).
URL: http://tavana.us (M. Tavana).

https://doi.org/10.1016/j.cie.2017.12.001
Received 28 March 2017; Received in revised form 24 November 2017; Accepted 1 December 2017
Available online 06 December 2017
0360-8352/ © 2017 Elsevier Ltd. All rights reserved.

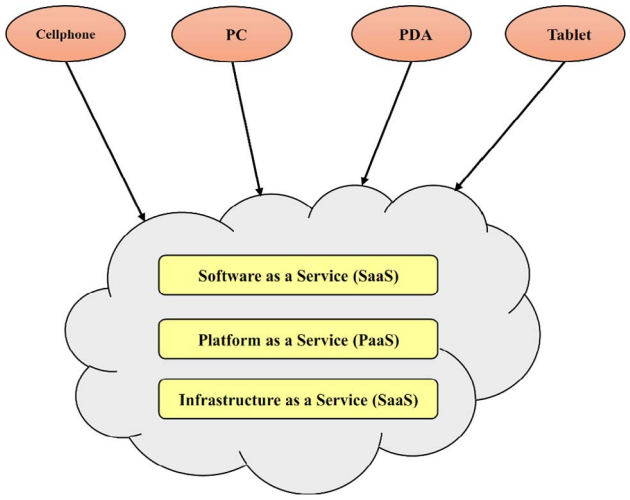


Fig. 1. Cloud computing layers.

refers to the process of minimizing resource usage by improving utilization (Masson et al., 2013). Appropriate decision-making is necessary to maintain a significant level of cost control for these problems.

Generally, in consolidation problems, the cost of energy consumption contributes over 75% of the operating costs in servers (Belady, 2007). Although large power usage is expensive, energy usage varies from one data center to another for many reasons. First, the energy price may vary from one region to another, and the servers, being distributed all over of the world, may be subject to different tax legislation. Second, the energy consumed by the servers depends on the types of hardware that they utilize. For instance, servers with faster and higher-capacity hard drives, larger fans or water cooling need to consume more energy than others. Under these circumstances, the operational costs of servers differ from each other. Consequently, making the right decision about turning some servers off or putting them into sleep state (e.g., server consolidation) becomes critical (Luo et al., 2014). Similarly, the number of running VMs is another factor affecting energy consumption of a server (i.e., dynamic energy consumption which is consumed only when the server is running the VMs) and depends on various factors. For example, memory access operations, such as loading and storing, use different sets of execution units compared to arithmetic operations. Thus, energy consumption can be reduced by using a smaller number of VMs, including effective ways to pin them on the servers (e.g. VM consolidation).

In addition, task consolidation costs are another factor that must be considered in resource management. In this context, an important challenge is how to assign tasks to servers so that the data exchanging costs (i.e., task migration cost) between servers is minimal. Migration costs become significant for data-intensive workflow applications with complex dependencies and a large quantity of data (e.g., hundreds of gigabytes in Montage (Deelman et al., 2005)). Figs. 2 and 3 show electronic funds transfer as workflow and task migration, respectively.

The other important cost in conjunction with task consolidation is the service level agreement (SLA) violation cost (i.e., penalty cost). In general, there are two types of applications in CC, namely, service applications and batch applications (Selvarani & Sadhasivam, 2010). Service applications are typically response time-sensitive, whereas batch applications are throughput-sensitive. In this work, we focus on batch applications. Many tasks, particularly in e-banking (e.g., check processing, daily interest calculations and automated clearing house transactions) should be processed into high-volume batches. For instance, ACH networks process large volumes of transactions in batches at specified times. In the ACH, the penalty cost includes the transactions that are not processed. Thus, providing a solution for a resource provisioning problem that considers the major goals of SPs (i.e., server, VM

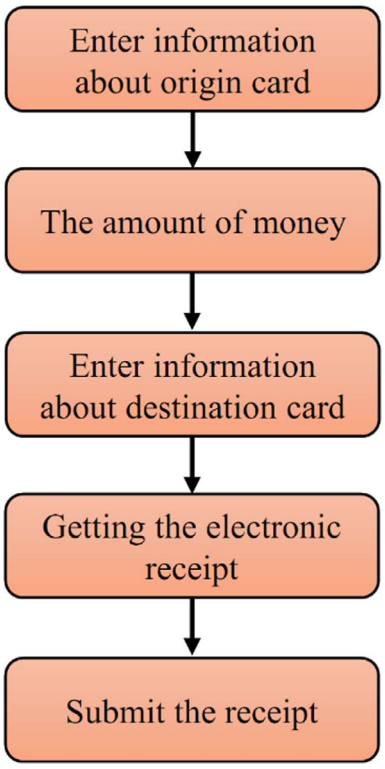


Fig. 2. Workflow of electronic funds transfer.

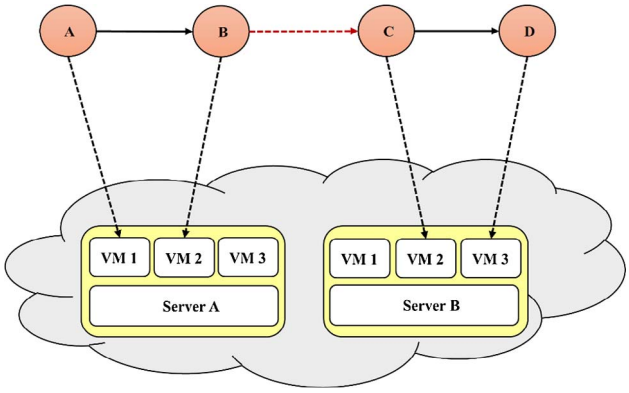


Fig. 3. Resource allocation with migration.

and task consolidation costs) can reduce the operating costs of a system.

All the costs mentioned above are unavoidable, which obliges SPs to employ an appropriate resource management policy to control them. For this purpose, we present a new approach to handle all consolidation problems motivated by the theory and practice of group technology (GT). In the following subsection, we provide the necessary background information about GT.

1.2. Background and definition of GT

GT was first developed by Shahdi-Pashaki, Teymourian, Booyavi, and Alizaheh (2015); Shahdi-Pashaki, Teymourian, Kayvanfar, Komaki, and Sajadi (2015) to consolidate the resources and tasks in CC environments. In fact, GT is a resource management technique for reducing production costs in manufacturing systems, which was first introduced by Mitrovanov (1966), and subsequently promoted by Burbidge (1971). Similarly, we exploit it to minimize cost when dealing with the consolidation problem. In this technique, similar components (parts) in terms of geometry, manufacturing operations, tools, etc. are

Table 1
A binary machine-part incidence matrix.

		Parts				
		P_1	P_2	P_3	P_4	P
Machine	M_1	0	1	0	1	1
	M_2	1	0	1	0	0
	M_3	0	1	0	0	1
	M_4	1	0	1	0	0

assembled together to increase flexibility and reduce system costs (Chan, Lau, Chan, & Lo, 2008; Mahesh & Srinivasan, 2002; Wemmerlöv & Hyer, 1989). One of the main applications of GT is called a cellular manufacturing system (CMS). The purpose of a CMS is to create machine groups (i.e., cells), which process similar components (i.e., part family) that are used for implementing new production strategies (e.g., just-in-time and lean production) (Wemmerlöv & Hyer, 1989). In this regard, the aim of cell formation (CF) is to optimize measures (e.g., setup and material handling cost) (Lei & Wu, 2005). A standard CF (SCF) simply takes into account the operation of parts and pays no attention to other manufacturing factors. In SCF, a production system can be described as a binary machine-part incidence matrix $[a_{ij}]$, where the element '1' ('0') states that part j requires (does not require) to be processed by machine i. This matrix $[a_{ij}]$ does not show machines and parts clustering. Thus, a grouping algorithm is needed to modify it. Intuition regarding the grouping concept is provided in Table 1 (Cheng, Goh, & Lee, 1996).

Table 1 displays machine-part incidence with no recognizable groups. However, by changing the position of the rows and columns, a matrix with a diagonal block structure can be generated, as shown in Table 2.

Two diagonal groups are visible in Table 2. In this matrix, there are two machine groups ($MG_1 = \{M_2, M_4\}$ and $MG_2 = \{M_1, M_3\}$), and two corresponding part families ($PG_1 = \{P_1, P_3\}$ and $PG_2 = \{P_2, P_4, P_5\}$). Mutually separable groups with completely independent machine groups, such as those described in Table 2, are not common in real-life problems. Table 3 describes a more realistic case.

A grouping algorithm can be used to generate two machine groups ($MG_1 = \{M_1, M_2\}$ and $MG_2 = \{M_3, M_4\}$), and two part families ($PG_1 = \{P_1, P_2\}$ and $PG_2 = \{P_3, P_4, P_5\}$), such as those shown in Table 1. GT often gives results containing exceptional elements, which take place when a part (e.g., part 5 in Table 3) must move between machine groups to finish all its operations. The two groups generated in this way are called independent groups. In general, decision-making about the production of parts (tasks) in cells (servers) and the minimization of the number of part movements (task migration in cloud) are the main (common) objectives of both CC and GT. Therefore, taking advantage of GT to enhance the performance of CC is a worthwhile project to pursue.

Li, Liu, and Luo (2017) recently conducted a similar study in multimedia cloud by proposing a content distribution method based on interest discovery and integrated utility of the users. They found the user's interest by using an improved feature extraction algorithm. They then grouped users in adjacent regions based on their similarity in service interest. They also proposed an improved extremum disturbed particle swarm optimization algorithm to minimize content distribution time and user cost. The main contributions of this paper are twofold:

Table 2
A matrix with a diagonal block structure.

			Parts				
			PG_1		PG_2		
			P_1	P_3	P_2	P_4	P_5
Machine	MG_1	M_2	1	1	0	0	0
		M_4	1	1	0	0	0
	MG_2	M_1	0	0	1	1	1
		M_3	0	0	1	0	1

Table 3
A realistic solution.

			Parts				
			PG_1		PG_2		
			P_1	P_2	P_3	P_4	P_5
Machine	MG_1	M_1	1	1	0	0	1
		M_2	1	1	0	0	0
	MG_2	M_3	0	0	1	1	1
		M_4	0	0	1	0	0

(a) Proposing a novel mathematical model based on GT to:
- control the energy consumption in servers by switching the idle servers off,
- determine the number and types of the VMs on the servers, and
- determine the task migration and penalty costs by finding an optimal task-VM and VM-server consolidation.

(b) Proposing a novel version of the COA with new definitions, features, and procedures based on discrete scheme representations to solve large problems (i.e., egg-laying, new young cuckoos grouping, immigration of cuckoos, etc.).

The remainder of the paper is structured as follows. A literature review focusing on resource allocation in CC is presented in Section 2. Section 3 presents the problem description and the mathematical model for GT. Section 4 describes a new version of the cuckoo optimization algorithm (COA) required by the discrete nature of the problem. Computational results and their analysis are presented in Section 5. Finally, Section 6 concludes.

2. Literature review

Several solutions have been presented for the resource management problem with the aim of cost optimization in CC. Most of the studies develop heuristic and meta-heuristic approaches and can be classified in three groups: (1) server consolidation approaches, (2) VM consolidation approaches and (3) task consolidation approaches. In the following, we discuss each one in detail.

2.1. Server consolidation approaches

In the context of server consolidation, many studies focusing mainly on energy have been conducted. For example, Pinheiro, Bianchini, Carrera, and Heath (2001) presented an approach to control power consumption by switching off idle nodes for serving multiple web-applications. Their approach also ensures reliable quality of service (i.e., throughput and time) in SLA but, in contrast to our work, doesn't consider other operational costs, i.e., penalty cost for SLA violations, data transfer cost and VM consolidation costs. The approach for energy management in homogeneous environments proposed by Chase, Anderson, Thakar, Vahdat, and Doyle (2001) selects a dynamic set of active servers according to the demand and request load of applications and switches idle servers to sleep or hibernation mode to save power consumption. These two approaches address only energy management for server consolidation costs and do not offer a mechanism to control SLA violations. Elnozahy, Kistler, and Rajamony (2002) used two techniques for power saving, namely, switching power of computing nodes on/off and Dynamic Voltage and Frequency Scaling (DVFS) to

determine the optimal resources in a single web-application environment. The response time is considered as fixed SLA content. A different approach is suggested by Nathuji and Schwan (2007) to optimize the energy consumption of virtual server's consolidation while maintaining SLAs. Speitkamp and Bichler (2010) present a decision model to find the exact solution for the server consolidation problem, considering real-world constraints. Since the problem is NP-hard, they applied a heuristic to address large-scale server consolidation projects. In addition, a preprocessing method for server load data allowing for the consideration of quality-of-service levels was also introduced.

2.2. VM consolidation approaches

VMs comprise the backend of most CC services and their consolidation constitutes an important area of research. Cardosa, Korupolu, and Singh (2009) proposed a VM allocation method for power saving in virtualized heterogeneous computing environments. One of the greatest limitations of this approach is that it does not support strict SLAs. Verma, Ahuja, and Neogi (2008) presented a heuristic algorithm for VM consolidation that optimizes the power consumption each time. Their approach does not consider strict SLA requirements or task consolidation costs. An approach was introduced by Srikantaiah, Kansal, and Zhao (2008) for VM-server assignment to control the energy consumption. Their model determines the energy consumption of different consolidations and then selects the most energy-efficient one. Goudarzi, Ghasemazar, and Pedram (2012) presented a heuristic for the VM consolidation problem to optimize the number of serviced applications and migration costs, as well as to balance the load in physical machines. These authors defined a SLA-based VM consolidation problem to minimize the total operational cost (i.e., power, migration and penalty costs in CC). Gao, Guan, Qi, Hou, and Liu (2013) applied a multi-objective ant colony algorithm to the VM placement problem. Their algorithm provided a set of non-dominated solutions to control the resource wastage and power consumption simultaneously. Erdem, Kiraz, Eski, Çiftçi, and Kubat (2016) used a multi-agent systems approach to represent the interactions between virtual laboratories in a cloud system. They also studied the likely entities and their communication, in addition to building a conceptual framework, for modeling cloud-based integration of virtual laboratories.

2.3. Task consolidation approaches

Task consolidation is also a popular topic for the resource management problem and a considerable number of publications has been devoted to this topic. For example, a heuristic algorithm was introduced by Goudarzi et al. (2012) for task consolidation to control total energy consumption in CC. However, the proposed heuristic did not capture the effect of SLA on VM resource provisioning. Lee and Zomaya (2012) defined an energy-aware task consolidation approach. They introduced two heuristics for task-resource allocation to control the active and idle energy consumption without performance degradation.

Other studies have employed more than one approach in order to control more factors simultaneously. In this regard, a new power management approach (i.e., soft resource scaling) is proposed by Nathuji and Schwan (2007). They used a combination of hardware scaling and VMs consolidation to reach higher power savings. Task consolidation costs (i.e., transfer cost and SLA violations cost) are not considered in their approach. Kusic, Kephart, Hanson, Kandasamy, and Jiang (2009) investigated the problem of power management in virtualized heterogeneous environments using limited lookahead control. They aimed at minimizing the power consumption and SLA violations as well as maximizing the profit of the SPs. Lee and Zomaya (2009) proposed a hybrid genetic algorithm and energy-conscious scheduling heuristic to simultaneously optimize makespan and energy consumption. Their method was shown to outperform other methods such as DBUS and HEFT. A biphasic energy-aware approach for scheduling the resources in data centers was introduced by Zhang, Yan, Wu, and Luo (2013). They defined a model to determine the energy consumption in servers and then proposed an algorithm, called best-fit-decreasing-power, to control the energy consumption and SLA violations. Hsu et al. (2014) proposed an energy-aware task consolidation to distribute tasks among virtual clusters. Although task migration was considered to control the energy, their model did not tackle all aspects of the problem such as servers and VMs. In this regard, designing proper strategies to cover consolidation problems (e.g., servers, VMs and tasks) can reduce computational costs and increase the performance of CC.

Additionally, several other studies address the resource management problem using different goals and solution approaches (i.e. heuristics and meta-heuristics) (Ergu, Kou, Peng, Shi, & Shi, 2013; Li, 2009; Pandey, Wu, Guru, & Buyya, 2010; Shahdi-Pashaki, Teymourian, Kayvanfar, et al., 2015; Shahdi-Pashaki, Teymourian, & Tavakkoli-Moghaddam, 2016; Zhong, Tao, & Zhang, 2010; Liu, Jin, Chen, Liu, Yuan, & Yang, 2010; Liu, Yin, Yasuda, & Lian, 2010; Hu, Gu, Sun, & Zhao, 2010; Wei, Vasilakos, Zheng, & Xiong, 2010; Xu, Zhao, Hu, & Hu, 2011).

All the previous studies address the problem of cost efficient resource management to optimize some consolidation costs (i.e., server consolidation, VM consolidation and task consolidation). However, none of them considers the costs associated with all consolidations simultaneously. In this regard, a cost aware mathematical model based on GT is presented in the current paper to investigate the problem of resource management in CC. The proposed model is organized to control operational costs in small sized problems. Moreover, for larger size problems, where finding an optimal solution using exact methods is computationally expensive, a COA is applied to find a near optimal solution. Some of the previous studies, especially those dealing with server consolidation, have considered the costs associated with server consolidation together with a mechanism for SLA assurance. Their methods are based on the decision to either switch the idle servers on/off or putting them in power saving mode (e.g., sleep, hibernation) to control consumption and SLA violations.

In contrast to our work, these studies do not consider several other operational costs existing in VM and task consolidation levels. Some studies did however address the costs related to the VM consolidation level. Two types of VM consolidation exist in CC, namely, initial placement (Cardosa, Singh, Pucha, and Chandra (2010)) and migration (and/or resizing) of VMs over time (Bobroff, Kochut, & Beaty, 2007; Malet & Pietzuch, 2010). In these studies, different goals have been investigated according to the type of consolidation i.e., task migration cost, energy costs and penalty costs. However, the number of VMs as a factor affecting both dynamic energy consumption and user satisfaction was not considered. Finally, another group of studies investigated task consolidation problems with different goals. But none of the previous studies considers the costs associated with all consolidation levels simultaneously.

2.4. GT and cell formation problem

GT focuses on grouping similar tasks/objects to improve the system proficiency. Many studies have been conducted on GT to investigate various aspects of this theory. A two-phase approach to minimize the intra and inter-cell movements for a cell formation problem (CFP) was proposed by Wemmerlöv and Hyer (1989). In the first phase, a multi-objective mathematical model is applied to form machine cells and part families while, in the second phase, a single-objective mathematical model is proposed to minimize the total intra and inter-cell movements. A multi-objective GA was proposed to solve a CMS problem by Solimanpur, Vrat, and Shankar (2004). Their model is able to search in multiple dimensions of the solution space and specify the directions of search systematically. A linear integer programming approach for studying a dynamic cell formation problem (DCFP) where demand and product mix change in each period was developed by Tavakkoli-

Moghaddam, Safaei, and Sassani (2008). These authors minimized the inter-cell movement and machine costs and used simulated annealing to solve the resulting problem. An algorithm was suggested by Mahdavi and Mahadevan (2008) to determine the cells and the sequence of machines in cells for a CMS. Li (2009) proposed a three-phase heuristic algorithm to study the CFP. They considered several production factors including production volume, batch size, alternative process routings, and cell size. A CFP with multifunctional machines in two phases was studied by Mahdavi, Baher, and Teymourian (2010) to achieve higher flexibility of a system. They also presented a mathematical model to minimize the dissimilarity of machines in a cell. Zhao and Wu (2000) proposed a GA to optimize inter and intra-cell movements cost, cell load variation, and exceptional elements for a CFP. Nair and Narendran (1998) developed a non-hierarchical clustering approach to create machine and part groups considering production sequence data. Mahdavi, Teymourian, Baher, and Kayvanfar (2013) considered cell formation and the cell layout problem simultaneously. They defined a new integrated mathematical model to minimize forward against backtracking movements. Other studies have used metaheuristic approaches to solve group/batch scheduling problems with respect to resource allocation and/or task grouping and sequencing metrics (e.g., Shahvaria, Salmasib, & Logendranc, 2009; Shahvari & Logendran, 2017).

The goals of the current study are to propose a mathematical model based on GT for: (1) server consolidation and minimizing the overall energy consumption; (2) VM consolidation and determining the number and type of VMs running on servers; and (3) task consolidation for controlling task migration and penalty costs. Moreover, we intend to propose a new version of the COA adapted to deal with large integer optimization problems.

3. Problem description based on GT

In this study, we assume a CC environment composed of K servers, P VMs, and J tasks. Each server can offer several different VM types in terms of their capacity for users to execute tasks in a specific time period. It is assumed that the servers have specific characteristics, such as constant capacity for the CPU, storage space, and memory which are also provided by the VM types at an abstract level. The tasks can be represented as workflows with pre-defined demand, processing time and quality of service, which are incorporated in a SLA. Each workflow consists of subtasks series that must be executed on their required VMs according to their priorities. Under these circumstances, SPs should consolidate servers as well as VMs in response to the users' requests in a cost-effective way to control energy, migration, and penalty costs. A simple example is presented in Tables 4 and 5 to elucidate the cloud consolidation problem-based GT.

Table 4 provides an example of a binary Task-VM incidence matrix in a CC system, where an entry of '1' ('0') indicates that VM type i is required (not required) to execute task j. Table 5 shows the final matrix obtained after the diagonalization process, in which two groups of VMs are created to consolidate in servers 1 and 2. Likewise, two groups of tasks (i.e., TG_1 and TG_2) are defined to consolidate into VMs. All of the three tasks of workflow 2 (i.e., T_1W_2, T_3W_2, T_4W_2) are consolidated in server 2, though T_2W_2 must migrate to server 1 for execution. Thus, the main objectives of a CCPGT are to consolidate the servers, VMs, and tasks simultaneously so as to control different consolidation cost (i.e., power and task migration costs).

Table 4
A binary Task-VM incidence matrix.

		Tasks						
		T_1W_1	T_2W_1	T_3W_1	T_1W_2	T_2W_2	T_3W_2	T_4W_2
VM	VM_1	1	0	0	1	0	0	0
	VM_2	0	0	0	0	1	1	0
	VM_3	0	1	1	0	0	0	1

Table 5
Diagonalization process with two groups of VMs within a CC system.

			Tasks						
			TG_1			TG_2			
			T_1W_1	T_2W_1	T_3W_1	T_1W_2	T_2W_2	T_3W_2	T_4W_2
VM	$Server_1$	VM_1	1	0	0	0	0	0	0
		VM_2	0	0	0	0	1	0	0
		VM_3	0	1	0	0	0	0	0
	$Server_2$	VM_1	0	0	1	1	0	0	0
		VM_2	0	0	0	0	0	1	0
		VM_3	0	0	0	0	0	0	1

The notations used in the problem formulation are defined as follows:

3.1. Nomenclatures

Indices:

j	Task ($j = 1,\ldots, J$)
i	Subtask ($i = 1,\ldots, I$)
p	VM type ($p = 1,\ldots, P$)
k	Server ($k = 1,\ldots, K$)
M	Large positive number

Parameters:

T_{ijp}	Processing time for subtask i of task j on machine p
a_{ijp}	1 if subtask i of task j will require machine type p; 0, otherwise
L_p	The capacity of machine type p
C_k	The capacity of server k
D_{SLA}	Demand(number) of task j stated in the SLA
UCE	The unit cost of energy consumption
CTM	The cost of task migration
CVM_p	The cost of creation of the VM type
PC	Penalty cost

Decision variables:

X_{ijpk}	1 if subtask i of task j is done on machine type p in server k; 0, otherwise
N_{pk}	Number of type p machines assigned to server k;
Y_{ijk}	1 if subtask i of task j is assigned to server k; 0, otherwise
V_k	1 if server k is used (turned on); 0, otherwise
DS_j	1 if the demand of task j is not met; 0, otherwise
D_j	Amount of the demand of task j that is satisfied by the service provider
H_k	The number of tasks assigned to server k
S_j	1 if there is a shortage for task j; 0, otherwise

3.2. Mathematical formulation

Using above the notations, the proposed model is given as follows:

$$F = \alpha \sum_{i}^{I} \sum_{j}^{J} \sum_{k}^{K} D_j \cdot CTM_j \cdot [Y_{ijk} \cdot (1 - Y_{i+1jk})] + \beta \sum_{k}^{K} UCE_k \cdot V_k \left(\sum_{i}^{I} \sum_{j}^{J} \sum_{p}^{P} X_{ijpk} \cdot T_{ijp} \right) + \gamma \sum_{p}^{P} UCE_p \left(\sum_{i}^{I} \sum_{j}^{J} \sum_{k}^{K} X_{ijpk} \cdot T_{ijp} \right) + \delta \sum_{p}^{P} \sum_{k}^{K} CVM_p \cdot N_{pk} + \sum_{j}^{J} (DS_j - D_j) \cdot S_j \cdot PC_j \quad (1)$$

s.t.

$$X_{ijpk} = a_{ijp} \cdot Y_{ijk} \quad \forall\, i,j,p,k \quad (2)$$

$$X_{ijpk} \leqslant N_{pk} \quad \forall\, i,j,p,k \quad (3)$$

$$\sum_{p}^{P} \sum_{k}^{K} X_{ijpk} = 1 \quad \forall\, i,j \quad (4)$$

$$\sum_{i}^{I} \sum_{j}^{J} D_j \cdot T_{ijp} \cdot X_{ijpk} \leqslant L_p \cdot N_{pk} \quad \forall\, p,k \quad (5)$$

$$\sum_{p}^{P} L_p \cdot N_{pk} \leqslant C_k \cdot V_k \quad \forall\, k \quad (6)$$

$$M(S_j - 1) < DS_j - D_j \quad \forall\, j \quad (7)$$

$$MS_j \geqslant DS_j - D_j \quad \forall\, j \quad (8)$$

$$\sum_{i}^{I} \sum_{j}^{J} Y_{ijk} = H_k \quad \forall\, k \quad (9)$$

$$M(V_k - 1) \leqslant H_k - 1 \quad \forall\, k \quad (10)$$

$$MV_k > H_k - 1 \quad \forall\, k \quad (11)$$

$X_{ijpk}, Y_{ijk}, V_k, DS_j, S_j$: binary $\quad \forall\, i,j,p,k$ (12)

N_{pk}, D_j, H_k: integer $\quad \forall\, i,j,k$ (13)

The objective function (1) consists of five terms, with the first one describing the total migration costs between the servers. Migration costs are imposed on the system when two sequential subtasks are not executed on one server. Therefore, the expression $[Y_{ijk}\ (1 - Y_{i+1jk})]$ indicates whether the current task is migrated ($Y_{ijk} = 1$, $Y_{i+1jk} = 0$) or not ($Y_{ijk} = 1$, $Y_{i+1jk} = 1$). The second term of the objective function represents the static energy cost. V_k determines the servers that are switched on ($V_k = 1$) or off ($V_k = 0$). If the server is turned on, its energy cost is calculated multiplying the unit cost of energy consumption (UCE_k) by the time required for the server to complete the different tasks ($X_{ijpk} \cdot T_{ijp}$). The third term of the objective function represents the dynamic energy cost (Seo, Jeong, Park, & Lee, 2008), which is determined multiplying the unit cost of energy consumption (UCE_p) by each VM ($X_{ijpk} \cdot T_{ijp}$). The fourth term of the objective function accounts for the number of VMs and has therefore been multiplied by the cost factor of each VM (CVM_p) so as to be comparable with other terms in the objective function. Each of these terms has preferences (α, β, γ, and δ) determined by the user. Finally, the fifth term calculates the total penalty cost for tasks that are not being met ($DS_j - D_j \geqslant 0$).

Constraint (2) describes the relationship between the variables X_{ijpk} and Y_{jpk}, implying that subtask i of task j is executed on VM p in server k ($X_{ijpk} = 1$) only if conditions ($Y_{ijk} = 1$) and ($a_{ijp} = 1$) are simultaneously satisfied. $Y_{ijk} = 1$ indicates that subtask i of task j is executed in server k, while $a_{ijp} = 1$ states that subtask i of task j needs machine type p. Constraints (3) and (4) ensure that each task is assigned to exactly one VM and one server. Moreover, tasks are allocated to the server that provides their required VMs. Constraints (5) and (6) guarantee that the VMs and servers' capacities are not exceeded. They state that the total capacity of the tasks assigned to a VM and the total capacity of the VMs assigned to a server must be lower than or equal to the total capacity of the VM and the server, respectively. Together, Constraints (7) and (8) represent shortage ($S_j = 1$) or lack of it ($S_j = 0$). Constraints (9)–(11) show the use ($V_k = 1$) or non-use ($V_k = 0$) of servers and indicate that a server is used only when at least one task is assigned to it ($H_k > 0$). Finally, binary and non-negative integer decision variables are introduced in Constraints (12) and (13).

4. Cuckoo optimization algorithm

The COA proposed by Rajabioun (2011) is an evolutionary optimization technique that imitates the lifestyle of the cuckoo bird. There are some differences in the habits and lifestyle of cuckoos compared to other birds. The COA starts with an initial population of cuckoos. Each cuckoo naturally has some eggs (from 5 to 20) that are laid in the nest of some other birds. The eggs that are most similar to the eggs of the host bird have more chances of becoming an adult cuckoo. Other eggs are identified and destroyed by the host birds. Since the eggs that survive become mature cuckoos, they form groups in a specific habitat for the living. Thus, the number of mature cuckoos in an area is a measure of its suitability. The best habitat among all the cuckoo groups in terms of the survival rate of eggs becomes the goal of the cuckoos in other groups. After identifying the goal habitat, all cuckoos begin to migrate towards it. During the migration process, they only traverse a part of the distance (% λ of all distance) with a given amount of deviation (Q) and they begin to lay their eggs within a certain radius called the egg laying radius (ELR). Fig. 4 schematically shows the migration and egg laying model of the cuckoos.

The COA has been successfully tested on a wide range of optimization problems. Azarbad, Ebrahimzadeh, and Addeh (2015) applied the COA for digital communication signals. Their computational results showed the high accuracy of their algorithm. Teoh, Wibowo, and Ngadiman (2014) compared the COA and GA for the university course time-tabling problem. Their findings indicated the superiority of the COA over GA in terms of both quality and convergence speed. Kaydani and Mohebbi (2013) applied the COA to permeability prediction and showed that it has a faster convergence rate than PSO. Ameryan, Akbarzadeh Totonchi, and Seyyed Mahdavi (2014) proposed a COA model to optimize intra-cluster distance summation in clustering problems. They concluded that the COA outperforms PSO, the imperialist competitive algorithm, and K-means clustering. Komaki, Teymourian, Kayvanfar, and Booyavi (2017) designed an improved DCOA to minimize completion time in three-stage assembly flow shop scheduling problems. Shahdi-Pashaki, Teymourian, Kayvanfar, et al. (2015) proposed a COA model to control the operational costs in resource allocation problems in CC.

In addition to the COA (Rajabioun, 2011), a similar algorithm called cuckoo search proposed by Yang and Deb (2010) has been widely used to solve optimization problems (Hanoun, Nahavandi, Creighton, & Kull, 2012; Komaki, Sheikh, Teymourian, & Malakooti, 2015; Li & Yin, 2013; Liu & Ye, 2013; Marichelvam, 2012; Marichelvam, Prabaharan, & Yang, 2014; Nie, Liu, Xie, Liu, & Yang, 2014; Ouaarab, Ahiod, & Yang, 2014).

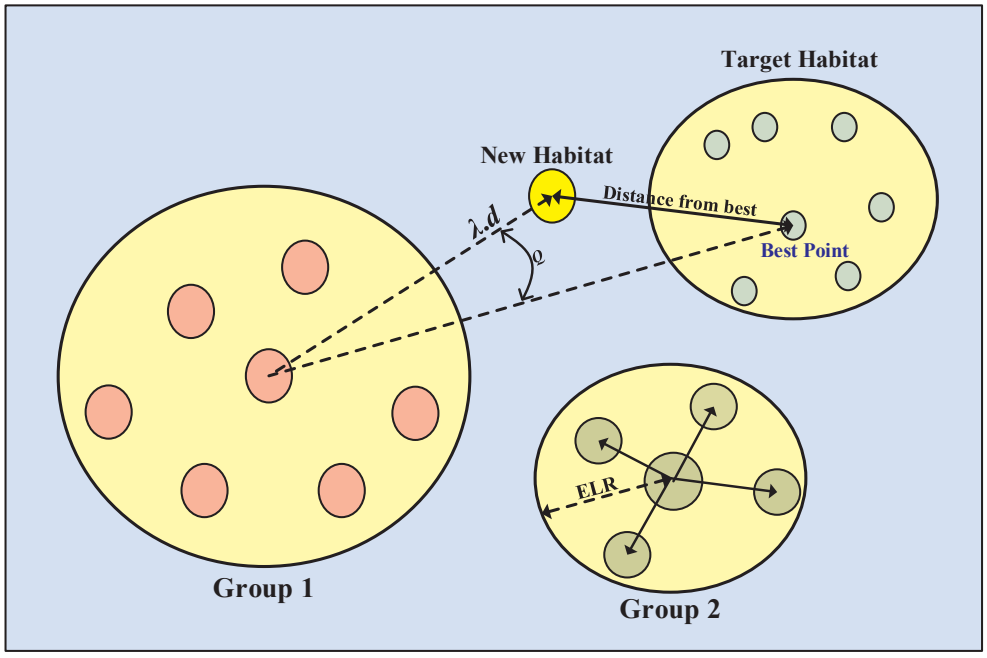


Fig. 4. Immigration and egg laying model of cuckoos.

Table 6
Habitat structure.

Habitat											
X				*Y*				*N*			
x_{11}	x_{12}	…	x_{1j}	y_{11}	y_{12}	…	y_{1j}	N_{11}	N_{12}	…	N_{1j}
x_{21}	x_{22}	…	x_{2j}	y_{21}	y_{22}	…	y_{2j}	N_{21}	N_{22}	…	N_{2j}
…	…	…	…	…	…	…	…	…	…	…	…
x_{i1}	x_{i2}	…	x_{ij}	y_{i1}	y_{i2}	…	y_{ij}	N_{i1}	N_{i2}	…	N_{ij}

4.1. Discreet cuckoo optimization algorithm

Since the original version of the COA is designed for continuous problems, it is necessary to modify or redefine some parts of the algorithm to adapt it to discrete problems. Hence, a discrete version of the COA (DCOA) is designed by redefining the immigration and ELR processes. Also, a new technique based on the Jaccard index is defined for grouping the cuckoos. The procedure of the DCOA is described below.

4.1.1. Habitat structure of the proposed DCOA

The representation of solutions has a great impact on the performance of all meta-heuristic algorithms. In the COA, each solution is represented by an array called the habitat. The structure of the habitat for the problem being considered is presented in Table 6.

The habitat is composed of three matrices (i.e., *X*, *Y* and *N*). Matrix *X* and *Y* are used to assign subtasks to VMs and subtasks to servers, with their elements varying from 1 to *p* and from 1 to *k*, respectively. Matrix *N* relates to the number of VMs in servers and its elements are positive integers. Matrix *X* is constant in all the habitats and is determined by the possibility of executing a task matrix. For example, $X_{ij} = m$ and $Y_{ij} = k$ means that subtask *i* of task *j* is done on VM type *m* in server *k*, simultaneously. Also, $N_{pk} = \varsigma$, $\varsigma > 0$ states that the number of VMs in

1	**For** all habitat;
2	**For each** habitat;
3	**While** (number of solution <Maximum number of egg)**, Do**
4	**Calculate** the dynamic ELR value by Eq.1;
5	**Generate** a random value within dynamic ELR as dissimilar rate between two solution;
6	**Calculate** the number of different elements ($ND^{ab}_{element}$) of two solutions by Eq. (17) # *section 4.1.6* #;
7	**Select** randomly the $ND^{ab}_{element}$ elements of matrix Y from current solution and change their values up to $\max_{k\in[1,c]} k$;
8	**Update** the elements of matrix N;
9	**End While**
10	**End**
11	**End**

Fig. 5. Egg laying procedure.

Table 7
An egg laying example.

X			Y			N		
(a) Current habitat								
1	1	2	3	2	1	1	2	4
3	3	1	3	1	1	5	2	3
3	2	3	1	1	1	4	1	2
(b) New habitat								
1	1	2	2	2	1	1	2	4
3	3	1	3	1	2	5	2	3
3	2	3	1	3	1	4	1	2

server k is equal to ς .

4.1.2. Generating the initial cuckoo habitat

To start the proposed *DCOA,* an initial habitat population is generated randomly as follows. First, matrix N is randomly initialized so that the server capacity is not exceeded. Then, the elements of matrix Y are initialized considering the possibilities defined by the tasks execution matrix (X).

4.1.3. Profit function

The profit value is a criterion for the suitability rate of an area (i.e., habitat) to raise cuckoo eggs. The profit of a habitat is obtained using a profit function, which, in this paper, is the same as the objective function of the model. Since the COA maximizes profit, we multiply the objective of the proposed model by minus one in order to minimize cost.

4.1.4. Egg laying

After determining the initial habitats, the cuckoos start laying eggs randomly in some other host birds' nests within their ELR. Since the original version of the ELR is designed for problems with a continuous space, a new version of the ELR compatible with a discrete solution space must be defined. Insofar as laying an egg in a certain radius is akin to applying the variable neighborhood (VNS) algorithm, so too, local search techniques from other algorithms can be used to search for new solutions. To this end, a mechanism is designed motivated by the mutation operator of the genetic algorithm (GA) to search for new solutions within a new ELR. The value of the ELR presented in (14) is proportional to the total number of eggs TN_{egg}, the number of current cuckoo's eggs $NC_{egg} = w_i\left(\frac{f_i}{f_{best}}\right)$, the profit function of the current habitat (f_i) and the size of the habitat (number of tasks, j, in the problem considered).

$$DELR_i = \alpha_i . j . \frac{NC_{egg}}{TN_{egg}} \tag{14}$$

where α_i is a parameter introduced to control the *DELR*, $w_i \sim$ U (L_{egg},U_{egg}), and the function $f\left(\frac{f_i}{f_{best}}\right)$ measures the effectiveness of the habitat for sustenance, the dynamic behavior of DELR for each habitat being proportional to its profit function. The larger the value of α_i, the better the global optimization capability. However, the computational effort of the DCOA can be excessive. Therefore, we introduce a dynamic α_i value to get a better convergence or intensification of the DCOA in the final iterations and more diversification in the early ones. To do so, we define a mechanism in Equation (15) – motivated by the cooling function in Simulated Annealing (SA) – that delivers α_i in each iteration

$$\alpha_i = \alpha_{\max} - N_i \frac{(\alpha_{\max} - \alpha_{\min})}{TN_i} \tag{15}$$

where α_{min} and α_{max} are the minimum and maximum limits of α_i through the DCOA, respectively. Also, N_i and TN_i identify the iteration number whose α_i is being computed and the total number of iterations, respectively.

4.1.5. Neighborhood structures

After determining the dynamic ELR, the cuckoos start to lay eggs randomly within it. Hence, we define a local search procedure to generate new solutions in the neighborhood determined by the discrete nature of the solution space as well as the matrix structure of the habitats. To do so, some of the random elements of the current solution must be repeated identically in the new solution. Then, the values of the

1	**Begin**
2	**Generate** K individual cuckoos group;
3	**Assign** k cuckoos' groups with minimum out-group similarity;
4	**While** (number of live cuckoos <total number of live cuckoos)**, Do**
5	**assign** a cuckoo to one group so that maximize intra-group similarity;
6	**End While**
7	**End**

Fig. 6. Pseudo code of group organizing.

1	**Begin**
3	**For** each habitat (i) in population**, Do**;
4	**Assign** λ_i to each habitat by Eq. (18);
6	**select** λ_i elements of target habitat randomly;
7	**generate** new habitat by copying the selected elements into same position in habitat i;
8	**Evaluate** new solution**;**
9	**If** the fitness of new habitat be better than habitat i; **Do**
10	**Replace** new habitat with habitat i **;**
11	**End If**
12	**End Fore**
13	**End**

Fig. 7. Pseudo code of immigration.

Table 8
An example of immigration.

X			Y			N		
(a) Target habitat								
1	2	3	2	3	3	2	1	2
3	3	2	2	3	1	1	5	5
2	2	1	1	2	2	4	2	2
(b) Current habitat								
2	3	2	1	1	1	1	3	4
2	3	3	3	1	1	3	1	4
1	2	1	3	1	2	5	2	2
(c) New habitat								
1	2	3	2	3	1	2	1	2
3	3	2	3	3	1	1	5	5
2	2	1	1	1	2	4	2	2

Table 9
VM repairing.

X			Y			N		
(a) Repaired habitat								
1	1	2	3	2	1	1	2	4
3	1	1	3	1	2	5	2	3
3	2	1	1	1	1	4	2	2
(b) New habitat								
1	1	2	3	2	1	1	2	4
3	1	1	3	1	2	5	2	3
3	2	1	1	1	1	4	2	2

Table 10
Server repairing.

X			Y			N		
(a) Current habitat								
1	1	2	3	2	1	1	2	4
3	3	1	3	1	1	5	2	3
3	2	3	1	1	1	4	2	2
(b) Repaired habitat								
1	1	2	3	2	1	1	3	3
3	3	1	3	1	1	5	2	3
3	2	3	1	1	1	4	2	2

remaining elements have to be changed based on the maximum and minimum values defined for each element. The pseudo-code and an example of Egg laying are presented in Fig. 5 and Table 7.

It is worth noting that, in the Egg laying process, first the elements of matrix Y are randomly changed up to $\max k_{k\in[1,c]}$ and then the elements of matrix N are updated according to these changes.

4.1.6. Growing up and group organizing

After all the eggs are dumped in the nests of other species, those less similar to the host birds' own eggs (% p of those with the worst profit function) will be identified and destroyed. The remaining ones will be hosted in the nests of the host birds to become chicks and grow up to mature cuckoos. They then form distinctive groups of cuckoos and breed until the laying season. Rajabioun (2011) suggested a K-means clustering approach for cuckoos grouping in continuous problems but, in this paper, a new grouping approach is proposed using the *Jaccard distance*. The Jaccard index, also known as the *Jaccard similarity coefficient*, is a statistic used to compare the similarity of sample sets and is defined in Eq. (16)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{16}$$

We use Eq. (17) as a Jaccard similarity coefficient for grouping the cuckoos

$$S_{ab}^{=}\frac{NI_{element}^{ab}}{TN_{element}^{ab}} \tag{17}$$

where S_{ab} is the similarity between solutions a and b, $NI_{element}^{ab}$ is the number of identical elements obtained by two solutions and $TN_{element}^{ab}$ is the total elements of the solutions, which is equal to $M \times N$ for a solution with M rows and N columns. Fig. 6 shows the Cuckoos group organizing process.

4.1.7. Immigration

During the laying season, the cuckoos migrate instinctively to the habitat with the maximum similarity between their own eggs and those of the host birds to increase the probability of birth. Thus, the new habitat that they reside in after immigration should be more similar to the target habitat than the initial one. The cuckoos don't fly all the way when moving towards the target habitat, they only fly a part of the way with a deviation. Given the matrix nature of the solutions, we define a migration operator that is consistent with the discrete space of the problem. This operator first selects λ random elements of the target point and copies them identically in the same position of the current

1	**Initialize** parameters K, E_{max}, E_{min}, C_{max},
2	**Generate** K initial habitat;
3	**While**(Iteration <Max Iter)**, Do**
4	**Procedure** *egg laying*; *#section 4.1.4 #*
5	**Delete** the eggs that are identified by the host birds ;
6	**If** (current population of Cuckoo > C_{max}), **Do**
7	**Delete** some eggs randomly;
8	**Else**
10	**Calculate** the profit of each cuckoo habitat; *#section 4.1.3 #*
11	**Select** target habitat
13	**Exert** *immigration operator*; *#section 4.1.7 #*
14	**Procedure** *Repairing*; *#section 4.1.8 #*
15	**Evaluate** the habitats;
16	**Replace** *the new habitats*;
17	**End IF**
18	**End While**

Fig. 8. Pseudo-code of the proposed COA.

solution. The other elements remain unchanged. As mentioned earlier, the distance traveled towards the target point is determined by λ. Thus, larger values of λ are assigned to solutions that are farther from the target points (or solutions with worse fitness values). We define a function that determines the values of λ proportionally to the fitness of each solution

$$\lambda_i = d \times \mathrm{TN}_{\mathrm{elements}} \quad (18)$$

where λ_i is the number of elements of the current solution that must be changed, while $d \sim (0,1)$ allows for the selection of λ_i elements of each solution randomly. The immigration process and an example are provided in Fig. 7 and Table 8, respectively.

4.1.8. Repairing

It is possible that some habitats (i.e., solutions) become infeasible after applying the operators. A solution (habitat) is infeasible if the number of VMs installed on servers and the number of tasks running on VMs exceed their capacity. In this case, a procedure modifies the infeasible solutions. The repairing procedure is applied on some parts (*X* or *Y*, or both) of an infeasible habitat, which is changed by operators as follows:

***Repairing the VMs* (*matrix Y*):** in this case, if the number of tasks assigned to a VM exceeds its capacity, then some tasks are selected and moved randomly to other VMs, which can execute them until the capacity of the VM is modified. The subtask 2 of task 2 and subtask 3 of task 3 are executed on VM 1 in server 1 for the current habitat, as shown in Table 8. The capacity of VM 1 is therefore exceeded. Thus, repairing and creating the new habitat requires moving subtask 3 of task 3 to VM 1 in server 3 randomly. Table 9 presents an example of VM repairing.

***Repairing the servers* (*matrix N*):** in this case, if the number of VMs exceeds the capacity limit of a server, some VMs from the server are randomly selected and moved to another server. This process continues until the capacity of all the servers is modified. Table 10 provides a simple example of server repairing, where the number of VMs in server 3 has exceeded its capacity. Repairing requires moving a unit from VM 1 in server 3 to server 2.

4.2. Proposed DCOA

The overall structure of the proposed DCOA is presented as a pseudo-code in Fig. 8. The algorithm starts by generating the *K* initial solutions (i.e., cuckoos' habitat) randomly and then new solutions are generated around each solution (i.e., laying eggs). The worse solutions in terms of profit are removed and the remaining ones are brought closer to the best solution (i.e., immigration). After repairing the infeasible solutions, the new habitats are evaluated and replaced with the previous ones. This process will continue to get the maximum number of generations (*G*).

5. Computational results and analysis

In this section, we evaluate the performance of the mathematical model using a numerical example and that of the DCOA through random generated problems. Then, we analyze the computational results derived from the different approaches. For our experimental setup, we use the characteristics of Amazon large (*m*1.*x*, *m*2.*x* and *c*1.*x*) instances with a CPU of (8, 6.5 and 20) EC2 Compute Units, a memory of (15, 17.1 and 7) GB, and a cost of ($0.299/h, $0.345/h and $0.840/h) for Windows. The servers are composed of a set of Intel processors with different cores, cache, and power consumption (i.e., Atom, i5, i7 and Xeon). The energy cost is assumed to be $ 0.15 per KW/h. To generate the workflows, we use the Pegasus workflow generator. The workflows are realistic (i.e., LEAD Data Mining Workflow, Glimmer, MEME-MAST, caDSR and McStats) and borrowed from Ramakrishnan and Gannon (2008). In addition, other workflows are generated with random characteristics including: number of tasks (2–20), computation time (seconds-minutes) and data size (kilobytes to megabytes). Task execution and processing time matrices (*X* and *T*) with $s \times t$ random elements are also created. The elements of matrices *X* and *T* are random round numbers in the range of 1 to *m* and 1 to *p*, where *m* and *p* are the maximum number of VMs and the maximum processing time of the tasks, respectively.

5.1. Numerical example

A numerical example is presented to illustrate the validity of the proposed mathematical model. To this end, a small example with size 3 × 3 × 3 (i.e., tasks × VM types × servers) is designed. The binary Task-VM incidence matrix is presented in Table 11 while Table 12 illustrates the processing time matrix of each workflow.

The above example is run 3 times considering different preferences (*α, β, γ,* and *δ*) for the costs that comprise the objective function. The different scenarios are implemented in Lingo 9 software on a PC Pentium (R) Dual-core CPU with 2.50 GHz and 2 GB RAM, using Win 7 as the operating system.

The output results are described in Table 13, including three non-dominated solutions for the problem. Table 13(a) gives the final consolidation matrix with α = 1, β = 1, γ = 2, and δ = 4 preferences. In this case, the SP should consolidate one number of VM_2 and VM_3 in $server_2$ and one number of VM_1 in $server_3$ to execute two groups of tasks, i.e., TG_1 (T_1W_2, T_2W_2, T_3W_2) and TG_2 (T_1W_1, T_2W_1, T_3W_1, T_1W_3, T_2W_3, T_3W_3), and hibernate $server_1$ to reduce the energy cost. This consolidation process causes two migrations, i.e., T_1W_1, T_1W_2.

Table 13(b) provides a solution with α = 1, β = 1, γ = 4, and δ = 2 preferences. Similarly to the previous scenario, Table 13(b) shows a consolidation with one number of VM_2 and VM_3 in $server_2$ and one number of VM_1 and VM_3 in $server_3$. $Server_1$ is also hibernated but a different consolidation for the tasks, i.e., TG_1 (T_2W_2, T_3W_2, T_2 W_1, T_1W_2) and TG_2 (T_1W_1, T_3W_1, T_1W_3, T_2W_3, T_3W_3), is obtained relative to consolidation 1. In addition, four migrations are accrued.

Finally, Table 13(c) suggests a solution with $\alpha = 4$, $\beta = 2$, $\gamma = 1$, and $\delta = 1$ preferences. In this case, only $server_2$ is consolidated, running one number of each VM type (1, 2 and 3) to execute workflow 3, while $server_2$ and $server_3$ are hibernated. Since all the tasks of workflow

Table 11
Binary Task-VM incidence matrix for the 3 × 3 × 3 numerical example.

		Tasks								
		T_1W_1	T_2W_1	T_3W_1	T_1W_2	T_2W_2	T_3W_2	T_1W_3	T_2W_3	T_3W_3
VMs	VM_1	1	0	0	0	1	1	0	0	0
	VM_2	0	1	0	0	0	0	1	0	1
	VM_3	0	0	1	1	0	0	0	1	0

Table 12
Processing time matrix.

		Tasks								
		T_1W_1	T_2W_1	T_3W_1	T_1W_2	T_2W_2	T_3W_2	T_1W_3	T_2W_3	T_3W_3
VMs	VM_1	2	1	5	1	2	2	3	3	2
	VM_2	3	4	2	1	3	2	1	2	1
	VM_3	5	3	2	3	3	4	5	2	4

Table 13
Final consolidation matrix.

(a) Consolidation 1: ($\alpha = 1$, $\beta = 1$, $\gamma = 2$, and $\delta = 4$)

		TG 1			TG 2					
		T_2W_2	T_3W_2	T_1W_2	T_1W_1	T_2W_1	T_3W_1	T_1W_3	T_2W_3	T_3W_3
$Server_1$ (off)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	0	0	0	0	0	0	0
	VM_3	0	0	0	0	0	0	0	0	0
$Server_2$ (on)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	0	0	1	0	1	0	1
	VM_3	0	0	1	0	0	1	0	1	0
$Server_3$ (on)	VM_1	1	1	0	1	0	0	0	0	0
	VM_2	0	0	0	0	0	0	0	0	0
	VM_3	0	0	0	0	0	0	0	0	0

(b) Consolidation 2: ($\alpha = 1$, $\beta = 1$, $\gamma = 4$, and $\delta = 2$)

		TG_1			TG_2					
		T_2W_2	T_3W_2	T_2W_1	T_1W_2	T_1W_1	T_3W_1	T_1W_3	T_2W_3	T_3W_3
$Server_1$ (off)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	0	0	0	0	0	0	0
	VM_3	0	0	0	0	0	0	0	0	0
$Server_2$ (on)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	1	0	0	0	1	0	1
	VM_3	0	0	0	1	0	0	0	1	0
$Server_3$ (on)	VM_1	1	1	0	0	1	0	0	0	0
	VM_2	0	0	0	0	0	0	0	0	0
	VM_3	0	0	0	0	0	1	0	0	0

(c) Consolidation 3: ($\alpha = 4$, $\beta = 2$, $\gamma = 1$, and $\delta = 1$)

		TG_1						TG_2		
		T_1W_1	T_2W_1	T_3W_1	T_1W_2	T_2W_2	T_3W_2	T_1W_3	T_2W_3	T_3W_3
$Server_1$ (off)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	0	0	0	0	0	0	0
	VM_3	0	0	0	0	0	0	0	0	0
$Server_2$ (on)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	0	0	0	0	1	0	1
	VM_3	0	0	0	0	0	0	0	1	0
$Server_3$ (off)	VM_1	0	0	0	0	0	0	0	0	0
	VM_2	0	0	0	0	0	0	0	0	0
	VM_3	0	0	0	0	0	0	0	0	0

3 are consolidated to $server_2$, no task migration occurred.

Consolidation 3 provides a better solution than consolidation 1 and 2 in terms of server cost, by consolidating $server_2$ and hibernating servers 1 and 3. In contrast, consolidations 1 and 2 hibernate server 1 and use the other two servers. Similarly, consolidation 3, with no migration, performs better than consolidations 1 and 2 with respect to migration cost and obtains a better solution than consolidations 1 and 2 by 2 and 4 migrations, respectively. In terms of penalty costs, $consolidation_1$ and $consolidation_2$ obtain better results by executing all the tasks, in contrast to $consolidation_3$, which rejects workflows 1 and 2.

5.2. Implementation of DCOA

We apply the DCOA to a set of randomly generated problems (i.e. small and large size ones). The fine-tuning of the algorithmic parameters i.e., the number of initial cuckoos (K), the maximum number of iterations (Max_{Iter}), the maximum number of living cuckoos (C_{max}) and the minimum and maximum eggs of each cuckoo (E_{min}, E_{max}), is initially performed. The values of these parameters, which are obtained experimentally, are presented in Table 14. The DCOA is programmed using MATLAB.R2010 programming language on a PC Pentium (R)

Table 14
The DCOA parameter values.

Parameters of DCOA	Values
Number of Initial Cuckoos	5
Minimum Number of Eggs Laid by Each Cuckoo (E_{min})	2
Maximum Number of Eggs Laid by Each Cuckoo (E_{max})	4
Maximum Number of Iterations	1000
Number of cuckoos groups	4
Maximum Number of Living Cuckoos (C_{max})	10

Table 15
Numerical results of Model, DCOA, GA, RR, and FF for both small and large problems.

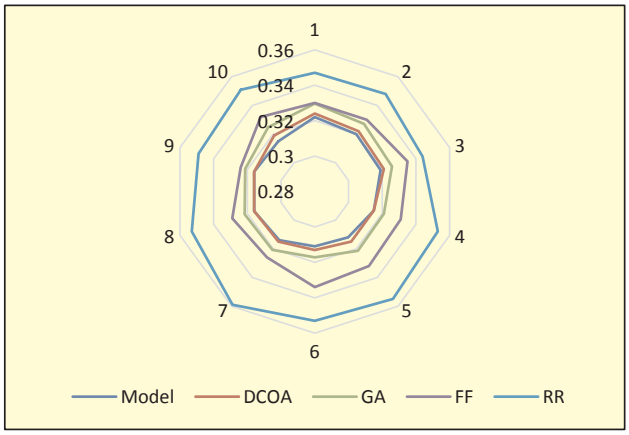
Problem ID	Energy cost										Migration cost			
	Model (Lingo)	DCOA	GA	RR	FF	RPD					Model (Lingo)	DCOA	GA	RR
						Model (Lingo)	DCOA	GA	RR	FF				
SSP-01	0.322	0.324	0.330	0.33	0.347	0.144	0.151	0.171	0.173	0.233	0.249	0.257	0.284	0.353
SSP-02	0.32	0.322	0.327	0.33	0.348	0.137	0.144	0.163	0.173	0.237	0.238	0.258	0.282	0.355
SSP-03	0.319	0.321	0.326	0.335	0.344	0.134	0.141	0.158	0.190	0.222	0.263	0.273	0.301	0.331
SSP-04	0.315	0.315	0.321	0.331	0.353	0.119	0.119	0.141	0.176	0.254	0.241	0.251	0.276	0.333
SSP-05	0.312	0.315	0.321	0.332	0.355	0.109	0.119	0.142	0.180	0.262	0.245	0.255	0.277	0.333
SSP-06	0.311	0.313	0.317	0.334	0.353	0.105	0.112	0.127	0.187	0.254	0.242	0.242	0.265	0.365
SSP-07	0.314	0.315	0.321	0.326	0.359	0.116	0.119	0.140	0.158	0.276	0.244	0.254	0.278	0.338
SSP-08	0.316	0.316	0.322	0.329	0.353	0.123	0.123	0.143	0.169	0.254	0.264	0.276	0.303	0.363
SSP-09	0.316	0.316	0.321	0.324	0.349	0.123	0.123	0.142	0.151	0.240	0.254	0.266	0.292	0.347
SSP-10	0.315	0.319	0.325	0.332	0.351	0.119	0.134	0.154	0.180	0.247	0.250	0.250	0.271	0.349
LSP-01	–	0.294	0.319	0.344	0.362	–	0.044	0.134	0.223	0.287	–	0.265	0.294	0.338
LSP-02	–	0.292	0.321	0.330	0.378	–	0.037	0.142	0.173	0.344	–	0.274	0.299	0.352
LSP-03	–	0.287	0.314	0.339	0.373	–	0.021	0.114	0.206	0.327	–	0.260	0.286	0.361
LSP-04	–	0.306	0.320	0.320	0.374	–	0.088	0.137	0.137	0.329	–	0.280	0.307	0.337
LSP-05	–	0.293	0.320	0.342	0.365	–	0.041	0.139	0.216	0.297	–	0.251	0.281	0.365
LSP-06	–	0.299	0.324	0.341	0.360	–	0.061	0.153	0.213	0.279	–	0.266	0.295	0.338
LSP-07	–	0.299	0.324	0.342	0.360	–	0.061	0.152	0.215	0.278	–	0.261	0.291	0.344
LSP-08	–	0.282	0.309	0.342	0.376	–	0.003	0.098	0.214	0.337	–	0.269	0.295	0.350
LSP-09	–	0.284	0.309	0.334	0.382	–	0.009	0.096	0.188	0.356	–	0.253	0.282	0.349
LSP-10	–	0.296	0.323	0.340	0.364	–	0.052	0.148	0.208	0.293	–	0.257	0.282	0.347
LSP-11	–	0.296	0.327	0.344	0.360	–	0.053	0.162	0.222	0.279	–	0.271	0.297	0.332
LSP-12	–	0.301	0.330	0.334	0.366	–	0.069	0.174	0.186	0.300	–	0.275	0.301	0.336
LSP-13	–	0.297	0.329	0.335	0.368	–	0.057	0.169	0.189	0.308	–	0.268	0.298	0.353
LSP-14	–	0.292	0.321	0.335	0.373	–	0.037	0.142	0.190	0.327	–	0.255	0.275	0.356
LSP-15	–	0.291	0.318	0.335	0.374	–	0.034	0.130	0.191	0.328	–	0.267	0.287	0.342
LSP-16	–	0.295	0.327	0.335	0.370	–	0.048	0.161	0.190	0.316	–	0.276	0.301	0.332
LSP-17	–	0.302	0.333	0.333	0.365	–	0.072	0.183	0.184	0.297	–	0.253	0.280	0.356
LSP-18	–	0.296	0.322	0.334	0.369	–	0.053	0.144	0.188	0.313	–	0.263	0.289	0.344
LSP-19	–	0.290	0.318	0.339	0.371	–	0.031	0.131	0.205	0.318	–	0.263	0.285	0.344
LSP-20	–	0.281	0.319	0.340	0.373	–	0.000	0.135	0.208	0.324	–	0.263	0.292	0.344

Problem ID	Migration cost					Penalty cost									
	RPD					Model (Lingo)	DCOA	GA	RR	FF	RPD				
	Model (Lingo)	DCOA	GA	RR	FF						Model (Lingo)	DCOA	GA	RR	FF
SSP-01	0.046	0.080	0.194	0.483	0.630	0.197	0.207	0.236	0.382	0.419	0.000	0.051	0.196	0.939	1.127
SSP-02	0.000	0.084	0.183	0.492	0.693	0.215	0.215	0.242	0.374	0.409	0.091	0.091	0.228	0.898	1.076
SSP-03	0.105	0.147	0.265	0.391	0.513	0.198	0.218	0.245	0.375	0.425	0.005	0.107	0.246	0.904	1.157
SSP-04	0.013	0.055	0.161	0.399	0.542	0.217	0.217	0.243	0.374	0.408	0.102	0.102	0.235	0.898	1.071
SSP-05	0.029	0.071	0.163	0.399	0.584	0.207	0.227	0.253	0.395	0.396	0.051	0.152	0.285	1.005	1.010
SSP-06	0.017	0.017	0.112	0.534	0.626	0.218	0.238	0.268	0.383	0.397	0.107	0.208	0.361	0.944	1.015
SSP-07	0.025	0.067	0.169	0.420	0.626	0.222	0.232	0.262	0.362	0.414	0.127	0.178	0.330	0.838	1.102
SSP-08	0.109	0.160	0.273	0.525	0.504	0.210	0.210	0.234	0.381	0.407	0.066	0.066	0.185	0.934	1.066
SSP-09	0.067	0.118	0.226	0.458	0.513	0.209	0.219	0.250	0.387	0.403	0.061	0.112	0.268	0.964	1.046
SSP-10	0.050	0.050	0.137	0.466	0.517	0.210	0.210	0.236	0.368	0.421	0.066	0.066	0.196	0.868	1.137
LSP-01	–	0.056	0.172	0.347	0.578	–	0.216	0.242	0.382	0.401	–	0.096	0.229	0.939	1.036
LSP-02	–	0.092	0.192	0.402	0.482	–	0.220	0.254	0.372	0.406	–	0.117	0.288	0.888	1.061

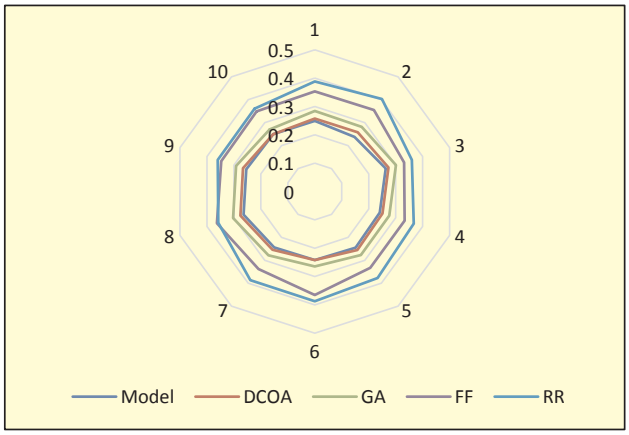
(continued on next page)

Table 15 *(continued)*

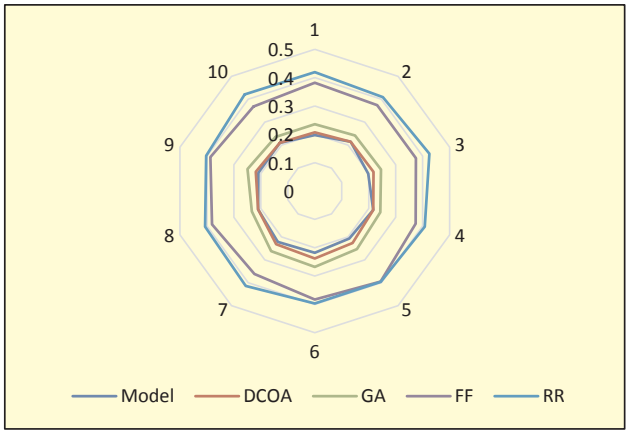
Problem ID	Migration cost					Penalty cost					RPD				
	RPD					Model (Lingo)	DCOA	GA	RR	FF	Model (Lingo)	DCOA	GA	RR	FF
LSP-03	–	0.036	0.139	0.438	0.502	–	0.211	0.239	0.375	0.412	–	0.071	0.214	0.904	1.091
LSP-04	–	0.116	0.222	0.343	0.522	–	0.198	0.221	0.382	0.419	–	0.005	0.122	0.939	1.127
LSP-05	–	0.000	0.121	0.454	0.526	–	0.217	0.241	0.360	0.422	–	0.102	0.224	0.827	1.142
LSP-06	–	0.060	0.174	0.347	0.574	–	0.227	0.254	0.366	0.405	–	0.152	0.288	0.858	1.056
LSP-07	–	0.040	0.159	0.371	0.570	–	0.208	0.253	0.374	0.417	–	0.056	0.282	0.898	1.117
LSP-08	–	0.072	0.175	0.394	0.510	–	0.208	0.236	0.391	0.399	–	0.056	0.199	0.985	1.025
LSP-09	–	0.008	0.123	0.390	0.582	–	0.220	0.248	0.366	0.412	–	0.117	0.261	0.858	1.091
LSP-10	–	0.024	0.122	0.382	0.574	–	0.202	0.243	0.378	0.419	–	0.025	0.234	0.919	1.127
LSP-11	–	0.080	0.181	0.323	0.578	–	0.224	0.252	0.372	0.403	–	0.137	0.277	0.889	1.046
LSP-12	–	0.096	0.200	0.339	0.546	–	0.233	0.261	0.372	0.394	–	0.183	0.323	0.888	1.000
LSP-13	–	0.068	0.186	0.406	0.502	–	0.225	0.256	0.362	0.412	–	0.142	0.298	0.838	1.091
LSP-14	–	0.016	0.097	0.418	0.546	–	0.197	0.223	0.371	0.431	–	0.000	0.130	0.883	1.188
LSP-15	–	0.064	0.145	0.363	0.554	–	0.210	0.233	0.387	0.402	–	0.066	0.181	0.964	1.041
LSP-16	–	0.100	0.198	0.323	0.558	–	0.204	0.230	0.378	0.416	–	0.036	0.168	0.919	1.112
LSP-17	–	0.008	0.114	0.418	0.550	–	0.214	0.247	0.368	0.416	–	0.086	0.252	0.868	1.112
LSP-18	–	0.049	0.152	0.371	0.558	–	0.214	0.240	0.393	0.392	–	0.086	0.220	0.995	0.990
LSP-19	–	0.049	0.135	0.371	0.560	–	0.206	0.230	0.379	0.413	–	0.046	0.166	0.924	1.096
LSP-20	–	0.048	0.165	0.370	0.561	–	0.199	0.221	0.369	0.431	–	0.010	0.123	0.873	1.188



(a). Energy costs



(b). Task migration costs

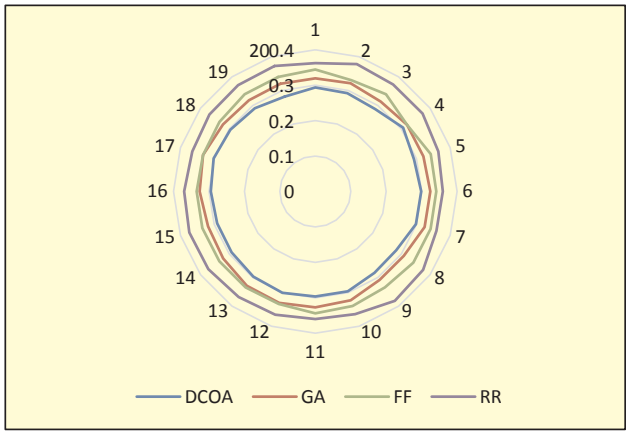


(b). Penalty costs

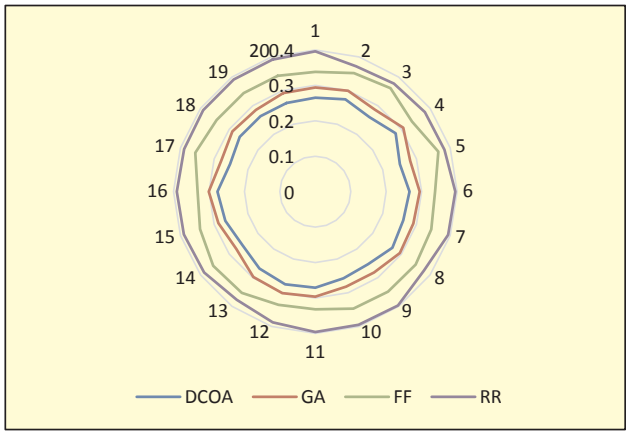
Fig. 9. Normalized costs for small problems.

Dual-core CPU with 2.50 GHz and 2 GB RAM, and Win 7 as the operating system.

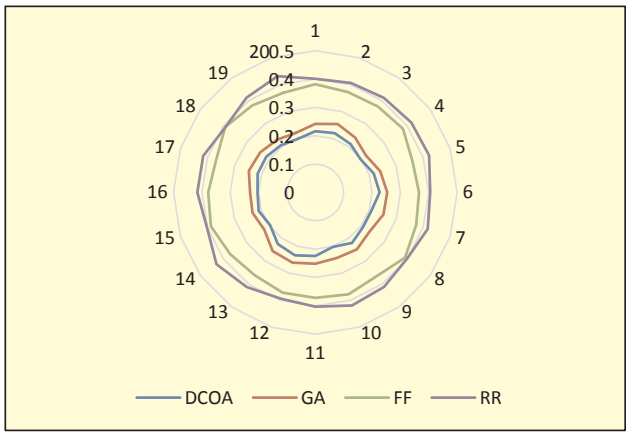
To compare our proposed mathematical model and the DCOA with other existing approaches, we select three popular algorithms for resource allocation in the literature, i.e., first fit (FF), round robin (RR) and a well-known metaheuristic genetic algorithm (GA) tuned as in our previous work (Shahdi-Pashaki, Teymourian, & Tavakkoli-Moghaddam, 2016). The results of all the methods are given in Table 15, where the first column presents the problem ID, including SSP and LSP for small size and large size problems, respectively. In this table, columns 2–6 show the energy costs, columns 12–16 the migration costs, and columns



(a). Energy costs



(b). Task migration costs



(c). Penalty costs

Fig. 10. Normalized costs for large problems.

22–26 the penalty costs.

In addition, Fig. 9 illustrates the normalized results (i.e., energy cost, migration cost, and penalty cost) obtained from all the methods for small problems. As shown in Fig. 9(a)–(c), the mathematical model clearly outperforms the FF and RR ones, while its solutions are slightly better than the DCOA and GA. Therefore, we conclude that the DCOA can produce good optimal solutions and perform satisfactorily in large problems.

Table 16
The ANOVA results.

Objectives	Source of variation	SS	df	MS	F	P-value	F crit
Energy costs	Between Groups	0.752	3	0.2505	513.05	2.5E-50	2.725
	Within Groups	0.037	76	0.0005			
	Total	0.789	79				
Migration costs	Between Groups	2.930	3	0.9767	883.61	5.7E-59	2.725
	Within Groups	0.084	76	0.0011			
	Total	3.014	79				
Penalty costs	Between Groups	14.765	3	4.9216	1708.31	1.3E-69	2.725
	Within Groups	0.219	76	0.0029			
	Total	14.984	79				

Fig. 10 shows the normalized results for DCOA, GA, FF, and RR in large problems. The results confirm that the DCOA performs better than FF, RR and GA for all the costs being considered.

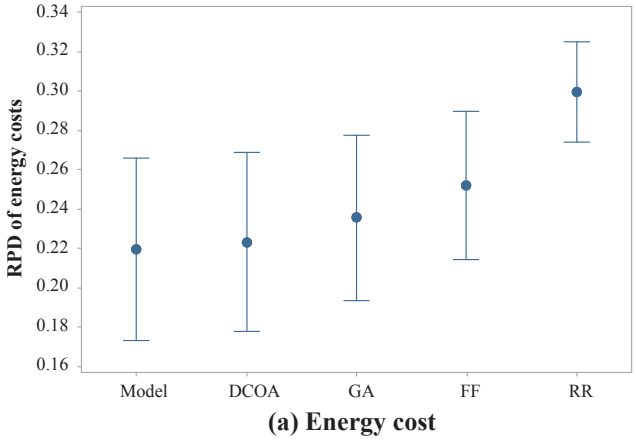
To validate these conclusions, we apply the analysis of variance (ANOVA) test to the results obtained from both sets of problems. The average Relative Percentage Deviation (RPD) is computed for each problem using Eq. (19) and presented in Table 15 (columns 7–11, 17–21, and 22–26)

$$\text{RPD} = \left(\frac{Sol_{Alg} - Sol_{Best}}{Sol_{Best}} \right) \tag{19}$$

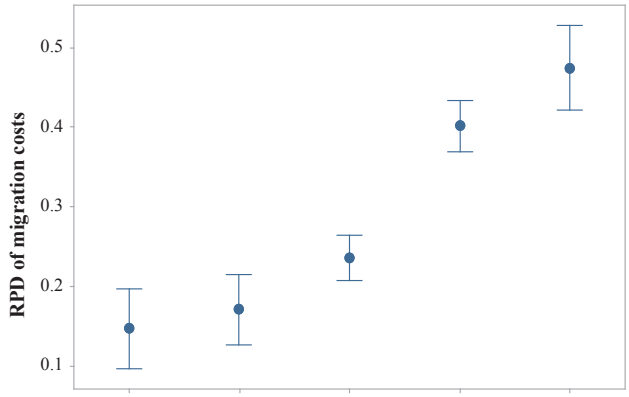
where Sol_{Best} is the best-known solution. According to the ANOVA test presented in Table 16, the null hypothesis is rejected because $F > F_{crit}$ (513.03 > 2.725 for energy costs, 883.61 > 2.725 for migration costs, 1708.31 > 2.725 for penalty costs), and the p-value < .5 (2.5E-50, 5.7E-59, and 1.3E-69 for energy costs, migration cost, and penalty costs, respectively). Hence, we apply a Tukey test to check each pair of means.

The means plot for all the methods with honestly significant difference (HSD) Tukey intervals at the 95% confidence level are presented in Figs. 11 and 12. In Fig. 11(a)–(c) we can see how the proposed model and the DCOA are not statistically different (their confidence intervals overlap) in small problems. However, the difference between their solutions and the FF and RR ones is statistically significant (their confidence intervals do not overlap), a tendency observed also for the GA when focusing on penalty costs. When considering large problems, the means plot with HSD Tukey intervals at the 95% confidence level for the DCOA, GA, FF and RR methods are shown in Fig. 12(a)–(c). The RPD values of the DCOA are statistically different and better on average than those of FF, RR and GA. Thus, the DCOA provides a much better solution on average than the FF, RR and GA methods.

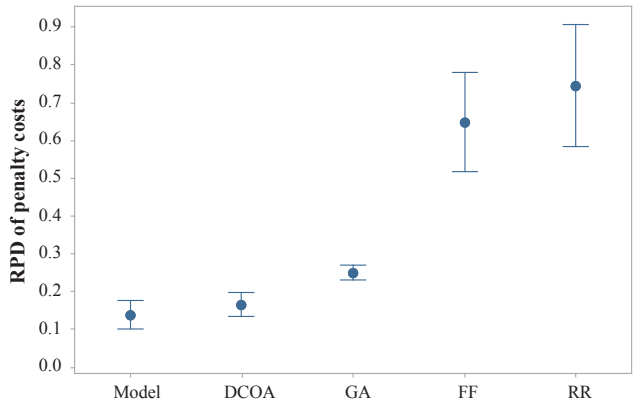
Finally, we have also compared the complexity of the DCOA, GA, FF and RR methods in terms of running time by generating 40 random problems. As shown in Fig. 13, though the DCOA and GA produce higher quality solutions, they take longer to find a solution compared to RR and FF. We also found that there is a significant difference between simple algorithms (FF and RR, with 16 s) and Meta heuristics ones (DCOA and GA, with 150–190 s) for small and medium size problems. For large problems and real instances the differences are slightly lower



(a) Energy cost

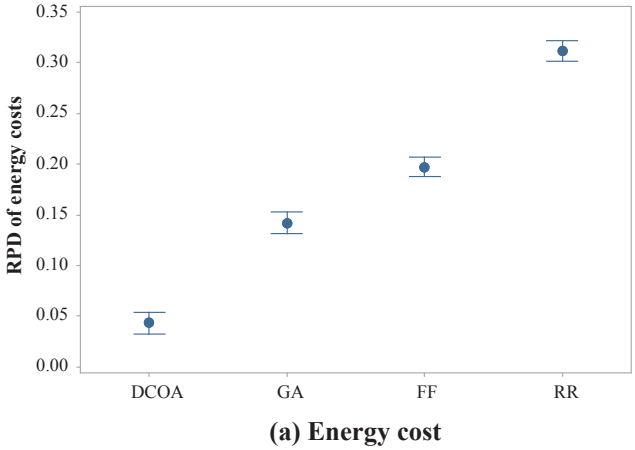


(b) Task migration cost

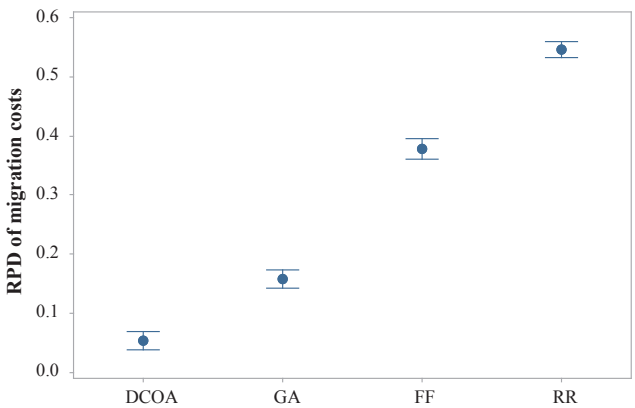


(c) Penalty cost

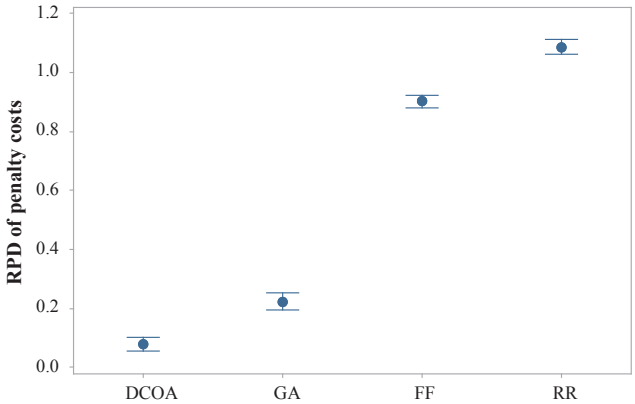
Fig. 11. Means plot and Tukey HSD intervals at the 95% confidence level for small instances.



(a) Energy cost



(b) Task migration cost



(c) Penalty cost

Fig. 12. Means plot and Tukey HSD intervals at the 95% confidence level for large instances.

(210 s for DCOA and FF, and 131 s for DCOA and RR). In contrast, the GA offers a solution with an even larger and increasing difference (410 s for GA and FF, and 330 s for GA and RR). Clearly, the DCOA consistently outperforms GA as the size of the problem considered increases. Therefore, we recommend using simple algorithms (i.e., FF and RR) for small problems if time is of the essence. However, we suggest using complex algorithms (i.e., DCOA and GA) for large problems when the solution quality is essential.

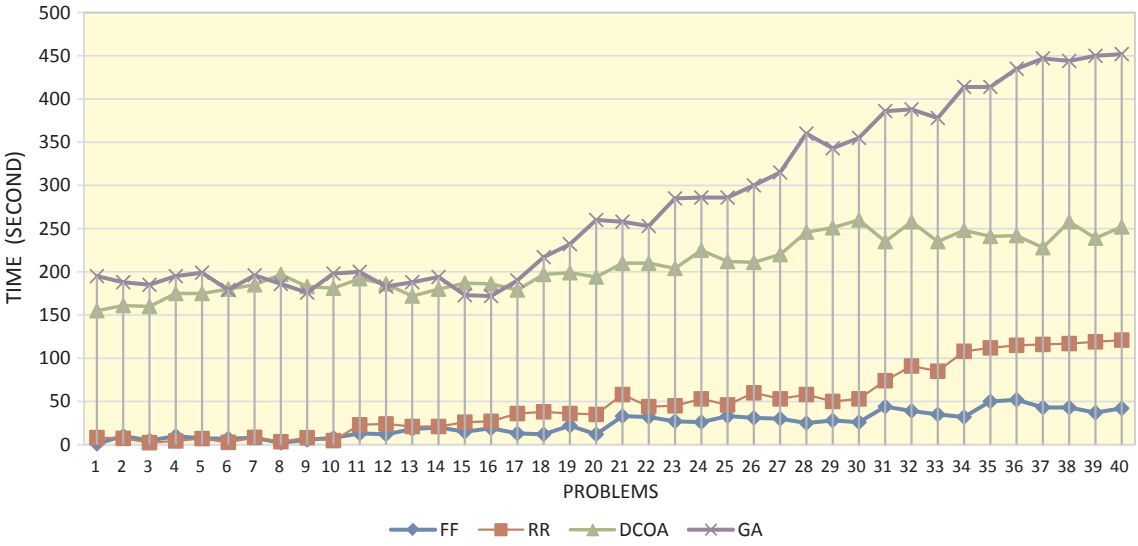


Fig. 13. Running time comparison of DCOA, GA, FF, and RR.

6. Conclusion

In this paper, we have designed a mathematical model using the main features of GT to handle consolidation problems, i.e., server consolidation, VM consolidation, and task consolidation. The objectives of the proposed model consist of controlling several operational costs in CC, namely, energy consumption costs incurred by the servers and VMs, penalty cost of unmet or rejected tasks and the costs which are imposed on systems because of task migration between servers. Moreover, we have defined a discrete version of the COA to handle large size problems (without an optimum solution), including specific adjustments such as a new grouping strategy based on the Jaccard similarity coefficient. To prove the validity and efficiency of the model, a numerical example was designed and run 12 separate times with different preferences. We have also compared our model and the DCOA with the GA, FF and RR methods through difference size test instances. The statistical analysis results show that our approaches outperform those of the GA, FF and RR methods.

Acknowledgement

The authors would like to thank the anonymous reviewers and the editor for their insightful comment and suggestions.

References

Ameryan, M., Akbarzadeh Totonchi, M. R., Seyyed Mahdavi, S. J. (2014). Clustering based on cuckoo optimization algorithm. In *Intelligent systems (ICIS), 2014 Iranian conference on* (pp. 1–6). IEEE.

Azarbad, M., Ebrahimzadeh, A., & Addeh, J. (2015). A new intelligent approach for recognition of digital satellite signals. *Journal of Signal Processing Systems, 79*(1), 1–14 (2013).

Belady, C. L. (2007). In the data center, power and cooling costs more than the IT equipment it supports. *Electronics Cooling, 13*(1), 24.

Bobroff, N., Kochut, A., & Beaty, K. (2007). Dynamic placement of virtual machines for managing sla violations. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on* (pp. 119–128). IEEE.

Burbidge, J. L. (1971). Production flow analysis. *Production Engineer, 50*(4.5), 139–152.

In R. Buyya, J. Broberg, & A.M. Goscinski (Eds.) (2010). *Cloud computing: principles and paradigms* (Vol. 87). John Wiley & Sons.

Cardosa, M., Korupolu, M. R., & Singh, A. (2009). Shares and utilities based power consolidation in virtualized server environments. In *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on* (pp. 327–334). IEEE.

Cardosa, M., Singh, A., Pucha, H., & Chandra, A. (2010). Exploiting spatio-temporal tradeoffs for energy efficient MapReduce in the cloud. *Dept. of CSE, Univ. of Minnesota, Tech. Rep, 10-008*.

Chan, F. T. S., Lau, K. W., Chan, L. Y., & Lo, V. H. Y. (2008). Cell formation problem with consideration of both intracellular and intercellular movements. *International Journal of Production Research, 46*(10), 2589–2620.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., & Doyle, R. P. (2001). Managing energy and server resources in hosting centers. In *ACM SIGOPS operating systems review* (Vol. 35, No. 5, pp. 103-116). ACM.

Cheng, C. H., Goh, C. H., & Lee, A. (1996). Solving the generalized machine assignment problem in group technology. *Journal of the Operational Research Society,* 794–802.

Deelman, E., Singh, G., Su, M. H., Blythe, J., Gil, Y., Kesselman, C., & Laity, A. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming, 13*(3), 219–237.

Elnozahy, E. M., Kistler, M., & Rajamony, R. (2002). Energy-efficient server clusters. In *Power-aware computer systems* (pp. 179–197). Berlin Heidelberg: Springer.

Erdem, M. B., Kiraz, A., Eski, H., Çiftçi, Ö., & Kubat, C. (2016). A conceptual framework for cloud-based integration of virtual laboratories as a multi-agent system approach. *Computers & Industrial Engineering, 102*, 452–457.

Ergu, D., Kou, G., Peng, Y., Shi, Y., & Shi, Y. (2013). The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. *The Journal of Supercomputing, 64*(3), 835–848.

Ferreto, T. C., Netto, M. A., Calheiros, R. N., & De Rose, C. A. (2011). Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems, 27*(8), 1027–1034.

Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences, 79*(8), 1230–1242.

Goudarzi, H., Ghasemazar, M., & Pedram, M. (2012). Sla-based optimization of power and migration cost in cloud computing. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on* (pp. 172–179). IEEE.

Hanoun, S., Nahavandi, S., Creighton, D., & Kull, H. (2012). Solving a multiobjective job shop scheduling problem using Pareto Archived Cuckoo Search. *In IEEE international conference on emerging technologies and factory automation, ETFA.* Art. no. 6489617.

Hsu, C. H., Slagter, K. D., Chen, S. C., & Chung, Y. C. (2014). Optimizing energy consumption with task consolidation in clouds. *Information Sciences, 258*, 452–462.

Hu, J., Gu, J., Sun, G., & Zhao, T. (2010). A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on* (pp. 89–96). IEEE.

Kaydani, H., & Mohebbi, A. (2013). A comparison study of using optimization algorithms and artificial neural networks for predicting permeability. *Journal of Petroleum Science and Engineering, 112*, 17–23.

Komaki, M., Sheikh, S., Teymourian, E., & Malakooti, B. (2015). Cuckoo search algorithm for hybrid flow shop scheduling problem with multi-layer assembly operations. In *Proceedings of the 2015 International Conference on Operations Excellence and Service Engineering Orlando, Florida, USA, September 10–11* (pp. 614–623). IEOM Society.

Komaki, G. M., Teymourian, E., Kayvanfar, V., & Booyavi, Z. (2017). Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. *Computers & Industrial Engineering, 105*, 158–173.

Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., & Jiang, G. (2009). Power and performance management of virtualized computing environments via look ahead control. *Cluster computing, 12*(1), 1–15.

Lee, Y. C., & Zomaya, A. Y. (2009). Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on* (pp. 92–99). IEEE.

Lee, Y. C., & Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing, 60*(2), 268–280.

Lei, D., & Wu, Z. (2005). Tabu search approach based on a similarity coefficient for cell formation in generalized group technology. *International Journal of Production Research, 43*(19), 4035–4047.

Li, L. (2009). An optimistic differentiated service job scheduling system for cloud computing service users and providers. In *Multimedia and Ubiquitous Engineering, 2009.*

MUE'09. Third International Conference on (pp. 295–299). IEEE.
Li, C., Liu, Y., & Luo, Y. (2017). Multimedia cloud content distribution based on interest discovery and integrated utility of user. *Computers & Industrial Engineering, 109*, 1–14.
Li, X., & Yin, M. (2013). A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. *International Journal of Production Research, 51*(16), 4732–4754.
Liu, K., Jin, H., Chen, J., Liu, X., Yuan, D., & Yang, Y. (2010a). A compromised-time-cost scheduling algorithm in SwinDeW-C for instance-intensive cost-constrained workflows on cloud computing platform. *International Journal of High Performance Computing Applications*.
Liu, C., & Ye, C. (2013). Cuckoo search algorithm for the problem of permutation flow shop scheduling, Shanghai Ligong Daxue Xuebao. *Journal of University of Shanghai for Science and Technology, 35*(1), 17–20.
Liu, C., Yin, Y., Yasuda, K., & Lian, J. (2010b). A heuristic algorithm for cell formation problems with consideration of multiple production factors. *The International Journal of Advanced Manufacturing Technology, 46*(9–12), 1201–1213.
Luo, J. P., Li, X., & Chen, M. R. (2014). Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers. *Expert Systems with Applications, 41*(13), 5804–5816.
Mahdavi, I., Baher, N. T., & Teymourian, E. (2010). A new cell formation problem with the consideration of multifunctional machines and in-route machines dissimilarity-A two phase solution approach. In *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on* (pp. 475–479). IEEE.
Mahdavi, I., & Mahadevan, B. (2008). CLASS: An algorithm for cellular manufacturing system and layout design using sequence data. *Robotics and Computer-Integrated Manufacturing, 24*(3), 488–497.
Mahdavi, I., Teymourian, E., Baher, N. T., & Kayvanfar, V. (2013). An integrated model for solving cell formation and cell layout problem simultaneously considering new situations. *Journal of Manufacturing Systems, 32*(4), 655–663.
Mahesh, O., & Srinivasan, G. (2002). Incremental cell formation considering alternative machines. *International Journal of Production Research, 40*(14), 3291–3310.
Malet, B., & Pietzuch, P. (2010). Resource allocation across multiple cloud data centres. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science* (p. 5). ACM.
Marichelvam, M. K. (2012). An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems. *International Journal of Bio-Inspired Computation, 4*(4), 200–205.
Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Applied Soft Computing Journal, 19*, 93–101.
Masson, R., Vidal, T., Michallet, J., Penna, P. H. V., Petrucci, V., Subramanian, A., & Dubedout, H. (2013). An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications, 40*(13), 5266–5275.
Mitrovanov, S. P. The Scientific Principles of Group Technology, 1966. English transl.
Nair, G. J., & Narendran, T. T. (1998). CASE: A clustering algorithm for cell formation with sequence data. *International Journal of Production Research, 36*(1), 157–180.
Nathuji, R., & Schwan, K. (2007). Virtual Power: coordinated power management in virtualized enterprise systems. In *ACM SIGOPS Operating Systems Review (Vol. 41, No. 6, pp. 265–278)*. ACM.
Nie, H., Liu, B., Xie, P., Liu, Z., & Yang, H. (2014). A cuckoo search algorithm for scheduling multi skilled workforce. *Journal of Networks, 9*(5), 1346–1353.
Ouaarab, A., Ahiod, B., & Yang, X.-S. (2014). Improved and discrete cuckoo search for solving the travelling salesman problem. *Studies in Computational Intelligence, 516*, 63–84.
Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on* (pp. 400–407). IEEE.
Pinheiro, E., Bianchini, R., Carrera, E. V., & Heath, T. (2001). Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on compilers and operating systems for low power* (Vol. 180, pp. 182–195).
Rajabioun, R. (2011). Cuckoo optimization algorithm. *Applied Soft Computing, 11*(8), 5508–5518.
Ramakrishnan, L., & Gannon, D. (2008). A survey of distributed workflow characteristics and resource requirements. *Indiana University,* 1–23.
Selvarani, S., Sadhasivam, G. S. (2010). Improved cost-based algorithm for task scheduling in cloud computing. In *Computational intelligence and computing research (ICCIC), 2010 ieee international conference on* (pp. 1–5). IEEE.
Seo, E., Jeong, J., Park, S., & Lee, J. (2008). Energy efficient scheduling of real-time tasks on multicore processors. *Parallel and Distributed Systems, IEEE Transactions on, 19*(11), 1540–1552.
Shahdi-Pashaki, S., Teymourian, E., Booyavi, Z., & Alizaheh, K. M. (2015). Resource Management for Workflows in Cloud Computing Environment based on Group Technology Approach. In *Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management, IEOM 2015*, Dubai, United Arab Emirates (UAE), March 3–5, 2015.
Shahdi-Pashaki, S., Teymourian, E., Kayvanfar, V., Komaki, G. M., & Sajadi, A. (2015b). Group technology-based model and cuckoo optimization algorithm for resource allocation in cloud computing. *IFAC-PapersOnLine, 48*(3), 1140–1145.
Shahdi-Pashaki, S., Teymourian, E., & Tavakkoli-Moghaddam, R. (2016). New approach based on group technology for the consolidation problem in cloud computing-mathematical model and genetic algorithm. *Computational and Applied Mathematics, 5*(14), 1–26.
Shahvari, O., & Logendran, R. (2017). A bi-objective batch processing problem with dual-resources on unrelated-parallel machines. *Applied Soft Computing, 61*, 174–192.
Shahvaria, O., Salmasib, N., & Logendranc, R. (2009). A meta-heuristic algorithm for flexible flow shop sequence dependent group scheduling problem. *In Proceedings of the 2009 International conference on value chain sustainability (ICOVACS 2009), Kentucky*.
Solimanpur, M., Vrat, P., & Shankar, R. (2004). A multi-objective genetic algorithm approach to the design of cellular manufacturing systems. *International Journal of Production Research, 42*(7), 1419–1441.
Speitkamp, B., & Bichler, M. (2010). A mathematical programming approach for server consolidation problems in virtualized data centers. *Services Computing, IEEE Transactions on, 3*(4), 266–278.
Srikantaiah, S., Kansal, A., & Zhao, F. (2008). Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems* (Vol. 10, pp. 1–5).
Tavakkoli-Moghaddam, R., Safaei, N., & Sassani, F. (2008). A new solution for a dynamic cell formation problem with alternative routing and machine costs using simulated annealing. *Journal of the Operational Research Society, 59*(4), 443–454.
Teoh, C. K., Wibowo, A., & Ngadiman, M. S. (2014). An adapted cuckoo optimization algorithm and genetic algorithm approach to the university course timetabling problem. *International Journal of Computational Intelligence and Applications, 13*(01).
Verma, A., Ahuja, P., & Neogi, A. (2008). pMapper: power and migration cost aware application placement in virtualized systems. In *Middleware 2008* (pp. 243–264). Berlin Heidelberg: Springer.
Wei, G., Vasilakos, A. V., Zheng, Y., & Xiong, N. (2010). A game-theoretic method of fair resource allocation for cloud computing services. *The journal of supercomputing, 54*(2), 252–269.
Wemmerlöv, U., & Hyer, N. L. (1989). Cellular manufacturing in the US industry: a survey of users. *The International Journal of Production Research, 27*(9), 1511–1530.
Xu, B., Zhao, C., Hu, E., & Hu, B. (2011). Job scheduling algorithm based on Berger model in cloud environment. *Advances in Engineering Software, 42*(7), 419–425.
Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation, 1*(4), 330–343.
Zhang, F., Yan, Y., Wu, W., & Luo, L. (2013). A heuristics approach for reducing power consumption of cloud data center. In Green Computing and Communications (GreenCom), In *2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing* (pp. 246–253). IEEE.
Zhang, W., Liu, J., Liu, C., Zheng, Q., & Zhang, W. (2015). Workload modeling for virtual machine-hosted application. *Expert Systems with Applications, 42*(4), 1835–1844.
Zhao, C., & Wu, Z. (2000). A genetic algorithm for manufacturing cell formation with multiple routes and multiple objectives. *International Journal of Production Research, 38*(2), 385–395.
Zhong, H., Tao, K., & Zhang, X. (2010). An approach to optimized resource scheduling algorithm for open-source cloud systems. In *China Grid Conference (China Grid), 2010 Fifth Annual* (pp. 124–129), IEEE.