Future Generation Computer Systems 69 (2017) 1–10

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Multi-Capacity Combinatorial Ordering GA in Application to Cloud resources allocation and efficient virtual machines consolidation

Huda Hallawi *, Jörn Mehnen, Hongmei He

Manufacturing Department, Cranfield University, Bedford MK43 0AL, UK

HIGHLIGHTS

- Cloud Resources Allocation as Vector Bin Packing under Multiple Constraints.
- Propose a novel bio-inspired approach based on Ordering Genetic Algorithm.
- Develop two algorithms (COFFGA and CONFGA) with the proposed approach.
- Evaluate the performance and decision robustness of proposed algorithms.
- COFFGA is superior to CONFGA and existing algorithms.

ARTICLE INFO

Article history:
Received 24 February 2016
Received in revised form
6 October 2016
Accepted 26 October 2016
Available online 2 November 2016

Keywords:
Cloud resources allocation
Cloud resources provisioning
Virtual machines consolidation
Vector bin packing
Genetic algorithm

ABSTRACT

This paper describes a novel approach making use of genetic algorithms to find optimal solutions for multi-dimensional vector bin packing problems with the goal to improve cloud resource allocation and Virtual Machines (VMs) consolidation. Two algorithms, namely Combinatorial Ordering First-Fit Genetic Algorithm (COFFGA) and Combinatorial Ordering Next Fit Genetic Algorithm (CONFGA) have been developed for that and combined. The proposed hybrid algorithm targets to minimise the total number of running servers and resources wastage per server. The solutions obtained by the new algorithms are compared with latest solutions from literature. The results show that the proposed algorithm COFFGA outperforms other previous multi-dimension vector bin packing heuristics such as Permutation Pack (PP), First Fit (FF) and First Fit Decreasing (FFD) by 4%, 34%, and 39%, respectively. It also achieved better performance than the existing genetic algorithm for multi-capacity resources virtual machine consolidation (RGGA) in terms of performance and robustness. A thorough explanation for the improved performance of the newly proposed algorithm is given.

1. Introduction

Cloud computing has emerged as a computing model aiming at providing computing resources as a service according to the pay-per-use paradigm [1,2]. Cloud computing is based on full virtualisation technology where a single physical machine is able to host several virtual machines which in turn are completely isolated. The net effect of having shared resource usage is having fewer physical servers with higher utilisation per server which will effectively minimise the hardware costs and the operational expenses [3,4]. However, the flexibility enabled by virtualisation has produced new computational challenges of managing a shared pool of resources over the competing instances of applications or jobs.

Resource allocation is the process of mapping the available resources to competing jobs based on the individual job requirements [5]. Computing resources must be well-managed to prevent overloading and waste of bandwidth, processing unit, memory, etc. This waste relates directly to significant financial loss for large Cloud service providers with regards to energy, operational cost as well as dissatisfaction of the Cloud service user [6,7]. Resources allocation systems control how multiple Virtual Machines (VM) share the underlying Physical Machines (PM). Fast and efficient resource allocation algorithms can help to save energy and cost while increasing customer satisfaction.

Resource allocation is typically performed in two stages as shown in Fig. 1. The first stage is the jobs assignment to the Virtual Machines: applications or jobs (both terms are used synonymously in the context of this paper) are executed on VMs. Each application has its own requirements of compute power, disk space and RAM space, communication bandwidth, priority, etc. (see [7]). Any VM must meet these requirements when resources are allocated.

* Corresponding author.
E-mail address: h.f.hallawi@cranfield.ac.uk (H. Hallawi).

http://dx.doi.org/10.1016/j.future.2016.10.025
0167-739X/

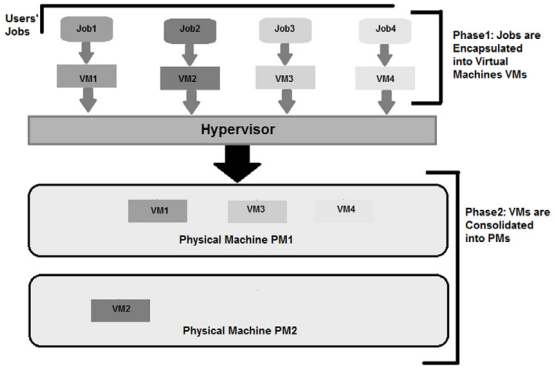

Fig. 1. Resource allocation phases in Cloud computing.

The second stage is the VM assignment to servers. One or several virtual machines can be lodged on the same server. The host is responsible for providing the computing resources to the VM. The server is typically identical to the physical machine with specified capacities. This phase is done by using a hypervisor running on the top of the physical hardware. The hypervisor enables to create virtual environment to operate virtual machines [7,8]. Xen and KVM are well known examples of hypervisors.

Primarily, Cloud resource allocation handles queries and assigns a number of independent services to physical machines. Mapping services with complex computing to physical machines with specified capacities can be transferred to a classic vector (or multi-capacity) bin packing problem (VBPP) [9]. Each dimension of the problem is corresponding to a type of resources such as CPU, RAM memory, disk space and bandwidth.

In order to optimally allocate VMs to PMs, several sophisticated techniques have been developed for different purposes such as obtaining good resources utilisation, minimising energy consumption or achieving good load balancing (see [10,11,5,6]).

This paper is dedicated to introducing a new hybrid optimisation algorithm which is aiming to calculate optimal solutions to the VBPP and thus to optimise virtual machine allocation and consolidation. Combinatorial Genetic Algorithms (GA) [12] are used to find the best VMs order, while the packing decision is handled by using an approximation heuristic in a way that minimises the number of non-ideal physical machines and total resources wastage in each machine. Two genetic algorithms have been employed. The first one uses Next Fit as a heuristic packing governor, while the second one uses First Fit. The aim is to consolidate as many VMs as possible and to reduce the resources wastage in each physical machine. This has been formulated into a novel fitness function.

The rest of this paper is outlined as follows: Section 2 discusses the preliminary research in Cloud resources allocation and vector bin packing problem solving then its application to Cloud resources allocation and consolidation. Section 3 discusses the approach followed in this paper towards a new problem formulation. Section 4 explains the design of the new hybrid algorithm and all associative configuration operations. Section 5 deals with data configuration and gives an outline to the results and comparisons; in Section 6 the performance of the new algorithms is presented and compared with existing algorithms from literature. The paper also aims at giving explanations of how the reformulation of the problem and the corresponding new algorithm affected this improvement in performance. Finally, Section 7 outlines the conclusions and future work.

2. Previous work

2.1. Overview of cloud resources allocation and VMs consolidation

Resources allocation is a central theme in Cloud computing, since it controls the way that resources and services are delivered to the end entities, at the same time it maintains an efficient utilisation of computing resources [1,3,4]. Cloud providers intend to optimise the usage of underlying resources, through planned allocation of VMs in servers (hosts). Cloud service providers are also keen to maintain a positive client experience. Conserving a good Quality of Service level in Cloud Resources Allocation was considered by Lin J. et al. in [13]. They studied the phenomena VM interference that resulted from interference between the VMs on the same PMs and its effects on the services' degradation, especially the sensitive applications. The QoS-aware VM placement (QAVMP) problem is formulated as problem as an Integer Linear Programming (ILP) model. Then a polynomial-time heuristic algorithm was developed to solve the QAVMP problem.

Energy consumption is an active research field in Cloud resource management. Ramani and Bohara in [14] developed a new approach for minimise the overall energy consumption. This approach is based on defining a temperature threshold of the hosts in Cloud data centre. It reduces the consumption of the maximum resources and controls the processor temperature at the same time, thus succeeding to decrease the energy consumption in total. Kumara, and Raghunathan in [15] proposed an integral thermal and compute controlled heuristic approach to minimise the energy consumption in IaaS Cloud. Shu et al. in [16] enhanced the immune clonal selection algorithm to design Cloud and Grid schedulers with a goal of simultaneous optimisation of the energy utilisation. Efficient load balancing is one of the major challenges in Cloud resources allocation process. It controls assigning the load to different nodes to prevent overloaded or under-loaded nodes. Many algorithms were proposed to tackle this problem for example Biased Random Sampling in [17].

VMs consolidation is a special problem that is mutually exclusive to resources allocation in virtualized hosted platforms like Cloud. It targets to consolidate maximum number of VMs onto minimum number of PMs, thus improving resources utilisation [3, 18]. Francu M. in [19] employed the Markov Decision Process (MDP) to generate long term precise migration decision which aims to improve the profit by avoiding the wrong decisions which may have an adverse effect on the total profit.

2.2. Existing approaches for solving vector bin packing in cloud resources allocation and consolidation

Several researchers have studied VBPP (or d-capacity bin packing problem) and its connection with Cloud resources allocation [20–23]. A range of approximation algorithms have been developed to solve the problem for different objectives as described in [24–26]. Examples of that are Next Fit, First Fit, Best Fit, Dot Product and Permutation Pack. Dot Product is used by Microsoft's virtual machine manger [27]. Single capacity Next-Fit [20] deals with a list of items in a given order one at a time. It checks whether there is a space in a current bin for the current item; if there is a space it will allocate the current item into the current bin and then continues with the next item. If it does not fit, close that bin and open a new bin and allocate the given item. According to [28,29] the Next-Fit algorithm is the quickest algorithm compared to other heuristic algorithms as it requires $O(N)$ time. When Next-Fit is extended to work with multi-capacity bin packing, the main difference is that it will deal with the d-capacity requirements item instead of one requirement only [20].

The First-Fit Algorithm (FF) also treats the items in a given order; however, it will allocate the given item to the first fit non-empty bin. If all opened bins do not have enough space to fit the current item, a new bin will be opened and the item allocated (see [28,29]). The First-Fit removes the restriction of the Next-Fit algorithm as it allows the current item to be packed in any non-empty bin which can accommodate the item, however, it often

provides non-optimal solutions. It is also known as a fast heuristic requiring $O(N \log N)$ time, where n is the number of items to be packed [30]. First-Fit has been modified to work with multi-capacity vector bin packing for scheduling and virtual machine packing as described in [22,27,31].

Literature [24,28,30] describe the First Fit Decreasing as another heuristic algorithm for traditional bin packing based on ordering the given items in a decreasing order and then applying the FF algorithm to make the packing decision. Due to the multiple dimensions in vector bin packing, the items are ordered according to predefined criteria either by taking the sum of the item weights $V\ (i) = \sum_{r=1}^{d} Wr$, where $V(i)$ is the volume of the ith item and d is the number of resources; or, following [9,25,32], by using the dot product of the item weights $V\ (i) = W_{i,\ 1} * W_{i,\ 2} * \cdots * W_{i,\ d}$. The main drawback of this algorithm lies in the fact that it ignores the correlation across the dimensions which can lead to excessive waste in resources.

Permutation Pack (PP) is a dimension-aware vector bin packing algorithm which takes advantage of the complimentary requirements for different resources with the goal of minimising the resource wastage and the number of the required bins [10,20]. It aims to pack those items which need excessive resources in one dimension in the same set of bins and the other items with different needs in other sets of bins. In Cloud resource allocation, we have a number of VMs, each of which is a vector of different resources, e.g. CPU and memory. PP will pack those VM which need more CPU than memory in the same Physical Machine, while consolidating the other VMs which need more memory than CPU in the other set of bins. Choose Pack and Dot Product are other types of dimension-aware vector bin packing algorithms (see [10,20,27]).

Evolutionary Algorithms have been applied to VBPP. Examples are Genetic Algorithms (GA) and Ant Colony Optimisation [11]. Especially, GA have recently been used in Cloud resources allocation and scheduling by many researchers [11,33–41] for various objectives covering load balancing, cost minimisation and resources utilisation. Wilcox [35] and Jing Xu [39] applied and tested a special kind of Genetic Algorithm called GGA (Grouping Genetic Algorithm) to virtual machine bin packing. The algorithm was originally developed to deal with various grouping problems including BPP. However, according to [41,42] the algorithm suffers from performance problems due to the difficulty of crossover and mutation.

Wilcox has applied RGGA (Reordering Grouping Genetic Algorithm) to Cloud virtual machine consolidation and developed a new crossover operator that avoids infeasible solutions within multi-dimensional constraints [35]. However, the GGA permutation is still complex compared to a classic GA. The proposed algorithm as introduced in this paper is based on the rationale of the combinatorial ordering GA of Corcoran and Wainwright [43] and its concise permutation technique. It was developed for order based problems, but it is also supplemented by a range of user interface tools that offers a workbench for other genetic algorithm research. It comes with two GA: generational GA and steady state GA, and introduces a variety of genetic operators for selection, crossover and mutation. This GA was applied to solve different combinatorial optimisation problems such as traditional Bin Packing, travelling salesman problem and multiprocessors scheduling [12].

3. Problem modelling

In this research Cloud resources allocation (as described in Section 1) will be formulated as multi-capacity vector bin packing, where a number of jobs need to be severed by a number of servers in a Cloud data centre.

3.1. Definition of vector bin packing

The vector bin packing problem this paper concerns is called a d-capacity bin packing problem which is a special generalisation of the traditional bin packing problem [9,20,21]. Given a set B, of m identical bins, the capacity of a bin is represented by a d-capacity vector $\overrightarrow{C} = (C_1,\ C_2,\ C_3,\ C_i,\ \ldots,\ C_d)$ where C_i is the ith component capacity and $\sum_{i=1}^{d} C_i > 0, C_i \geq 0$.

Assume a set of n items, $L = \{X_1, X_2, X_3, \ldots, X_n\}$, where the items of L are required to be packed into as few bins as possible without exceeding the bin capacity. An item is also represented as a d-capacity vector $\overrightarrow{X_j} = (X_{j,\ 1},\ X_{j,\ 2},\ X_{j,\ 3},\ X_{j,\ k},\ \ldots, X_{j,\ d})$, where X_{jk} is the kth component requirement of the jth item $0 < X_{j,\ k} < C_j$.

A solution of the vector bin packing problem can be represented as $B = \{B_1, B_2, B_i, \ldots, B_m\}$, where m bins will be required to pack the n items where each bin accommodates S items. The packing decision for a bin can be represented as:

$B_i = \{X_{2,\ i}, X_{7,\ i},\ X_{5,\ i}, \ldots X_{s,\ i}\}$.

The packing decision is subject to the following constraints:-

$$(1)\ \forall X \quad \sum_{i=1}^{m} X_{j,\ i} = 1 \tag{1}$$

$$(2)\ \forall X \quad \sum_{r=1}^{d} 0 < X_{j,\ r} < C_{i,\ r} \tag{2}$$

$$(3)\ \forall B, \quad X\overrightarrow{Bi} + \overrightarrow{Xj} \leq \overrightarrow{C}. \tag{3}$$

The first constraint means that each item must be accommodated in a single bin; m is the number of bins used to pack the available items. The second constraint means that the item requirement in each dimension should not exceed the corresponding capacity of that bin. The third constraint shows that allocating a new item to a bin must not exceed the total capacity of that bin.

The fundamental problem of vector bin packing – and hence also its generalisation of d-capacity bin packing – is NP-hard (see [6, pp. 7], and [10, pp. 125]). This implies that for large real-life problems such as Cloud Resource Allocation deterministic algorithms quickly reach computational barriers.

3.2. Problem formulation

In order to formalise the problem, a number of assumptions have been made:

(1) Requirements of each job are predefined probably by the user.
(2) Each job will be encapsulated in a VM; we aim to work with Infrastructure as a Service (IaaS) where a Cloud user leases a slice of hardware in the Cloud data centre, which is completely isolated from the other users.
(3) All jobs need a fixed amount of CPU and memory for a specified amount of time, i.e. this paper is dealing with a static scheduling problem. Only CPU and memory requirements are considered. Considering running time is future research. Synthetic workloads have been used in this paper. They are described in more detail in Section 4. The workload is assumed to be static and no job leaves or changes the resource requests.
(4) The Cloud cluster comprises only homogeneous servers.
(5) Each VM should be mapped to just one server; it is assumed that all VMs have deterministic load equal to the encapsulated job load.

The mapping of the VMs aims to minimise the number of used servers and to reduce the resource wastage in each

Table 1
Parameters used in the problem formulation.

$W_{j,\,i}\ i \in [1, \ldots, M],\ j \in [1, \ldots, R]$	Total resources wastage in all servers.
$[SRCi, \ldots,\ SRMi]\ i \in [1, \ldots, M]$	The capacity vector of the ith server; CPU capacity, and memory capacity respectively.
$[C_{k,\,i},\ M_{k,\,i}],\ K \in [1, \ldots, A]\ i \in [1, \ldots, M]$	The vector of CPU and memory requirements for the kth VM packed in Server_i according to the packing decision.
$P_{j,\,i} \in [0,\ 1],\ i \in [1, \ldots, M],\ j \in [1, \ldots, N]$	Packing decision

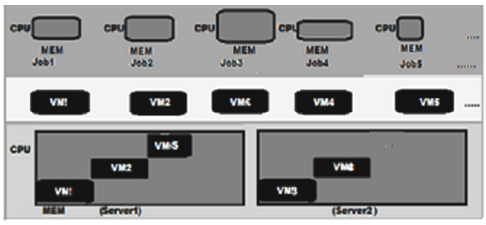

Fig. 2. Problem formulation of resource allocation seen as 2-D vector bin packing in two levels.

server. This is a challenge when one considers multi-dimensional requirements. Based on the above assumptions, the multi-capacity VMs placement in a Cloud data centre can be converted into a 2-D vector bin packing problem. On that basis two novel algorithms have been formulated namely CONFGA and COFFGA for solving the multi-capacity VM placement problem.

The number of the jobs in workloads determines the size of the vector. It equals to the number of items to be packed. Fig. 2 presents how resource allocation and VMs consolidation are handled as 2-D vector bin packing in two levels. The first level deals with mapping of the workload requirements to VMs with two different capacities, regarding CPU and memory, while the second level controls the VMs deployment over the available physical servers (PMs) under the constraints of unique packing per server and not exceeding the server capacities. The system under consideration comprises of N Jobs, correspondingly N VMs, M servers, R resources and A the number of VMs being packed in one server according to the resulted packing decision. The new algorithm targets to use as few servers as possible as described by formula (4) and to minimise the resources wastage in each server as formulated in formula (5). The goals and constraints can be formulated as follows:

$$\text{Goals}: \quad \min \sum_{i=1}^{M} Si \tag{4}$$

$$\min \sum_{i=1}^{M} \sum_{j=1}^{R} W_{j,\,i}, \quad R = [C_{k,\,i},\ M_{k,\,i}] \tag{5}$$

$$\text{with } W_i = \left(SRC_i - \sum_{K=0}^{a} C_{k,\,i}\right) + \left(SRM_i - \sum_{K=0}^{a} M_{k,\,i}\right), \quad k \in [0, \ldots, A]. \tag{6}$$

Constraints:

$$\sum_{i=1}^{m} P_{j,\,i} = 1 \quad j \in [1, \ldots, N] \tag{7}$$

$$\sum_{j=1}^{n} C_j P_{j,\,i} < SRCi \quad i \in [1, \ldots, M] \tag{8}$$

$$\sum_{j=1}^{n} M_j P_{j,\,i} < SRMi \quad i \in [1, \ldots, M]. \tag{9}$$

The parameters used in formulas 4 to 9 are stated in Table 1. Formula (5) indicates that consolidation decision resulted from the algorithm should minimise all resource wastage in all used servers. Formula (6) describes how to count the resource wastage. In the case considered in this paper, $R = 2$ represents CPU and Memory requirements; K is the index of the VMs packed in one server as results from our algorithm. Formula (7) indicates that each VM should be assigned to just one server. The formulas (8) and (9) specify that the VMs consolidation decision should not exceed server capacity in any dimension.

4. Novel multi-capacity combinatorial ordering GA

4.1. Algorithm design

The target is to develop an optimisation algorithm for vector bin packing. The approach followed in this paper is to combine a genetic algorithm with multi-dimension aware heuristic algorithms, such as multi-dimension First-Fit and multi-dimension Next-Fit heuristics, to optimise the packing solution. The GA is dealing with multi-dimension vector bin packing as a combinatorial optimisation problem. GA is an iterative procedure which borrows the concepts of the natural selection and survival of the fittest individual from the natural evolution. By choosing the suitable representation of the problem and emulating the biological selection and reproduction techniques, the GA can search a large problem space. It has been pointed out that the classical GA performs poorly in the application for bin packing problems [43,44]. Therefore, a variation of the classic GA had to be developed to deal with bin packing problems more efficiently. L. Corcoran and R. Wainwright have introduced a software package [12] which helps to develop genetic algorithm for different combinatorial optimisation problems. The proposed algorithm is based on the above mentioned software package.

Bin packing is an optimisation problem which aims to minimise the number of bins used to pack a group of items. In order to find a good solution for ordering the items and to find relations between item requirements in the vector bin packing problem, the genetic algorithm was combined with another heuristic algorithm. The idea is to use a cooperation of GA and traditional multi-dimension packing heuristics to produce packing decision under multiple constraints.

The GA evolves solutions to find a new packing solution. The goal is to find the best VM order to be packed by the heuristic objective function. The optimal solution should have the minimum number of servers and least resources wastage; the order (chromosome) is evaluated by using a fast multi-dimensional heuristic as an objective function. The heuristic is part of the objective function which guides the GA in its search for optimum packing solutions. Two new hybrid algorithms have been developed. These two algorithms follow the same basic procedure; however they apply different objective functions for getting their packing decision. The first algorithm is called Combinatorial Order Next Fit Genetic Algorithm (CONFGA). CONFGA uses Next-Fit as a packing solution governor and the second one uses First-Fit instead. Both FF and NF heuristics have been introduced in Section 2. The Combinatorial Order First-Fit Genetic Algorithm is called COFFGA. As Next-Fit heuristic does not provide backtrack packing. This will lead to better results with COFFGA than CONFGA as discussed later.

Multi-Capacity Combinatorial Ordering GA Procedure

- Initially, J=0 ; VMs=0; T=0; // number of jobs; number of VMs; termination condition
- Initially, OptSolution = NULL;
- To host a number of jobs with multi-capacity demands in hosted servers: Initialize a descriptive file that identifies number of servers, capacity of the servers' resources, number of jobs [services], and the requirements of each job [service].
- Call INITIALIZE VMs.
- Call FIND PLACEMENT USING COMBINTORIAL ORDERING GA.
- Call PLACE VMs.

```
INITIALIZE VMs
While J < number of jobs: // from descriptive file
 Begin
      J = J+1;
       Encapsulate a given job in a proper VM according to the
      job requirements.
      VMs= VMs+1;
 End
```

```
FIND PLACEMENT USING COMBINTORIAL ORDERING GA
     Initiate Termination Condition, Pool Size;
     ChromLength=VMs; // set the length of each Chromosome
     Pool = N; //create a pool of N chromosomes with random
                 VMs orders
     While T < Termination Condition do:  //run GA
          Begin
     N = Pool Size;
     T = T+1;
     While N≠0: //Find Fitness for all Pool Chromosomes
          Begin
               Fetch servers' capacities and VM's requirements;
               Pack the given VMs using Multi-Capacity vector
               bin packing heuristic, based on the VMs' order in
               a chromosome;
               Use packing function to convert the chrome VMs
               order to PackingSolution;
               Calculates the chromosome fitness using the
               fineness function in (4-C);
               N = N – 1;
               bestSolution = findBestOrder(Pool);
              End
     For i=1..PoolSize/2; // apply selection and crossover
               Begin
              [chrom1, chrom2] = Mini-Roulette; (4-D).
              [chd1, chd2] = Crossover(chrom1, chrom2); (4-E).
              Save(newPool, chd1, chd2);
       End
      Pool=newPool;
   End
```

```
PLACE VMs
If (OptSolution=NULL or fit(OptSolution)<fit(BestSolution)
          OptSolution = Best Solution;
         OptPacking = PackingSolution(Best Solution); //the
PackingSolution Associated with the Best Solution
 Apply OptPacking (VMs, Servers);
```

In the box above the proposed hybrid Multi-Capacity Combinatorial Ordering GA Procedure is shown in more detail. The procedure also describes the way of applying the proposed Combinatorial Ordering GA to Cloud resources allocation problem. It is divided into three main functions: INITIALIZE VMs, FIND PLACEMENT USING COMBINTORIAL ORDERING GA, and PLACE VMs. Each function is dedicated to specific purpose, the INITIALIZE VMs covers the initialisation of VMs according to the input workload. On the other hand, FIND PLACEMENT USING COMBINTORIAL ORDERING GA function produces the deploying decision of the pre-initialised VMs. It explains the design of the proposed Combinatorial Ordering GA that wraps a fast heuristic as an objective function to make the optimal packing decision. Finally, PLACE VMs function deploys the packing decision generated by the previous function onto Cloud servers.

4.2. Chromosome encoding

Vector bin packing is an optimisation ordering problem; in a GA the chromosome is encoded to represent a permutation of the order of VMs; this is similar to the encoding one would use in the well-known travelling salesperson problem [45,46]. In permutation encoding every chromosome is a string of numbers that represent a position in a sequence.

The length of each chromosome is equal to the number N of the VMs to be packed at any time. The chromosomes can be presented as:

0	1	2	3	4	…	N-1
3	18	7	15	8	…….	20

Chromosome 1

0	1	2	3	4	……	N-1
9	5	20	3	8	…….	20

Chromosome 2

The chromosome is encoded as a one dimension array of N integers. The chromosome gene is an index to the VM attributes as shown below:

0		1		2		…		N-1	
7		8		11		….		6	
↕		↕		↕		↕		↕	
VM(7)		VM(8)		VM(11)				VM(6)	
CPU	MEM	CPU	MEM	CPU	MEM	.	.	CPU	MEM

Two VM attributes (CPU and Memory) have been encapsulated in the algorithm. More VM attributes could be investigated in future work. Using integer encoding in one dimension facilitates crossover and permutation operations.

4.3. Objective and fitness function

Generally, the objective function is incorporated inside the fitness function of a GA. The objective function is a vital component also of the proposed algorithm. In the proposed new approach, the heuristic packing algorithm is "encapsulated" in the fitness function. In the following, this new concept will be explained in more detail.

Each packing solution will be associated with a chromosome. Therefore, the performance of an individual can be evaluated based on the packing solution. The objective function receives the pre-described chromosome. The objective function will find chromosome packing decision using the wrapped heuristic (NF or FF) and the input VMs requirements using the following operations:

- Match the VMs in the given chromosome order with their actual VMs requirements. This step changes the chromosome from a one dimension array of integers into an array of structures. Any array element comprises all required multi-dimensional attributes associated with a VM. This operation feeds the packing heuristic in the following operation with all required inputs.
- Apply the packing function which comprises of the chosen multi-dimensional heuristic (FF or NF) to find the packing order under the capacities constraints. The packing solution is represented as an array of two elements. The first element represents the number of VMs to be packed while the second element marks the index of a host server. The corresponding array encoding is shown below:

0	1	2	3	4	…	N-1
(20, 1)	(4,2)	(6,1)	(11, 3)	(9, 1)	…	(VM_j, $Server_i$)

- Determine the number of servers required to host the given VM packing order (chromosome) and the total resources wastage in each server.

The objective function also provides the parameters to feed the fitness function such as the number of servers for the associated packing decision and the resources wastage. The output of the algorithm is the packing solution, associated to the best chromosome (individual). The packing solution describes which VM will go to which server.

Previous genetic algorithms in Cloud resource allocation and vector bin packing used also heuristics e.g. for initialising the pool of individuals. However, these algorithms were suffering from the extra difficulties caused by the inadequate genome encoding during the crossover and the permutation operations. The proposed approach is different. In this case, the heuristic is used as integral part of the objective function. The objective function uses a permutation as an input (chromosome), which greatly reduces the complexity of the encoding and crossover.

In Genetic Algorithms, a fitness function is used to quantify the quality of the individuals in the population, thus directly reflecting the performance of the individuals. Each chromosome bears its individual fitness value. The individuals are selected for crossover depending on their fitness values. Therefore, the fitness function is the key driver within a Genetic Algorithm. In this paper the bi-objective goal of the proposed algorithms is to minimise, both, the total number of servers and the normalised capacity of the servers. This is formulated in Eq. (10):

$$f(O) = TS * R_{tot} \tag{10}$$

$$TS = \sum_{i=1}^{s} S_i \tag{11}$$

$$R_{tot} = R_{cpu} + R_{MEM} \tag{12}$$

$$R_{CPU} = \sum_{1=1}^{s} \frac{SRCi - \sum_{K=0}^{a} C_{k,\,i}}{SRCi} \tag{13}$$

$$R_{MEM} = \sum_{1=1}^{s} \frac{SRMi - \sum_{K=0}^{a} M_{k,\,i}}{SRMi} \tag{14}$$

where $f(O)$ is the fitness of the given chromosome order, TS is the total number of servers in the given packing decision, S is the number of servers needed to host the VMs of the given order, R_{tot} is total normalised residual capacity in the used servers and R_{cpu} normalised CPU residual capacity; the total wastage is reduced by minimising the residual in each server as presented in [20,21]. Normalising the total wastage helps to reach the right convergence, as it assigns each chromosome a unique fitness.

4.4. Selection strategy

Selection is a mechanism that favours well-fitted individuals and rejects the others. It chooses individuals of a current generation for producing offspring. This mechanism follows Darwin's principle of "Survival of the fittest". Different selection strategies have been developed for GA, such as roulette, rank biased or uniform random. The proposed algorithm adopted mini-selection roulette strategy [43]. The mini-selection roulette is similar to the classic roulette selection where each individual is assigned a slice of a circular roulette wheel where the size of the slice of the roulette wheel is proportional to the individual's fitness $F(i)$. However, the exception in the mini-selection roulette wheel strategy is that chromosomes are assigned a fitness $\acute{F}(i)$ which is reciprocal to $F(i)$. $P(i)$ is the probability that the chromosome is selected and that is proportional to $\acute{F}(i)$. This means that the smaller the $F(i)$ value the larger the value of $P(i)$. $P(i)$ is formalised in Eq. (15):

$$P(i) = \frac{\acute{F}(i)}{\sum_{j=1}^{n} F(j)}, \tag{15}$$

$$\acute{F}(i) = \frac{1}{F(i)}. \tag{16}$$

4.5. Crossover operation and mutation

Generally, crossover in genetic algorithms is the process of producing new individuals by substituting and reforming genes of two or several subsequently selected parental chromosomes. Crossover is a particularly important operator in genetic algorithms. Different types of chromosomes can have different crossover operations. Since a chromosome variation can be seen as a permutation introduced by the GA, a number of crossover operations are available for permutation of a chromosome such as Order crossover, Partially matched crossover (PMX), Position Crossover and Asexual Crossover [47,48]. All these types of crossovers have been dedicated to prevent repetition or missing the encoded values in the new ordered chromosomes. For instance, for the Order Crossover operation, two strings are aligned and two crossover points are selected and then a sliding motion applied for filling the left holes by transferring the mapped positions. For example, consider two chromosomes with 10 genes:

Parent 1: 4 8 7| 3 6 5| 1 10 9 2
Parent 2: 3 1 4| 2 7 9| 10 8 6 5

By merging parent 2 with parent 1, the places of 3, 6, 5 are left with holes.

Child 2: H 1 4 | 2 7 9 | 10 8 H H

The given holes will be filled with sliding motion that starts from the second crossover point:

Child 2: 2 7 9 | H H H | 10 8 1 4

Similarly the holes are filled from the parent 1, thus the resulted children will be represented as:

Child 1: 3 6 5 2 7 9 1 10 4 8
Child 2: 2 7 9 3 6 5 10 8 1 4

For this paper all the mentioned crossover operators have been tested. Order crossover and asexual crossover gave best results. Therefore, this type of operator has been adopted for the experiments.

5. Experimental setup

The GA used in all experiment is the generational GA with the parameters of the above described operators presented in Table 2. These parameters have not been chosen randomly, they were selected based on literature [35,39] and on experience after performing 300 experiments with different parameter settings. A supplement file (Parameters Setup) is attached separately to give more details about the parameter setup (see Appendix A).

In general, any new algorithm should be compared against similar existing algorithms. Since there was no standard benchmark for comparison, a set of systematic experiments has been performed to evaluate the performance of the proposed algorithms. The algorithm performance was tested in three parts:

Table 2
Algorithm parameters.

Genetic parameters	Magnitude
Crossover probability	0.8
Population size	75
Type of crossover	Order
Mutation rate	0.1
Selection type	Mini roulette
Size of chromosome	$N = 20$–4500

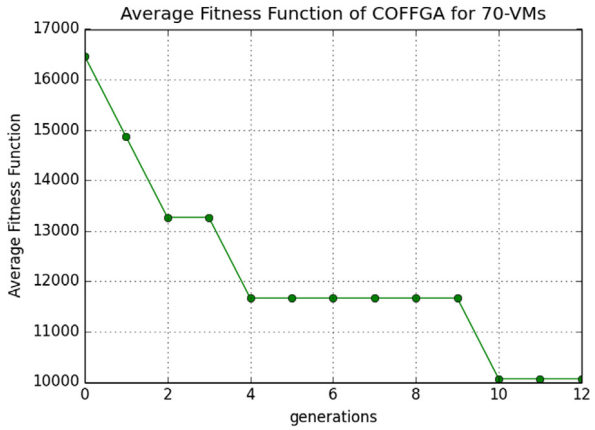

Fig. 3. Average COFFGA performance with 70 VMs problem instance.

(1) The first experiment is dedicated to the comparison of the two proposed algorithms CONFGA and COFFGA.
(2) The second experiment focuses on comparing the new algorithms with existing vector bin packing heuristics on a set of Multi-capacity vector Bin Packing Problems (MCVBPPs). These experiments have been designed to highlight the particular properties of the algorithms.
(3) The third experiment is dedicated to compare COFFGA with RGGA and other existing MCVBPP.

The proposed algorithms have been implemented in the C programming language and built on the LibGA development package [49]. LibGA is a set of fast routines written in C with a convenient developer interface. The underlying operating system is LINUX Ubuntu 14.4LTS.

5.1. Comparison of COFFGA and CONFGA

To evaluate the performance of the two new algorithms CONFGA and COFFGA, they were first compared against each other. The comparison was done using a data set that was developed particularly for this comparison. The data set contains 26 problem instances. The problems were constructed similar to the ways described in [48] where the authors based their analysis on Cloud servers capabilities [27,50]. However, the new data set has been modified to represent multi-capacity vector bin packing instead of the conventional bin packing [48]. The data set is constructed using the following parameters:

$N = [10–350]$ Number of virtual machines
$S = 128$ Number of servers.

$\overrightarrow{C1} = [\overrightarrow{C1,\ 1},\ \overrightarrow{C1,\ 2}]$ equals to [SRCi, SRMi], as defined in Table 1.
$Wj = [[C_{i,\,1}/3, C_{i,\,2}/5], [C_{i,\,1}/5, C_{i,\,2}/7], [C_{i,\,1}/9, C_{i,\,2}/3], [C_{i,\,1}/11, C_{i,\,2}/5], [C_{i,\,1}/7, C_{i,\,2}/9], [C_{i,\,1}/13, C_{i,\,2}/11]]j$ where W_j the vector of the requirements of the jth VM, $j = 1, \ldots, N$.

5.2. Conventional vector bin packing heuristic

Generally, any new algorithm should be compared with its predecessor from literature. This comparison focuses on demonstrating the benefit gained by using COFFGA and CONFGA against the conventional vector bin packing heuristics First Fit and First Fit Decreasing in terms of packing decision robustness. Given that the output of the new algorithms is a packing decision specifies which VM should be deployed to which server with the aim of minimising the number of servers (bins) and resources wastage, consequently maximising utilisation of each server. The packing decisions have been compared against the packing decisions generated by previous multi-dimensional bin packing heuristics. The comparison was done using the same data set described in (5.1).

5.3. Reordering grouping genetic algorithm

This comparison has been done to compare the proposed algorithm with literature sources that used genetic algorithm for vector bin packing. RGGA has been proposed in [35] with the similar target of solving multi-capacity vector bin packing in application to a Cloud Virtual Machines consolidation problem. The number of servers equals to the number of bins while the number of VMs equals to the number of items to be packed. For the comparison with RGGA here 10 sets of data were used. Data sets 0–9 deal with big problems of up to 4500 VMs. They deal also with different problems sizes of 500–4500 VMs. Their data in [35] was designed in a way that each VM can sit perfectly in a server and the servers can consolidate the VMs with zero residual capacities in all resources which do not happen in real cloud consolidation. However, RGGA is a steady state GA with 75 as a population size, with 0.8 crossover probability and 0.1 mutation probability.

6. Results

6.1. COFFGA and CONFGA

About 430 experiments were performed to evaluate the proposed algorithms (COFFGA and CONFGA) in terms of performance and packing robustness. These experiments were carried out using 26 problem instances (data initialisation as discussed in (5.1)); the problems sizes are varied by changing the number of VMs (20–340), and their CPU and memory requests.

Figs. 3 and 4 present the average performance of COFFGA and CONFGA in a problem of size 70 VMs over 8 runs. They show that COFFGA is competitively performed better than CONFGA; COFFGA finds minimum fitness function about 10 000, while the average fitness function of CONFGA is 30 000. At the same time, COFFGA easily converges after 10 generations, whereas CONFGA needs 250 generations. The main reason of this variance is that COFFGA uses the First Fit heuristic that provides backtracking packing feature which leads to produce minimum fitness function for the same chromosome. The minimum fitness function makes COFFGA converges easily and accompanies with better packing decision.

Fig. 5 compares the average of 8 fitness functions for 26 problem instances of COFFGA and CONFGA. It can be clearly seen that COFFFGA provides minimum fitness function for all cases, correspondingly COFFGA finds packing decision with minimum number of required servers. For each problem instance the fitness function solutions vary slightly in each run, but they do not extremely affect the number of required servers, almost the same number of servers over the 8 runs. In the worst case (+1) server is rarely required for the same problem instance across all experiments. This shows the precision and consistency in the performance of the proposed algorithms, and it also proves the reliability of the COFFGA and CONFGA. The results show

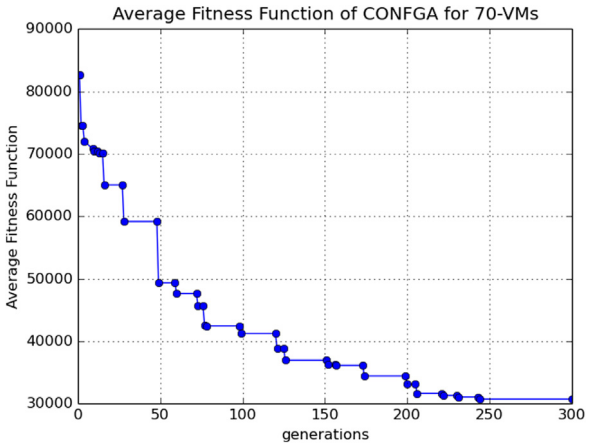

Fig. 4. Average CONFGA performance with 70 VMs problem instance.

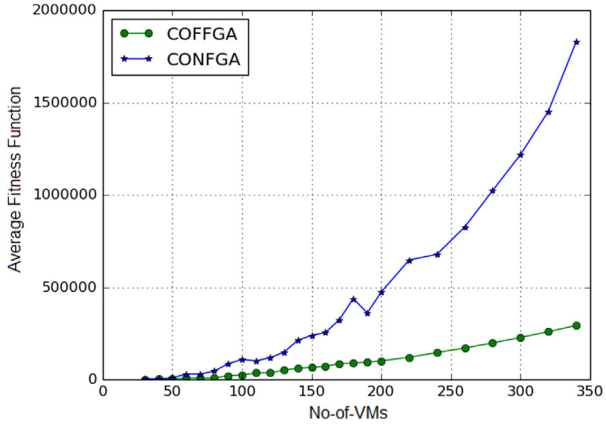

Fig. 5. Comparison between COFFGA and CONFGA in terms of average fitness function for 26 problem instances over 8 runs.

that COFFGA performed better than CONFGA for both metrics, i.e. performance and packing robustness. In term of performance, COFFGA needed 750–7500 function evolutions as average. The variation is caused by the problem sizes. CONFGA required about 1500–37 500 function evaluations.

Robust packing decision is a decision with best possible packing. Therefore, the decision within minimum number of servers is more robust than the other decisions. In terms of packing robustness, COFFGA appears more robust than CONFGA as its decision required less number of servers for all problem instances (see Fig. 6 in (Section 6.2)). Section 6.2 compares the number of required servers of the COFFGA and CONFGA for all problem instances.

6.2. Conventional vector bin packing heuristics

The proposed algorithms aim to find new solution for multi-capacity vector bin packing problems, therefore they were compared with the traditional multi-capacity first fit heuristic (MFF), and Multi-Capacity first fit decreasing heuristic (MFFD). The number of severs per packing solution in COFFGA and CONFGA for different problem sizes were compared against the number of servers associated with packing decision made by MFF and MFFD. MFF is implemented according to the definition of FF in Section 2. The main idea of MFFD is that VMs should be ordered before submitting to the FF algorithm. VMs order has been made by giving a priority to each VM according to $P(VM) = \sum_{j=1}^{r} W_j$, where r is

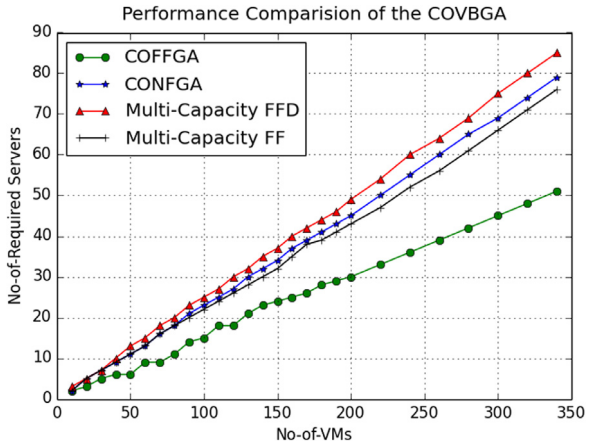

Fig. 6. Comparison combinatorial ordering vector bin-packing genetic algorithms (COVBGA) with preliminary vector bin packing heuristics.

the number of VMs requirements; then VMs order is configured according to VMs properties in a decreasing order.

Fig. 6 shows the comparison of CONFGA, COFFGA, MFF, and MFFD; it can be clearly observed that COFFGA far yields the least no of servers in all problem instances. It consistently produces best result than all other tested algorithms.

It has succeeded to reduce the number of used servers (PMs) by 39% when compared to Multi-Capacity FFD. At the same time, the number of servers is decreased by 34% compared to CONFGA. COFFGA has improved Multi-Capacity FF performance by 31.4%. It is also noticeable that CONFGA results are quite similar to MFF which means incorporating Next-Fit heuristic in ordering genetic algorithm has improved the performance of Next-Fit heuristic. Next Fit was considered earlier as an inefficient solution for vector bin packing [20]. FFD shows the worst performance among the compared algorithms which means combining the item requirements for ordering the items is inefficient for finding a priority to order the given items in multi-capacity vector bin packing.

6.3. Reordering grouping genetic algorithm

In this section we will show that the proposed algorithms outperform RGGA which was considered the best genetic algorithm in terms of multi-capacity bin packing for virtual machines consolidation. The comparison was made using a specified data set of RGGA that comprises 10 problems of various sizes between 500 and 4500 VMs. RGGA developers devoted their data to be very hard in a way no free space is allowed per server. However, they knew the optimal packing for each problem. This paper aimed to do a fair comparison; therefore, here the set of algorithm parameters were set to the values used in RGGA as described in Section 5.3. COFFGA is able to find the optimal packing. Table 3 presents a comparison of CONFGA, COFFGA, RGGA, Permutation Pack (PP) and FFD over all problems.

It should be noted that COFFGA consistently overtakes RGGA and Permutation Pack (PP) for all problems by minimising the number of the servers by about 4%. It also significantly outperforms FFD in each problem instance.

Importantly, COFFGA is able to reduce the function evaluations used in RGGA by 77.33% from 7500 in RGGA to 1700 in COFFGA, while CONFGA decreasing this by 26.66% from 7500 in RGGA to 5500 in CONFGA. This comparison shows that the proposed algorithms perform well on different problem sizes and independently of the underlying data.

Table 3
Comparison between COFFGA, CONFGA and RGGA algorithms.

DATA SETS	Our result		RGGA result			
	CONFGA	COFFGA	Opt	FFD	PP	RGGA
DSET1	61	59	59	61	61.4	60
DSET2	116	112	112	121	117.8	113
DSET3	203	191	191	207	196.5	192
DSET4	226	216	216	234	223.9	217
DSET5	253	241	241	262	250.3	242
DSET6	281	267	267	269	277.7	268
DSET7	343	320	320	353	331.8	321
DSET8	399	371	371	412	384.7	372
DSET9	462	425	425	465	439.8	426.02
DSET10	523	481	481	541	499.5	482

7. Conclusions

In this paper, the problem of Cloud resources allocation and consolidation has been studied and solved by converting it to multi-capacity vector bin packing. The contribution lies in two areas: the first area is d-capacity vector bin packing, and the second area is optimisation in cloud resources allocation.

In terms of d-capacity vector bin packing our conclusions are: firstly, finding independent dimension order for items needed to be packed before applying heuristics highly affects packing decision in vector bin packing problem. Secondly, the proposed approach of wrapping heuristic in the ordering genetic algorithm has succeeded in providing superior d-capacity vector bin packing with robust packing decisions. For that, two new hybrid algorithms CONFGA and COFFGA have been proposed. It was found that COFFGA is superior to other tested algorithms as it reduced the number of severs by (39%) over FFD, (31%) compared with FF, and (4%) of PP.

In term of cloud resources allocation, it was analysed how the number of servers required to run a set of VMs can be minimised by applying genetic warping heuristic approach. The proposed algorithms COFFGA and CONFGA were applied. The algorithms were able to deal with different problem sizes ranging from 20 VMs to 4500 VMs. Minimising the number of running servers and the resources wastage in each server lead to maximising resources utilisation in total. The proposed algorithms are capable of producing VMs assignment decision which strongly increases the consolidation density without violating the quality of the required services.

In comparison with state-of-the-art multi-dimensional aware genetic algorithm for virtual machine consolidation, the algorithms succeeded to find the optimal assignment decision of RGGA data. The results show that COFFGA and CONFGA do not only offer improving solutions but also the expense for getting the solutions compared to RGGA, because it reduced the number of servers by finding the optimal solution and also reduced the function evaluations from 7500 in RGGA to 1700 in COFFGA and 5500 in CONFGA.

The future plan is to develop a comprehensive multi-objective scheduling algorithm in application to Cloud resources allocation. The algorithm shall deal with all Cloud resources allocation problems like load balancing, energy consumption, and Makespan. Another future plan is considering the dynamic Cloud resources allocation problem that needs to go through variable size multi-capacity vector bin packing.

Acknowledgement

This work has been supported by the Ministry of Higher Education and Scientific Research of Iraq. The authors gratefully acknowledge this support.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.future.2016.10.025.

References

[1] A. Escalante, A. Escalante, Handbook of Cloud Computing, Springer, New York, Dordrecht, Heidelberg, London, 2010, http://dx.doi.org/10.1007/978-1-4419-6524-0.
[2] C. Höfer, G. Karagiannis, Cloud computing services: taxonomy and comparison, J. Internet Serv. Appl. (2011) http://dx.doi.org/10.1007/s13174-011-0027-x, 19 June 2011 - Springer. [Online] http://link.springer.com/article/10.1007/s13174-011-0027-x#page-1.
[3] B. Wang, Y. Cheng, W. Chen, Efficient consolidation-aware VCPU scheduling on multicore virtualization platform, Future Gener. Comput. Syst. 56 (2016) 229–237. http://dx.doi.org/10.1016/j.future.2015.08.007.
[4] K. Radha, B. Rao, S. Babu, K. Rao, V. Reddy, P. Saikiran, Allocation of resources and scheduling in cloud computing with cloud migration, Int. J. Appl. Eng. Res. (ISSN: 0973-4562) 9 (19) (2014) http://news.bbc.co.uk/1/hi/world/americas/4808342.stm.
[5] J.W. Smith, I. Sommerville, Understanding tradeoffs between power usage and performance in a virtualized environment, in: IEEE Sixth International Conference on Cloud Computing, IEE Computer Society, 2013, http://dx.doi.org/10.1109/CLOUD.2013.138, ISBN: 978-0-7695-5028-2.
[6] Z.A. Mann, A taxonomy for the virtual machine allocation problem*, Int. J. Math. Models Methods Appl. Sci. 9 (2015) 269–276.
[7] J. Lin, C. Chen, C. Lin, Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications, Future Gener. Comput. Syst. 37 (2014) 478–487. http://dx.doi.org/10.1016/j.future.2013.12.034.
[8] S. Gabriel, M. Barbulescu, A comparison of the performance and scalability of Xen and KVM hypervisors, in: Proceeding to Networking in Education and Research International IEEE Conference, 2013 RoEduNet, twelth ed., pp. 1–6.
[9] N. Dahmania, F. Clautiauxb, S. Krichena, G. Talbib, Self-adaptive metaheuristics for solving a multi-objective 2-dimensional vector packing problem, Appl. Soft Comput. 16 (2014) 124–136.
[10] M. Stillwella, D. Schanzenbacha, F. Vivienb, H. Casanova, Resource allocation algorithms for virtualized service hosting platforms, Parallel Distrib. Comput. 70 (9) (2010) 962–974. http://dx.doi.org/10.1016/j.jpdc.2010.05.006.
[11] K. Chandrasekaran, U. Divakarla, Load balancing virtual machine resources in cloud using genetic algorithm, in: ICCN 2013, pp. 156–168.
[12] L. Corcoran, R. Wainwright, Using LibGA for solving different combinatorial optimization problems*, in: The Application Handbook of Genetic Algorithms, Vol. 1, CRC Press, 1995, pp. 143–172.
[13] J.W. Lin, C.H. Chen, C.Y. Lin, Integrating QoS awareness with virtualization in cloud computing systems for delays applications, Future Gener. Comput. Syst. 37 (2014) 478–487.
[14] M. Ramani, M. Bohara, Energy aware load balancing in cloud computing using virtual machines, J. Eng. Comput. Appl. Sci. 4 (1) (2011) 1–5.
[15] M. Kumara, S. Raghunathan, Heterogeneity and thermal aware adaptive heuristics for energy efficient consolidation of virtual machines in infrastructure clouds, J. Comput. System Sci. 82 (2016) 191–212.
[16] W. Shu, W. Wang, Y. Wang, A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing, J. Wirel. Commun. Netw. 64 (2014) http://dx.doi.org/10.1186/1687-1499-2014-64.
[17] M. Randles, D. Lamb, A. Taleb-Bendiab, A comparative study into distributed load balancing algorithms for cloud computing, in: IEEE International Conference on Advanced Information Networking and Applications Workshops, IEE Computer Society, 2010, pp. 551–556.
[18] H. Ferdaus, M. Murshed, Virtual machine consolidation in cloud data centers using ACO metaheuristic, in: Euro-Par 2014: Parallel Processing: 20th International Conference, Springer, 2014, pp. 306–317.
[19] M.E. Frincu, Scheduling highly available applications on cloud environments, Future Gener. Comput. Syst. 32 (1) (2014) 138–153.
[20] W. Leinberger, G. Karypis, V. Kumar, Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints, in: Proceedings International Conference on Parallel Processing 1999, IEEE, 1999, 404412, 0-7695-0350-0. http://dx.doi.org/10.1109/ICPP.1999.797428.
[21] R. Masson, T. Vidal, J. Michallet, P. Penna, V. Petrucci, A. Subramanian, H. Dubedout, An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems, Expert Syst. Appl. 40 (2013) 5266–5275. http://dx.doi.org/10.1016/j.eswa.2013.03.037.
[22] L. Dubies, Optimizing resources allocation while handling SLA violations in cloud computing platform, in: Proceedings of the IEEE 27th International Symposium on Parallel & Distributed Processing, IPDPS, pp: 79-87. 978-1-4673-6066-1. http://dx.doi.org/10.1109/IPDPS.2013.67.
[23] M. Stillwell, F. Vivien, Casanova, Virtual machine resource allocation for service hosting on heterogeneous distributed platforms, in: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium, 2012, pp. 786–797. http://dx.doi.org/10.1109/IPDPS.2012.75.
[24] W. Tian, X. Liu, C. Jin, Y. Zhong, LIF: A dynamic scheduling algorithm for cloud data centers considering multi-dimensional resources*, J. Inf. Comput. Sci. 10 (12) (2013) 3925–3937.

[25] A. Alahmadi, A. Alnowiser, M. Zhu, D. Che, P. Ghodous*, Enhanced first-fit decreasing algorithm for energy aware job scheduling in cloud, in: International Conference on Computational Science and Computational Intelligence, 2014.
[26] A. Wolke, C. Pfeiffer, Improving enterprise VM consolidation with high-dimensional load profiles, in: 2014 IEEE International Conference on Cloud Engineering, 978-1-4799-3766-0/14. http://dx.doi.org/10.1109/IC2E.2014.12.
[27] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, Validating Heuristic for Virtual Machines Consolidation, TechReport, Microsoft, Microsoft Research Silicon Valley, Mountain View, CA 94043, 2011.
[28] K. Maruyama, S. Chang, D. Tang, A general packing algorithm for multidimensional resource requirements, Int. J. Comput. Inf. Sci. 6 (1977) 131–149.
[29] C. Chekuri, Approximation Algorithms for Scheduling Problems (Ph.D. dissertation), Stanford University, 1998.
[30] R. Yesodha, T. Amudha, A comparative study on heuristic problems to solve bin packing problem, Internat. J. Found. Comput. Sci. Technol. (IJFCST) 2 (6) (2012) http://dx.doi.org/10.5121/ijfcst.2012.2603.
[31] S. Genaud, J. Gossa, Cost-wait trade-offs in client-side resource provisioning with elastic clouds, in: Proceeding of 2011 IEEE 4th International Conference on Cloud Computing, Published by IEEE Computer Society, 2011, http://dx.doi.org/10.1109/CLOUD.2011.23.
[32] S. Doddavula, M. Kaushik, A. Jain, Implementation of a fast vector packing algorithm and its application for server consolidation, in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011, pp. 332–339. http://dx.doi.org/10.1109/CloudCom.2011.52.
[33] Z. Xu, X. Xu, X. Zhao, Task scheduling based on multi-objective genetic algorithm in cloud computing*, J. Inf. Comput. Sci. (2015) 1429–1438. http://dx.doi.org/10.12733/jics20105468.
[34] N. Quang-Hung, P. Nien, N. Nam, N. Tuong, N. Thoai, A genetic algorithm for power aware virtual machine allocation in private cloud, in: International Federation for Information Processing 20132013, pp. 183–191.
[35] D. Wilcox, A. McNabb, K. Seppi, Solving virtual machine packing with a reordering grouping genetic algorithm, in: Congress on Evolutionary Computation, CEC, 2011 IEEE, 2011, pp. 362–369. ISBN:978-1-4244-7834-7. http://dx.doi.org/10.1109/CEC.2011.5949641.
[36] Y. Ding, X. Qin, L. Liu, T. Wang, Energy efficient scheduling of virtual machines in cloud with deadline constraint, Future Gener. Comput. Syst. 50 (2015) 62–74. http://dx.doi.org/10.1016/j.future.2015.02.001.
[37] J. Gu, J. Hu, T. Zhao, G. Sun, A new resource scheduling strategy based on genetic algorithm in cloud computing environment, J. Comput. 7 (1 /2012) (2012) 42–52. http://dx.doi.org/10.4304/jcp.7.1.42-52.
[38] R. Kaur, N. Ghumman, Load balancing tactics in cloud computing: A systematic study, Int. J. Adv. Sci. Tech. Res. (2014) available online on: http://www.rspublication.com/ijst/index.htm.
[39] J. Xu, J. Fortes, Multi-objective virtual machine placement in virtualized data center environments, in: International Conference on Green Computing and Communications IEEE, 2010. http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.137.
[40] H. Ravani, H. Bheda, V. Patel, Genetic algorithm based resource scheduling technique in cloud computing, Int. J. Adv. Res. Comput. Sci. Manag. Stud. 1 (2013) 168–174. available online at: www.ijarcsms.com.
[41] E. Browna, R. Sumichrastb, Impact of the replacement heuristic in a grouping genetic algorithm, Comput. Oper. Res. 30 (2003) 1575–1593.
[42] M. Quiroz-Castellanosa, L. Cruz-Reyesa, Jose T. Jimenezb, S. Claudia Gómez, H. Huacujaa, A. Alvimc, A grouping genetic algorithm with controlled gene transmission for the bin packing problem, Comput. Oper. Res. 55 (2015) 52–64. http://dx.doi.org/10.1016/j.cor.2014.10.010.
[43] L. Corcoran, R. Wainwright, LibGA: a user-friendly workbench for order-based genetic algorithm research, in: Proceedings of the 1993 ACM/SIGAPP: Symposium on Applied Computing: States of the Art and Practice, pp. 111-117. http://dx.doi.org/10.1145/162754.162828.
[44] H. Imia, T. Yakawa, A new design of genetic algorithm for bin packing, in: The 2003 Congress on Evolutionary Computation, Vol. 2, IEEE, ISBN: 0-7803-7804-0, 2003, pp. 1044–1049. http://dx.doi.org/10.1109/CEC.2003.1299783.
[45] R. Malhotra, N. Singh, Y. Singh, Genetic algorithms: Concepts, design for optimization of process controllers, Comput. Inf. Sci. 4 (2011) Canadian Center of Science and Education.
[46] K. Sastry, D. Goldberg, G. Kendall, Genetic algorithms, in: E. Burke, G. Kendall (Eds.), Search Methodologies (Introductory Tutorials in Optimization and Decision Support Techniques), Springer, New York, 2005, pp. 97–125. Affiliated with University of Illinois.
[47] S. Aggarwal, R. Garg, P. Goswami, A review paper on different encoding schemes used in genetic algorithms, Int. J. Adv. Res. Comput. Sci. Softw. Eng. 4 (1) (2014) ISSN: 2277 128X.
[48] R. Klein, A. Scholl, C. Jurgens, Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem, Comput. Oper. Res. 24 (7) (1997) 627–645.
[49] ftp://ftp.aic.nrl.navy.mil/pub/galist/src/ga/libga100.tar.Z.
[50] VMWare. Server consolidation overview, building a virtual infrastructure, September 2009.

Huda Hallawi is currently a Ph.D. student at School of Aerospace Manufacturing and Transport, Cranfield University (UK). She received the B.S. degree and M.S. degree in Computer Engineering and Information Technology from University of Technology, Baghdad in 2006 and 2009, respectively. She worked as assistant lecturer in Computer Department at Karbala University for three years. Her research interests include Cloud Computing, Resources Management and Scheduling Algorithms, Artificial Intelligence, and Smart Card Application Development.

Jörn Mehnen is a Reader in Computational Manufacturing at Cranfield University (UK). His research focuses on application of computer science technologies in industry. Dr. Mehnen hold the position of the Deputy Director of the EPSRC Centre for Through Life-Engineering Services at Cranfield University. He has been working on several EU project related to Cloud Computing and Cloud Manufacturing and is author of the Springer book "Cloud Manufacturing". Dr. Mehnen got a prestigious EPSRC HVM Catapult Fellowship award which concerns the practical application of IoT devices in manufacturing environments. He is also working on projects concerning Big Data, data analytics and data visualisation for industry. With more than 100 journal, book and conference papers he has published extensively both in the field of computer science as well as manufacturing.

Hongmei He has been a lecturer in Cybersecurity for Manufacturing and Information Intelligence in the Manufacturing Informatics Centre at Cranfield University since Oct 2013. Previously, she worked for the University of Kent, the University of Ulster, and the University of Bristol, successively. She received the Ph.D. in Computer Science and the M.Sc. in Multimedia and Internet Computing from Loughborough University in 2006 and 2003, respectively.

Her research involves Multidisciplinary Computing around the core of Computational Intelligence for a wide range of applications, such as Cybersecurity, Big Data Analytics, Optimisation, Complex Networks, and Robotics & Autonomous Wireless Sensor Networks.