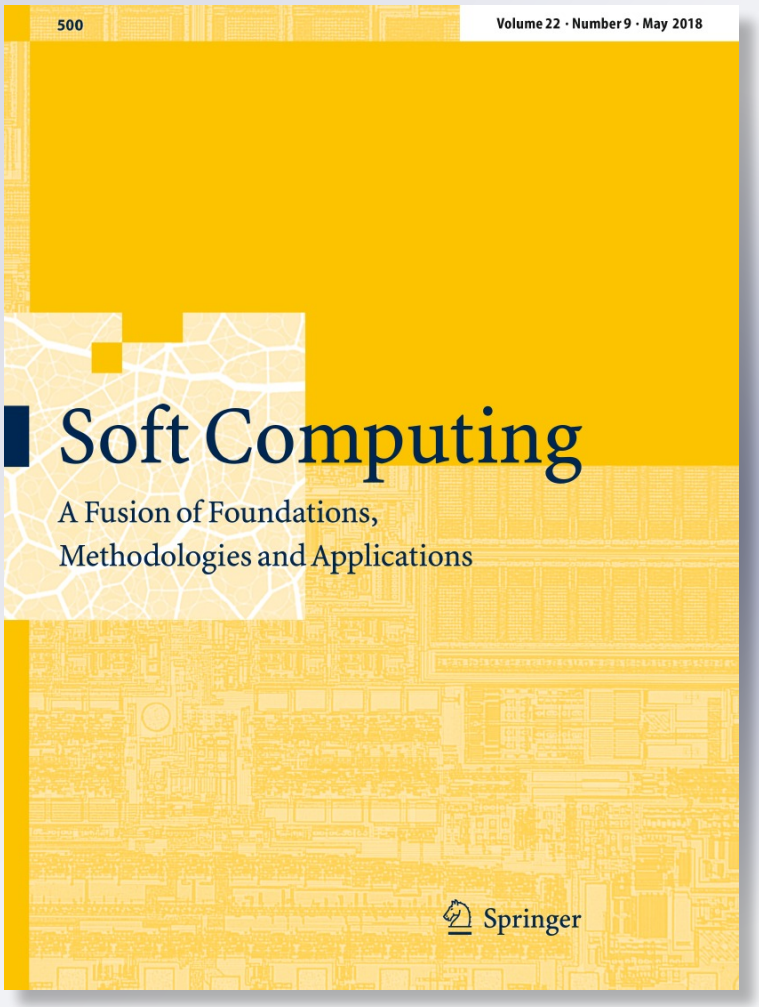
Engineering simulated evolution for integrated power optimization in data centers

Sadiq M. Sait & Ali Raza

Soft Computing
A Fusion of Foundations,
Methodologies and Applications

ISSN 1432-7643
Volume 22
Number 9

Soft Comput (2018) 22:3033-3048
DOI 10.1007/s00500-017-2556-0



Springer

Soft Comput (2018) 22:3033–3048
https://doi.org/10.1007/s00500-017-2556-0

METHODOLOGIES AND APPLICATION

Engineering simulated evolution for integrated power optimization in data centers

Sadiq M. Sait[1] · **Ali Raza**[2]

Published online: 1 April 2017

Abstract Cloud computing has evolved as the next-generation platform for hosting applications ranging from engineering to sciences, and from social networking to media content delivery. The numerous data centers, employed to provide cloud services, consume large amounts of electrical power, both for their functioning and their cooling. Improving power efficiency, that is, decreasing the total power consumed, has become an increasingly important task for many data centers for reasons such as cost, infrastructural limits, and mitigating negative environmental impact. Power management is a challenging optimization problem due to the scale of modern data centers. Most published work focuses on power management in computing nodes and the cooling facility in an isolated manner. In this paper, we use a combination of server consolidation and thermal management to optimize the total power consumed by the computing nodes and the cooling facility. We describe the engineering of an evolutionary non-deterministic iterative heuristic known as simulated evolution to find the best location for each virtual machine (VM) in a data center based on computational power and data center heat recirculation model to optimize total power consumption. A "goodness" function which is related to the target objectives of the problem is defined. It guides the moves and helps traverse the search space using artificial intelligence. In the process of evolution, VMs with high goodness value have a smaller probability of getting perturbed, while those with lower goodness value may be reallocated via a compound move. Results are compared with those published in previous studies, and it is found that the proposed approach is efficient both in terms of solution quality and computational time.

Keywords Cloud computing · Power management · Resource provisioning · Virtual machine assignment · Combinatorial optimization · Simulated evolution · Non-deterministic algorithms · NP hard problems

Communicated by V. Loia.

✉ Sadiq M. Sait
sadiq@kfupm.edu.sa

Ali Raza
aliraza@kfupm.edu.sa

[1] Department of Computer Engineering and Center for Communications and IT Research, Research Institute, King Fahd University of Petroleum and Minerals, Dhahran 31261, Kingdom of Saudi Arabia

[2] Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Kingdom of Saudi Arabia

1 Introduction

Cloud-based data centers have emerged as a popular computing paradigm. The IT industry is rapidly adopting cloud for hosting and delivering Internet-based applications and services (Systems IT Governance Research Team 2008). As a result of such a proliferation, there has been an increase in the power density and power consumption of data centers. In 2013, 91 Billion kWh of power was consumed by US data centers. It is estimated to increase to approximately 140 Billion kWh by 2020, which is equivalent to $13 Billion per year in electricity bills, and emission of nearly 150 Million Metric tons of annual carbon pollution (The Climate Group on behalf of the Global eSustainability Initiative (GeSI) 2015).

In some cases, power supply available for data centers is limited. For example, Wall Street bank Morgan Stanley was not able to run a new data center in Manhattan due to unavailability of the power needed to operate the center (Brown and Reams 2010). 30% of data center providers have identi-

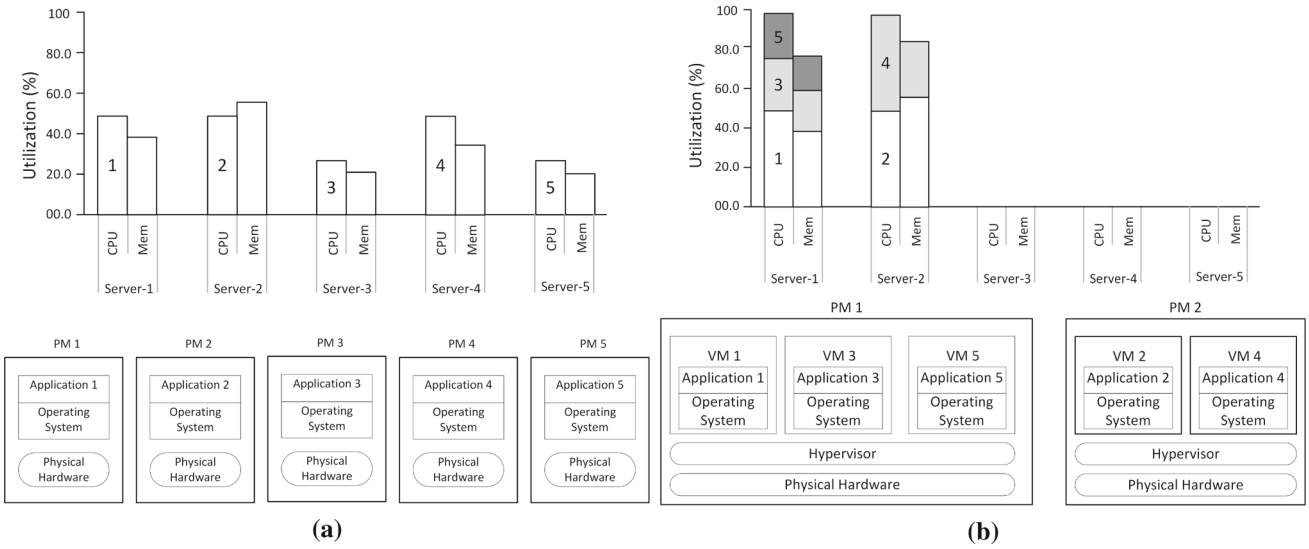


Fig. 1 **a** Applications running on individual servers, **b** applications running on individual VMs sharing physical resources

fied power availability as a key factor being limiting server deployment (Filani et al. 2008). Power required by the computing nodes and the cooling facility in a data center is a crucial issue for data center providers since it dominates the operational costs (Systems IT Governance Research Team 2008). For example, based on a 3-year amortization schedule, Amazon estimated that cost and operation of the severs at its data centers accounts for 53% of the total budget, and cost related to power is 42% of the total, including infrastructure for cooling and direct power consumption (Hamilton 2009). Therefore, considerable amounts of power can be saved, and significant contribution can be made to greater environmental sustainability by improving power efficiency in cloud data centers.

A typical data center hosts number of servers corresponding to maximum load, but the utilization invariably is around 30–70% (Barroso and Hölzle 2007; Bohrer et al. 2002). Significant percentage of servers sits idle waiting for user requests and consumes about 60–70% of the power required when fully utilized (Basmadjian et al. 2012; Feng et al. 2005; Fu et al. 2010). This significant idle power not only adds to the computational power but also to that consumed by the cooling facility to remove heat. Server consolidation has been proposed to reduce idle power by assigning user requests to a subset of servers and powering-off the rest (Chase et al. 2001; Pinheiro et al. 2001; Sait 2016). More servers are powered-on as more user requests arrive. Server consolidation makes use of virtualization, a prominent technology that makes cloud computing possible. Virtualization allows resources of a single large server to be sliced into multiple isolated execution environments so that multiple operating systems can coexist on a single physical machine. User requests are translated into computational requirements that are mapped to VMs with desired characteristics. Multiple VMs are assigned to a single physical server, sharing the same underlying machine's computing resources, which results in fewer physical servers (Barham et al. 2003). Figure 1 illustrates the benefits of virtualization and how it helps to minimize the number of active servers.

As shown, five applications are running on five different servers. This is a wastage of resources since all of the servers are underutilized. With virtualization, we can translate these applications into VMs, and those VMs can run on fewer servers, as illustrated in Fig. 1b.

Server consolidation attempts to assign VMs to a minimum number of servers and this results in high utilization per server. Servers consume large amounts of computational power because of higher utilization, and this power is dissipated as heat. Consequently, the active servers reach higher temperature due to the creation of hot spots by the heat dissipation. This heat needs to be extracted to avoid overheating and resultant failing of the servers (Wang et al. 2009). Computer Room AC (CRAC) units provide cold air, which enters the servers through the front air inlets in the cold aisle, picks up heat from the circuitry, and exits via outlet to the hot aisles. Accordingly, the outlet temperature rises compared to the inlet temperature. Air conditioners positioned above the hot aisle extract this hot air, but a large fraction of this heat recirculates to the cold aisle increasing the inlet temperature. The temperature of supplied air is adjusted (reduced) so that the inlet temperature is lower than the safe value to avoid overheating of the servers. Cooling facilities exert more effort to provide cold air at lower temperature, i.e., more electric power is required by the cooling facility at higher inlet temperature. For the VM assignment shown in Fig. 1b, servers reach high temperature due to hot spots; therefore, the CRAC

unit will consume a large amount of power to supply cold air to reduce the temperature of the servers to a safe limit. Hence, minimizing the number of servers does not necessarily minimize total power consumed. Computational power and cooling power must be optimized in an integrated fashion to minimize the total power consumption in data centers.

In this paper, we present a simulated evolution (SimE)-based heuristic. We developed a goodness measure to intelligently traverse the search space. Our approach finds a near-optimal VM assignment by considering the trade-off between computational power and cooling power in short amount of time. Its performance is compared with that of two well-known iterative heuristics, namely Simulated Annealing (SA) and Tabu search (TS), and with two popular constructive algorithms, the improved versions of First Fit Decreasing FFD_{imp} and Least-Loaded algorithm LL_{imp}. Results are presented that demonstrate the effectiveness of the proposed algorithm over a wide range of problem instances.

The remainder of this paper is organized as follows: Sect. 2 discusses related work in this area. Section 3 describes data center configuration and power model. Section 4 formally defines the problem, and Sect. 5 explains our proposed approach. Experimental methodology and results are discussed in Sect. 6. Section 7 concludes this work and suggests future work.

2 Related work

In the last few years, much work has been done for optimal VM assignment with the objective of minimizing number of active servers (PMs), thereby decreasing computational power. Gao et al. (2013) proposed a modified ant colony optimization algorithm that minimizes total resource wastage and power consumption in physical machines. Xu et al. (2015) proposed improved multi-objective particle swarm optimization (IMOPSO) for reducing computational power and migration time. Kramer et al. (2012) used server consolidation with the concept of dynamic voltage/frequency scaling (DVFS) to improve the power efficiency. Wang et al. (2016) used genetic algorithm based on the number of physical machines and Service Level agreements to reduce computational power, but they did not consider cooling power. Doddavula et al. (2011) proposed a Magnitude Classified algorithm based on First Fit Decreasing (FFD) for server consolidation. Ajiro and Tanaka (2007) suggested improvements to the classical FFD and Least-Loaded (LL) algorithms to optimize computational power. All of these schemes attempt to minimize computational power by minimizing the number of active severs, but also ignore cooling infrastructure despite it being a significant percentage of total power consumed.

To minimize the power consumption of cooling infrastructure, Sullivan (2000) and Patel et al. (2003) optimized cooling power by improving air flow. Moore et al. (2005) proposed to predict heat profiles using software-based infrastructure. Mukherjee et al. (2007) used a software-based infrastructure to control resource management for thermal-aware allocation. These schemes only optimize cooling power.

Al-Qawasmeh et al. (2015) used nonlinear programming technique to optimize cooling and idle power. Nonlinear programming technique has poor scalability, and it may not be suitable for practical scenarios.

In this work, we combine linear power model (Fan et al. 2007) and a heat recirculation model (Tang et al. 2006) to address the problem of minimizing total power consumption in data centers by considering the trade-off between the computational power and the cooling power. We engineer an evolutionary non-deterministic optimization heuristic known as simulated evolution (SimE). Similar to other non-deterministic search algorithms, SimE is based on moves and possesses hill-climbing capability. One key requirement of SimE is to define an appropriate way to estimate the *goodness* of the current assignment of a movable element. In our case, the movable elements are VMs and goodness measure is based on server consolidation and thermal management. The process of evolution, guided by goodness value, tends to converge reasonably fast to a good quality solution by considering the trade-off between the computational power and the cooling power. Many other non-deterministic heuristics, such as Simulated Annealing (SA), Tabu Search (TS), lack this domain knowledge feature and work mostly with random moves. Further details of goodness function developed for VM assignment are provided in Sect. 5.

3 Data center power model

This section provides the necessary preliminaries and system models that are required to define the problem. Table 1 shows the parameters and notations used throughout this paper.

3.1 A typical data center configuration

A typical data center is arranged in a hot-aisle/cold-aisle configuration as shown in Fig. 2. Racks containing servers are installed on a raised floor. All servers are connected to a high-speed network, typically in star topology via central switch. CRAC units extract hot air from the top and deliver cold air through pressurized floor plenum.

3.2 Power consumption in data centers

The total power (P_{total}) demand of a data center is defined as a sum of computational power (P_{CN}) and cooling power (P_{AC}). For the sake simplicity we are not considering power consumed by network devices; storage; lightning; humidifier;

Table 1 Notations and definitions

Symbol	Definition
VM	Virtual machine
PM	Physical machine
n	Number of VMs
m	Number of servers (PMs)
P_{total}	Total power
P_{AC}	Cooling power
P_{CN}	Computational power
P_j^{idle}	Idle computational power of jth server
P_j^{busy}	Average busy computational power of jth server
U_j^p	CPU utilization of jth server
U_j^m	Memory utilization of jth server
x_{ij}	1 if ith VM is assigned to jth server otherwise 0
y_j	1 if jth server is ON otherwise 0
v_i^c	CPU requirement of ith VM
v_i^m	Memory requirement of ith VM
COP	Coefficient of performance
T_{sup}	Temperature of air supplied by CRAC
T_{red}	Maximum allowed inlet temperature
ρ	Air density in kg/m^3
f	Air flow rate in m^3/s
Q	Heat rate in Watt (W)
c_p	Specific heat of air in kJ/kg K
A	Heat cross-interference coefficient matrix
T_{out}^i	Inlet temperature of ith server
T_{in}^i	Outlet temperature of ith server
K	Thermodynamic constant matrix, $\mathbf{K} = \text{diag}(K_i)$
$\overrightarrow{T_{\text{in}}}$	The vector $\{T_{\text{in}}^i\}_n$
$\overrightarrow{T_{\text{out}}}$	The vector $\{T_{\text{out}}^i\}_n$
D	Distribution matrix

and losses due to distribution network consisting of switch gear, conductors, DC–AC and AC–DC converters, and UPS (uninterruptible power source).

$$P_{\text{total}} = P_{\text{CN}} + P_{\text{AC}} \tag{1}$$

3.2.1 Computational power

Power consumption by computing nodes varies with the computing activity. Fan et al. (2007) proposed that computational power consumption of servers can be accurately described by a linear model. Power consumed by a server j can be express as in Eq. 2 (Gao et al. 2013).

$$P_j = \begin{cases} \left(P_j^{\text{busy}} - P_j^{\text{idle}}\right) \times U_j^p + P_j^{\text{idle}}: & \text{otherwise} \\ 0: & \text{if } U_j^p = 0 \end{cases} \tag{2}$$

where U_j^p is the CPU utilization of jth server, P_j^{busy} is the average power value when jth server is fully utilized, and P_j^{idle} is the average power value when the server is in idle state.

Total power consumption of all computing nodes (P_{CN}) in a data center can be calculated as:

$$P_{\text{CN}} = \sum_{j=1}^{m} P_j \tag{3}$$

where v_i^c is the CPU requirement of ith VM, and x_{ij} is the assignment variable, its value is 1 if ith VM is assigned to the jth server otherwise its value is zero.

3.2.2 Cooling power

Power consumed by the cooling facility is defined as given in Eq. 4 (Moore et al. 2005).

$$P_{\text{AC}} = \frac{P_{\text{CN}}}{\text{COP}(T_{\text{sup}})} \tag{4}$$

where P_{CN} is the power consumed by computing nodes. P_{AC} is inversely proportional to *Coefficient of Performance* (COP). COP varies with temperature of the air supplied (T_{sup})

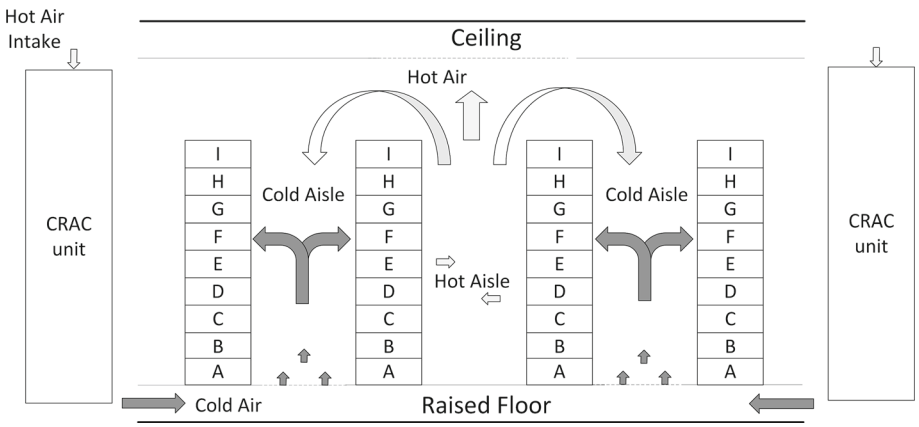


Fig. 2 A typical data center configuration

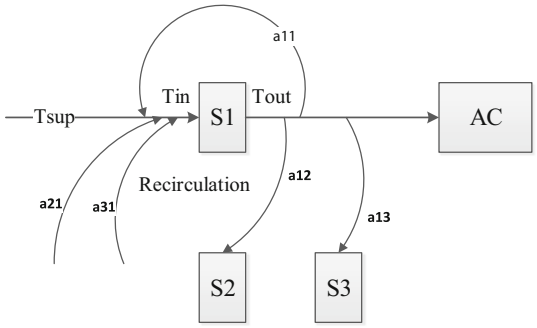


Fig. 3 Cross-interference among computational nodes. Exhaust hot air from server S1 will be partially removed by AC and will partially recirculate to other servers' inlets. It is also affected by the hot air from other servers

by the CRAC units. We are using COP model of chilled-water CRAC units at the HP Labs Utility Data Center (Moore et al. 2005), which is defined by Eq. 5.

$$\mathrm{COP}(T_{\mathrm{sup}}) = 0.0068T_{\mathrm{sup}}^2 + 0.0008T_{\mathrm{sup}} + 0.458 \tag{5}$$

As temperature (T_{sup}) of the cold air supplied by CRAC units air increases, COP increases, and CRAC units use less cooling power (P_{AC}) to remove the same amount of heat (computational power).

3.2.3 Total power consumption

In this work, we combined linear power model (Eq. 3) proposed in Fan et al. (2007) and cooling power model (Eq. 4) proposed in Tang et al. (2006) to address the problem of minimizing total power consumption in data centers by considering the trade-off between the computational power and the cooling power. Using Eqs. 3 and 4, total power consumption can be written as:

$$\begin{aligned} P_{\mathrm{total}} &= \left(1 + \frac{1}{\mathrm{COP}(T_{\mathrm{sup}})}\right) P_{\mathrm{CN}} \\ &= \left(1 + \frac{1}{\mathrm{COP}(T_{\mathrm{sup}})}\right) \sum_{j=1}^{m} P_j \end{aligned} \tag{6}$$

3.3 Heat recirculation

Heat recirculation can be described as a phenomenon of a server's outlet heat recirculating and affecting the inlet temperature of another server. Tang et al. (2006, 2008) showed that a cross-interference matrix, obtained by computational fluid dynamic simulation, can be used to define heat recirculation. Cross-interference matrix is denoted as $A_{m\times m}$, where each element a_{ij} is the fraction of heat transferred from the outlet of ith server to the inlet of jth server (Fig. 3).

The increase in the inlet temperature of the servers due to heat recirculation is given as (Tang et al. 2006):

$$\overrightarrow{T_{\mathrm{in}}} = \overrightarrow{T}_{\mathrm{in_{old}}} + \mathbf{D} \times \overrightarrow{P} \tag{7}$$

where $\overrightarrow{T_{\mathrm{in}}}$ is the inlet temperature vector, $\overrightarrow{P}$ represents computational-power-consumption vector of m servers, and $\overrightarrow{T}_{\mathrm{in_{old}}}$ is inlet temperature vector before loading. Distribution matrix $\mathbf{D}$ is defined as:

$$\mathbf{D} = [(\mathbf{K} - \mathbf{A}^{\mathrm{T}}\mathbf{K})^{-1} - \mathbf{K}^{-1}] \tag{8}$$

where $\mathbf{K}$ is a $m \times m$ diagonal matrix whose entries are thermodynamic constants of the servers, i.e., $K_i = \rho f_i c_p$.

This means that each inlet temperature rises due to heat from recirculation. The temperature of supplied air is adjusted so that the inlet temperature is lower than a manufacturer specified value (T_{red}). This is necessary to avoid overheating and failing of the servers (Wang et al. 2009). In this work, we use $T_{\mathrm{red}} = 25\,^{\circ}\mathrm{C}$ (Tang et al. 2006).

$$T_{\mathrm{sup}} = T_{\mathrm{in_{old}}} + T_{\mathrm{adj}} \tag{9}$$

where

$$T_{\mathrm{adj}} = T_{\mathrm{red}} - \max(T_{\mathrm{in}}) \tag{10}$$

3.4 An illustrative example

Suppose the distribution vector $\mathbf{D}$ of the data center shown in Fig. 1 is given as:

$$\mathbf{D} = \begin{bmatrix} 0.0723 & 0.0252 & 0.0071 & 0.0097 \\ 0.0128 & 0.0463 & 0.0009 & 0.0092 \\ 0.0157 & 0.0025 & 0.0272 & 0.0139 \\ 0.0035 & 0.0009 & 0.0076 & 0.0455 \end{bmatrix}$$

The distribution vector is obtained through computational fluid dynamic simulation of the given data center's configuration. Similarly, suppose $T_{\mathrm{sup}} = 25\,^{\circ}\mathrm{C}$, $P_{\mathrm{idle}} = 150\,\mathrm{W}$, and $P_{\mathrm{busy}} = 215\,\mathrm{W}$. The flow rate ($f_i$) of each server's fan is assumed to be 8.0 m^3/s. For the VMs assignment shown in Fig. 1b, servers 1 and 2 consume 215 W of power; whereas, servers 3, 4, and 5 are turned off. Using Eq. 7, inlet temperature (T_{in}) of these five servers is 46, 37, 29, 26, and 27 °C. The temperature (T_{sup}) of air supplied by the CRAC units must be 4 °C to ensure that $T_{\mathrm{in}} \leq T_{\mathrm{red}}$. Total power ($P_{\mathrm{total}}$) consumed is given by Eq. 6

$$\begin{aligned} P_{\mathrm{total}} &= \left(1 + \frac{1}{\mathrm{COP}(4)}\right)(215 + 215 + 0 + 0 + 0) \\ &= 1186.72\,\mathrm{Watts} \end{aligned}$$

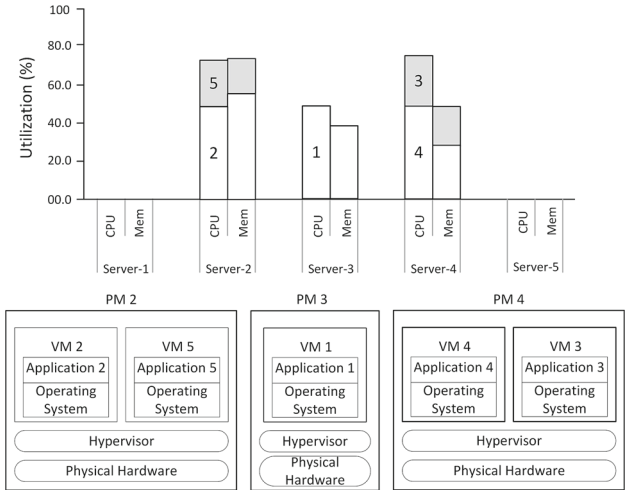


Fig. 4 VMs allocated to avoid hot spots

Now, consider the VM assignment shown in Fig. 4. Servers 2,3, and 4 are consuming 198.75, 182.50, and 198.75 W of power, respectively; whereas, servers 1 and 5 are turned off. Inlet temperature (T_{in}) of the five servers is 33, 36, 33, 35, 26 °C. The temperature (T_{sup}) of air supplied by the CRAC units must be 14 °C to ensure that temperature of the servers is within the safe limit.

Therefore,

$$P_{\text{total}} = \left(1 + \frac{1}{\text{COP}(14)}\right)(0 + 198 + 182 + 198 + 0) = 909.44 \text{ Watts}$$

This allocation of VMs is using more servers as compared to the assignment shown in Fig. 1b, but it is consuming less total power. Hence, minimizing the number of servers does not necessarily minimize total power consumption. Computational power and cooling power must be optimized in an integrated fashion to minimize the total power consumption in data centers.

4 Problem formulation

In this section, we formally define the virtual machine placement problem and discuss the cost function and constraints.

We consider two dimensions, CPU and memory to characterize a VM and PM. Suppose there are n VMs to be assigned. Every VM $v_i, i \in \{1, 2, 3, \ldots, n\}$ is defined as a two-dimensional requirement vector, $v_i = \{v_i^c, v_i^m\}$ where each dimension represents a normalized value of one type of resource requested (CPU and memory). These VMs are to be allocated to n PMs with the assumption that every VM request can be satisfied by one sever. We assume a homogeneous data center, where all PMs have the same capacity.

Let T_j^c and T_j^m be the threshold values of CPU and memory resources, associated with each PM p_j, $j \in \{1, 2, 3, \ldots, m\}$, respectively. The assignment solution is represented by a $m \times n$ matrix X, where:

$$x_{i,j} = \begin{cases} 1: & \text{if } i\text{th VM is assigned to the } j\text{th PM} \\ 0: & \text{otherwise} \end{cases} \quad (11)$$

In addition, we define the following binary decision variable:

$$y_j = \begin{cases} 1: & \text{if } j\text{th PM is in use} \\ 0: & \text{otherwise} \end{cases} \quad (12)$$

The given problem can be formulated as:

$$\text{minimize } P_{\text{total}} = \left(1 + \frac{1}{\text{COP}(T_{\text{sup}})}\right) \sum_{j=1}^{m} P_j$$

subject to

$$\sum_{i=1}^{n} v_i^c \times x_{ij} \leq T_j^c \times y_j \quad \forall j \in J \quad (13)$$

$$\sum_{i=1}^{n} v_i^m \times x_{ij} \leq T_j^m \times y_j \quad \forall j \in J \quad (14)$$

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in I \quad (15)$$

$$y_j, x_{ij} \in \{0, 1\} \quad \forall i \in I \quad \text{and} \quad \forall j \in J \quad (16)$$

$$T_{in} \leq T_{\text{red}} \quad (17)$$

Constraints (13) and (14) guarantee that the threshold capacity of each server is not exceeded. Moreover, constraint (15) ensures that a VM is placed in exactly one server and constraint (16) represents the domain of variables $x_{i,j}$ and y_j. Finally, constraint (17) makes sure that temperature of the servers is within the given safe limit.

5 Proposed approach

In this section we describe the approach adopted to allocating VMs while optimizing our objectives.

As mentioned earlier, minimizing the number of servers does not necessarily minimize total power consumption. This was illustrated by an example in Sect. 3.4. There is a trade-off between computational and cooling power. The implications of the hot spots on the cooling power are discussed in Sect. 3.3. We are interested in the VM assignment problem with objective of minimizing total power consumption using our model. VM assignment problem is often formulated as a vector bin packing problem (VBP), where the VMs that

are treated as objects (n) are packed into servers that are treated as bins (b). The computational complexity of VBP is $O(b^n)$. Clearly, it is impractical to enumerate all possible assignments for a large number of VMs (objects). Even the one-dimensional version of this problem is NP hard (Sait and Shahid 2015; Sait 2016). Finding optimal assignment to such problems is computationally infeasible for large problem sets. In this work, we engineered simulated evolution (SimE) for optimal placement of VMs to minimize total power consumption in data centers in minimal time.

SimE is a very popular heuristic which the authors have used to solve many optimization problems (Khan et al. 2002; Youssef et al. 2002). Similar to other non-deterministic search heuristics, SimE is also based on moves and possesses hill-climbing capability to find the solution with lowest costs reasonably quickly. One major difference between SimE and other heuristics is the goodness function. A key requirement of SimE is to come up with goodness function(s) that play a key role in guiding the traversal of search space to find optimal solution in lesser time. This goodness function requires domain knowledge of the problem and cognition of the designer of the heuristic.

In this section we describe our simulated evolution (SimE)-based VM assignment algorithm. We begin with a brief discussion of the basic SimE heuristic.

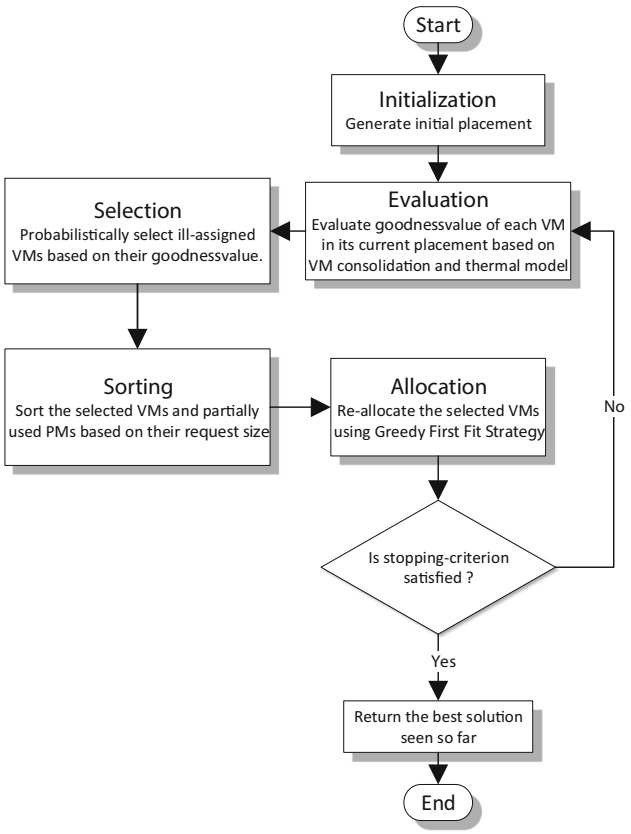


Fig. 5 Flowchart of SimE

5.1 Simulated evolution algorithm

Simulated evolution (SimE) algorithm was proposed by Kling and Banerjee (1987). The algorithm combines constructive perturbation and iterative improvement and saves itself from getting stuck to the local minima by following a stochastic approach. The core of the algorithm is the goodness estimator. SimE assigns each moveable element a goodness value. The goodness value indicates how well a certain movable element is currently assigned. The more the goodness value, the lesser is the probability of the element being selected for re-allocation.

The flow of the proposed SimE-based VM allocation algorithm for total power minimization in data centers is shown in Fig. 5. SimE starts with an initial solution Φ of a set V containing n movable elements (VMs). SimE then follows an evolution-based approach to find better solutions from one iteration to the next by perturbing some ill-assigned elements (VMs) while retaining the near-optimal ones. The algorithm consists of three sequential steps; evaluation, selection, and allocation; that are executed in each iteration. The process of iterative improvements continues until the solution average goodness value reaches at its maximum, or no considerable improvement in solution quality is observed after a given number of iterations (Sait and Youssef 1999).

5.1.1 Goodness evaluation

This step involves the evaluation of goodness (fitness) g_i of each VM v_i assigned to PM p_k in current solution Φ'. Effective goodness measures can be thought of based on the domain knowledge of the optimization problem (Sait and Youssef 1994, 1999). This goodness measure is expressed as a single number in the range of zero to one. For our VM assignment problem, we used a joint goodness function that is based on our objective; i.e., to reduce computational and cooling power.

Computational Power is directly proportional to the number of active severs as given by Eq. 3. We can save computational power by assigning the given VMs to fewer number of servers. We define a goodness value gs of ith VM assigned to kth server as:

$$gs_i = \frac{v_i^c + v_i^m}{p_k^c + p_k^m}, \quad gs_i \in [0, 1] \tag{18}$$

where v_i^c and v_i^m are CPU and memory requirements of VM v_i, and p_k^c and p_k^m are the available CPU and memory resources of partially used PM p_k after removing VM v_i from PM p_k in the current solution Φ'. Equation (18) assumes a minimization of resource wastage in PM p_k (Fig. 6). The

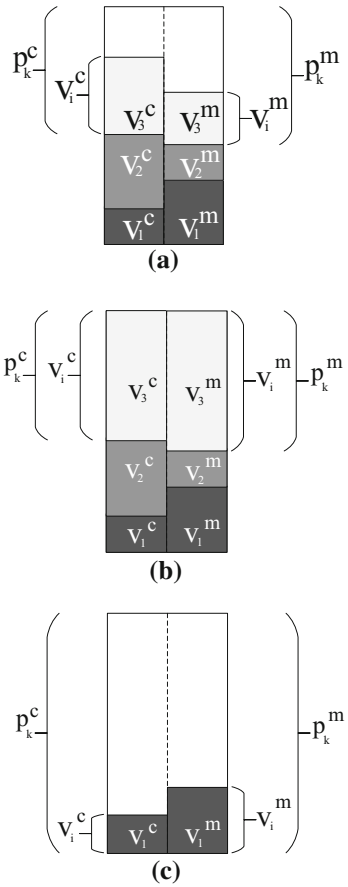


Fig. 6 **a** Goodness measure for minimizing number of servers $gs_i = \frac{v_i^c + v_i^m}{p_k^c + p_k^m}$, **b** the 3rd VM has a goodness measure of 1 and it should not be selected for re-allocation, **c** the 1st VM has a goodness value of 0, it is ill-assigned and should be reallocated

objective of this goodness measure is to minimize the number of active servers (Sait and Shahid 2015).

Our other objective is to minimize cooling power consumption. Hot spots are created in data centers due to heat recirculation. Some of these recirculation effects can lead to situations where the observed consequence of the inefficiency is spatially uncorrelated with its cause; in other words, the heat vented by one machine may travel several meters before arriving at the inlet of another server. From Eq. 7, we can define increment in the inlet temperature ($\delta\overrightarrow{T_{\text{in}}}$) of the severs due to heat recirculation as:

$$\delta\overrightarrow{T_{\text{in}}} = \mathbf{D} \times \overrightarrow{P} \tag{19}$$

where $\delta\overrightarrow{T_{\text{in}}} = \{\delta T_{\text{in}}^1, \delta T_{\text{in}}^2, \delta T_{\text{in}}^3, \ldots, \delta T_{\text{in}}^m\}$ and $\mathbf{D}$ is given by Eq. 8. Similarly, increment in the inlet temperature of the ith server due to heat recirculation of jth server is given as:

$$\delta\overrightarrow{T_{\text{in}}^i}(j) = \overrightarrow{d_{i,j}} \times \overrightarrow{p_j} \tag{20}$$

We define Recirculation Effect (RE) of jth server as:

$$\text{RE}_j = \sum_{i=1}^{m} \delta\overrightarrow{T_{\text{in}}^i}(j) = \sum_{i=1}^{m} \overrightarrow{d_{i,j}} \tag{21}$$

RE indicates the contribution of a server to heat recirculation and creation of hot spots (Fig. 7). In order to avoid hot spots and to minimize cooling power (P_{AC}) consumption, servers with higher RE value must be avoided. RE value can be used to define the goodness value of VMs assigned to a server. If VMs are assigned to the servers with higher RE value, they should have lower goodness value and should be considered for re-allocation. Since the goodness measure must be a single number expressible in the range [0, 1]. We translate RE to a goodness measure gt as:

$$gt_i = 1 - \frac{\text{RE}_j - \min(\text{RE})}{\max(\text{RE})}, \quad gt_i \in [0, 1] \tag{22}$$

where gt_i is the goodness of ith VM assigned to jth server. The goodness value gt focuses on inefficiencies; i.e., it will lower the total amount of heat that recirculates within the data center.

Overall goodness value used by our algorithm is defined as:

$$g_i = \alpha \times gs_i + \beta \times gt_i, \quad g_i \in [0, 1] \tag{23}$$

where α and β are constant ranging from 0 to 1 and $\alpha + \beta = 1$. The goodness function given in Eq. 23 strongly reflects the target objectives of the given problem. The quality of a solution can also be estimated by summing up the goodness of all the VMs.

5.1.2 Selection

In this step, elements are selected for relocation. Elements with lesser *goodness* values have a higher probability of getting selected for re-allocation. This step divides Φ' into two disjoint sets; a set V_s of selected elements and a partial solution Φ_{p} containing rest of the elements of the solution Φ'. The selection operator has a non-deterministic nature, i.e., an individual with a high goodness (close to one) still has a nonzero probability of being assigned to the selection set V_s. It is this element of non-determinism that makes SimE capable of escaping local minima. The Bias value (B) is used to deflate or inflate the goodness of elements. In many cases, a value of $B = 0$ would be a reasonable choice, as in our case (Sait and Youssef 1999).

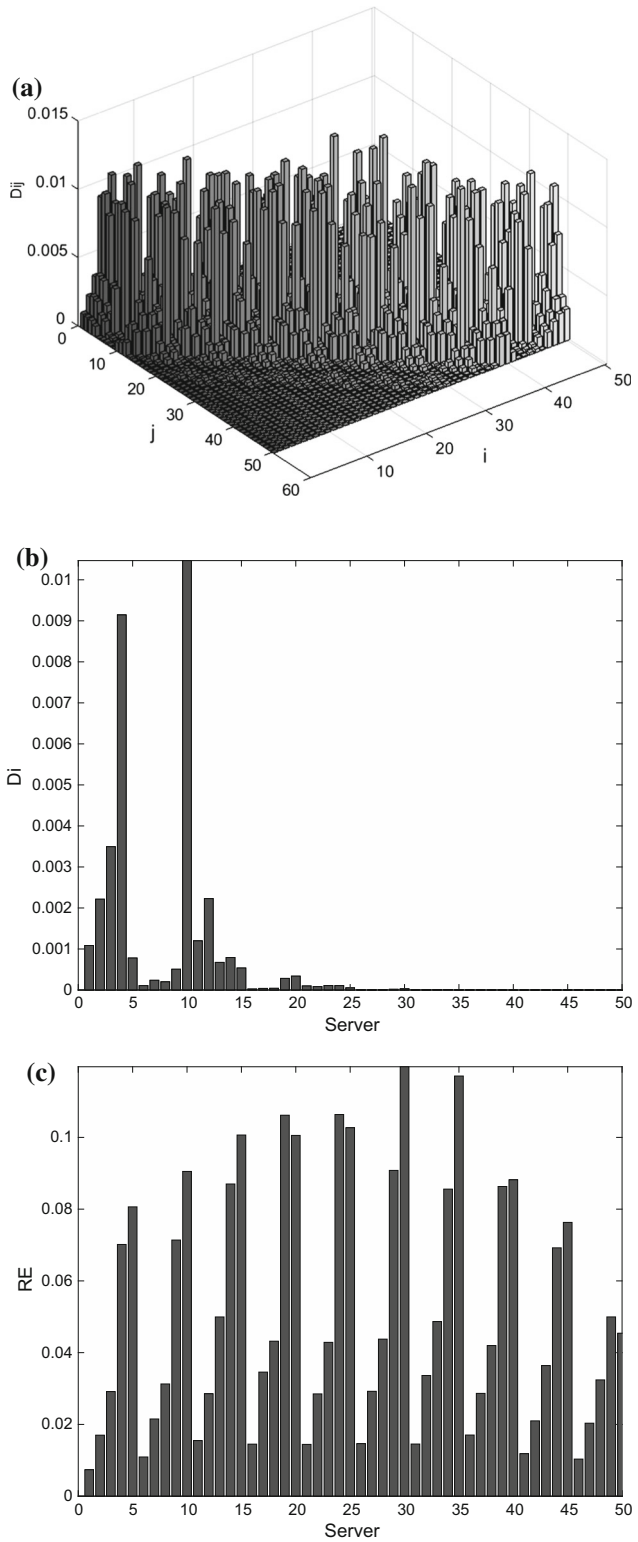


Fig. 7 **a** A sample distribution matrix for fifty servers, **b** a distribution vector for 1st server, **c** recirculation effect (RE) values of 50 servers

ALGORITHM
Simulated_Evolution $(V,\ Stopping-criteria)$;
/* $\mathbf{\Phi_i}$: Initial Solution; */
/* $\mathbf{\Phi_p}$: Partial Solution; */
/* $\mathbf{\Phi'}$: New Solution; */
/* $\mathbf{V}$: Set of all VMs, where $|V| = n$; */
/* $\mathbf{V_s}$: Selected VMs for re-allocation; */
/* $\mathbf{P_a}$: Active PMs in Φ_p; */
/* $\mathbf{B}$: Selection bias; */
/* **maxSelection**: Upper limit of the selection set size; */
INITIALIZATION;
  $\Phi_i = initial\_placement(V)$;
  $\Phi' = \Phi_i$;
Repeat
EVALUATION:
  ForEach $v_i \in V$ **Do**
    $g_i = Evaluate(v_i)$; /* *Evaluate goodness value* */
  EndForEach;
SELECTION:
  $\Phi_p = \Phi'$;
  $count = \mathbf{0}$;
  ForEach $v_i \in V$ **Do**
  If $(Random \leq (1 - g_i + B)) \wedge (count \leq maxSelection)$
    $V_s = V_s \cup \{v_i\}$;
    $\Phi_p = \Phi_p - \{v_i\}$;
    $count = count + 1$;
  End If;
  EndForEach;

ALLOCATION:
Sort the VMs in set V_s *based on their resource demand* ;
  ForEach $v_i \in V_s$ **Do**
    Allocate(v_i ,Φ_p); /* *using First Fit Strategy* */
  EndForEach;
    $\Phi' = \Phi_p$;
Until *Stopping-criterion is satisfied*;
Return (*BestSolution*);
End *Simulated_Evolution*.

Fig. 8 Simulated evolution algorithm for VM assignment

5.1.3 Allocation

The allocation step takes the elements of set V_s and the partial solution Φ_p and generates a complete new solution Φ' with the elements of set V_s mutated according to allocation strategy. The goal of *Allocation* strategy is to favor improvements over the previous iteration, without being too greedy (Sait and Youssef 1999). The design of allocation strategy is problem specific. Just like in the design of goodness function, the choice of allocation strategy also requires ingenuity on the part of the designer. In this work we adopted a variant of FFD heuristic as our allocation strategy. The VMs selected during the *selection* step are sorted in decreasing order of their request sizes (Rv_i) computed using Eq. (24).

$$Rv_i = \left(v_i^c\right)^2 + \left(v_i^m\right)^2. \tag{24}$$

Subsequently, First Fit algorithm is applied (Fig. 8) to generate the new solution Φ'. This algorithm attempts to assign the selected VMs to the servers with low RE value.

6 Experimental results

In this section we provide performance evaluation of our proposed approach with those in the literature. We compare it with the well-known iterative heuristics, Simulated Annealing (SA) and Tabu search (TS), and with two popular constructive algorithms, improved version of classic First Fit Decreasing (FFD_{imp}) and Least-Loaded (LL_{imp}) algorithms proposed by Ajiro and Tanaka (2007).

6.1 Baseline algorithms

FFD and its variants are real-world VM allocation policies available in the literature. OpenNebula and Eucalyptus are two popular cloud platforms. The allocation policies available on these open platforms are Packing, Load-aware, and GREEDY which are different variants of FFD (Helion Eucalyptus Docs Team 2015; OpenNebula 2015). Ajiro and Tanaka (2007) suggested improvements to the classical FFD and Least-Loaded (LL) algorithms for reducing computational power by minimizing number of servers. Improved FFD (FFD_{imp}) and improved LL (LL_{imp}) are different from their conventional counterparts in the sense that they seek near-optimal assignment in multiple passes. These improved versions provide better quality assignments than that of their single-pass implementations. This improvement, however, comes at the expense of increased run time (Ajiro and Tanaka 2007).

Simulated Annealing (SA) and Tabu search (TS) are well-established non-deterministic heuristics (Nahar et al. 1989). They have been adopted to solve various combinatorial optimization problems. SA is inspired from annealing of the metals where good solutions are preferred with time, whereas TS uses *adaptive* (flexible) memory for hill climbing (Sait and Youssef 1999).

Simulated Annealing (SA) is a well-established non-deterministic heuristic (Nahar et al. 1989). It has been adopted to solve various combinatorial optimization problems. One typical feature of SA is that it accepts all the solutions with improved cost like a greedy algorithm, but it also, to a limited extent, accepts changes which lead to inferior solutions. This feature gives SA the hill climbing capability that allows it to escape from the local optimal solutions initially and reach a more optimal solution at the end of the search. The odds of accepting inferior rated solutions are large in the beginning; but as the search progress, fewer bad solutions are accepted and finally only solutions with improved cost are accepted. SA produces high-quality solution regardless of the initial configuration. It is robust, effective, and easy to implement (Sait and Youssef 1994, 1999). We are using a modified Simulated Annealing called Simulated Quenching (SQ) (Ingber 1993). SQ solution methodology resembles the cooling process of molten metals through annealing. The algorithm and the analogy of the technique remain the same as that of SA except for the annealing schedule. SQ uses an exponential schedule.

$$T_{k+1} = \alpha T_k$$

where α is the cooling factor. The *Metropolis* routine is invoked after updating (lowering) temperature T. The annealing procedure halts when *Time* exceeds the allowed time. The *Neighbor* function perturbs the current assignment by randomly assigning a VM to a server. It is ensured that the new assignment is feasible.

Tabu search (TS) is a general iterative heuristic introduced by Glover (1989, 1990) for solving combinatorial optimization problems. Tabu search is a generalization of local search. This scheme works by moving from one solution to another in a hill-climbing fashion. Unlike local search which stops when no improved new solution is found in the current neighborhood, Tabu search continues the search from the best solution in the neighborhood even if it is worse than the current solution. One of its features is its systematic use of *adaptive* (flexible) memory. Tabu search differs from genetic algorithm which are "memoryless," and also from branch-and-bound, A* search, etc., which are rigid memory approaches. The initial assignment matrix X generated by randomly assigning VMs to the servers using admission control. As search proceeds, neighbor assignments are generated moving a VM to a server. The selection of VM and server is done randomly. Maintaining a Tabu List is an important step in TS. Each entry in the Tabu List contains the following information or attributes:

- VM that was selected for the move,
- Server *from* where VM is being moved, and
- Server *to* where VM is being moved.

After a successful move, the inverse of the move is stored in Tabu List. Every new move is checked against the moves stored in the Tabu List to make sure no reverse moves are being made, and heuristics can escape local optimal. However, Tabu List moves are ignored if the new assignment is better than the best obtained thus far (aspiration criterion).

6.2 Simulation setup

Programs for the proposed SimE, SA, TS, FFD_{imp}, and LL_{imp} heuristics are coded in MATLAB. The simulations are run

on a computer equipped with Intel core^{TM} i3 with 2.4 GHz CPU and 6 GB RAM.

For simplicity, we assume a homogeneous hardware environment. All servers have the same power consumption and computing capability. We are using $\bar{p}_{\text{busy}} = 215\,\text{W}$ and $\bar{p}_{\text{idle}} = 162\,\text{W}$ as reported by Gao et al. (2013) through experiments conducted on a Dell^{TM} server. We are using 150 servers and 1 CRAC unit. The flow rate (f_i) of each server's fan is set to 8.0 m^3/s.

SA has four parameters which need to be tuned carefully (Sait and Youssef 1999). After trial runs, appropriate values of these parameters are found to be $T_0 = 800$, $\alpha = .98$, $\beta = 1.1$, and $M = 9$. Similarly, parameters of TS (Sait and Youssef 1999) are set as TabuListSize = 10 and MaxTrialAssignments = 15. SimE is set to stop after 30 iterations if there is no significant improvement in cost and maxSelection is set to 40% of total VMs. For our goodness estimator α and β both are set to 0.5.

6.2.1 Data for simulation

The problem instances were a set of two resource demand vectors representing the CPU and memory utilization of VMs. Servers were assumed to be identical, that is, all PMs have the same resource capacity fixed at 90% although the proposed approach is equally applicable for the heterogeneous case. Due to non-deterministic behavior, average of results obtained from 20 independent runs is reported.

It was observed that for some types data (workloads) it is easy to reach near-optimal solutions. This is similar to the phase transitions observed in physical systems (Hartmann and Weigt 2006). To cover a wide range of possible workloads, we generated problem instances with two different average resource values and several correlations of CPU and memory utilization, employing the method proposed by Ajiro and Tanaka (2007). The pseudocode for this is given in Fig. 9. In our experiment, we used two kinds of average values and five different probabilities. We set both v^c and v^m to 25%, and then to 45%. We set P to 0.00, 0.25, 0.50, 0.75, and 1.0 to get strong-negative, weak-negative, no, weak-positive, and strong-positive correlations. We also tested for 50% loadings as well as 70% loadings. Average resource value is related to the size of a single request; whereas, loading is the ratio of overall requirements of all VMs, which need to be assigned, to the sum of all available resources. We define loading as:

$$\text{Loading} = 0.5 \times \frac{\sum_{i=1}^{n} \left(v_i^c + v_i^m\right)}{\sum_{j=1}^{m} \left(M_i^c + M_i^m\right)} \tag{25}$$

where v_i^c and v_i^m are the CPU and memory requirements of ith VM, and M_i^c and M_i^m are the CPU and memory capacities of ith server. The 0.5 factor is used because two elements, CPU and memory, are considered. Please note that

```
for i = 1 to n
  v_i^c ← rand(2v̄^c)
  v_i^m ← rand(v̄^m)
  r = rand(0, 1)
  if (r < P and v_i^c ≥ v̄^c) or (r ≥ P and v_i^c < v̄^c)
    v_i^m ← v_i^m + v̄^m
  end if
end for
```

Fig. 9 Pseudocode to generate different problem instances with certain correlations

the requirements of the ith VM are fixed. In our work, it is assumed that the loading is continuous and constant throughout the assignment process. We are especially considering environments, such as HPC data centers, where tasks requirements remain almost fixed and take days to finish. Whenever utilization (loading) is changed, VMs are removed and/or introduced, and assignment algorithm needs to be run again. Our approach is also applicable to situations when a data center starts its operation after a maintenance state, or when a data center optimizer/controller takes a decision at the backend.

6.3 Results

In this section we highlight and discuss the results of our experiments.

From Table 2 the following observations can be made:

- For all algorithms, total power consumption decreases by decreasing the loading from 70 to 50%. This is because it is difficult to avoid hot spots with increased loading.
- Similarly, total power decreases by decreasing average resource value from 45 to 25%.
- The timing performance of SA, FFD_{imp} and LL_{imp} strongly depends on the correlation between CPU and memory utilization. On the other hand, execution time of SimE and TS varies slightly across different correlation.
- TS gives better results as compared to SA but requires more CPU time.
- In each case, SimE outperforms other algorithms while requiring less execution time.

Breakdowns of total power P_{total} into computational power P_{CN} and cooling power P_{AC} at various correlations are shown in Fig. 10. It can be observed that, for all correlations, total power consumption of SimE is the lowest. Similarly, CPU time at various correlation is illustrated in Fig. 11.

SA and TS perform better than FFD_{imp} and LL_{imp}. FFD_{imp} and LL_{imp} attempt to assign VMs to a minimum number of servers. They do not consider heat recirculation in the data center; therefore, resultant assignment reduces computational power but creates hot spots due to higher utilization

Table 2 Comparison of the SimE, SA, and TS with other techniques

Reference value	Corr.	Algorithm	50% Loading				70% Loading			
			P_{total}	max(T_{in})	s	T (s)	P_{total}	max(T_{in})	s	T (s)
$\overline{v^c} = \overline{v^m} = 25\%$	Strong +ve	FFD_{imp}	43.09	43.68	89.00	28.80	59.20	43.69	122.00	33.52
		LL_{imp}	41.11	43.26	87.00	10.90	57.94	43.66	119.00	8.85
		SA	33.53	37.28	102.75	215.31	55.51	41.28	137.10	80.88
		TS	30.65	36.13	97.40	181.80	54.25	41.97	125.70	237.60
		SimE	25.40	31.35	90.45	4.55	49.42	40.49	123.65	8.03
	Weak +ve	FFD_{imp}	43.66	43.69	90.00	45.21	60.96	43.68	126.00	51.41
		LL_{imp}	41.07	43.45	85.00	6.91	58.53	43.75	119.00	6.97
		SA	34.09	37.39	103.80	215.61	56.36	41.37	138.10	80.87
		TS	30.80	36.09	97.80	178.92	53.52	41.50	127.50	381.23
		SimE	24.24	30.57	87.85	3.93	44.69	39.44	118.70	7.31
	Zero	FFD_{imp}	47.60	43.69	100.00	47.71	64.88	43.79	135.00	84.26
		LL_{imp}	42.71	43.20	91.00	10.52	59.68	43.64	123.00	14.54
		SA	36.41	38.08	108.00	125.79	57.71	41.74	138.10	80.64
		TS	32.46	36.43	102.20	183.50	55.43	41.90	129.00	228.70
		SimE	23.98	30.51	87.15	3.69	43.83	39.08	117.95	6.94
	Weak −ve	FFD_{imp}	48.47	43.68	102.00	33.14	67.70	43.75	143.00	121.74
		LL_{imp}	42.93	43.23	91.00	6.20	60.01	43.56	125.00	22.06
		SA	36.69	38.08	108.55	78.06	56.28	41.30	138.70	83.62
		TS	33.14	36.55	104.00	186.72	50.42	40.05	132.50	257.67
		SimE	23.14	30.29	84.05	3.06	39.44	37.17	116.05	6.53
	Strong −ve	FFD_{imp}	51.43	43.68	109.00	43.26	71.32	43.82	150.00	134.78
		LL_{imp}	44.02	43.26	93.00	4.94	61.62	43.71	126.00	9.89
		SA	36.44	37.95	107.80	81.32	58.67	41.67	140.40	80.66
		TS	33.14	36.55	104.00	186.72	52.02	40.18	135.00	259.56
		SimE	23.36	30.28	85.55	3.02	39.82	37.39	116.70	6.57
$\overline{v^c} = \overline{v^m} = 45\%$	Strong +ve	FFD_{imp}	41.30	43.56	86.00	14.33	63.60	43.71	131.00	20.59
		LL_{imp}	39.84	43.17	85.00	6.86	62.42	43.61	129.00	7.37
		SA	27.96	34.43	94.40	84.61	58.60	41.70	139.70	74.00
		TS	27.18	33.65	93.90	168.99	51.74	40.19	133.80	197.60
		SimE	26.54	32.47	90.9	2.28	50.27	39.82	133.0	2.63
	Weak +ve	FFD_{imp}	46.56	43.57	98.00	25.74	67.94	43.33	145.00	35.36
		LL_{imp}	45.25	43.14	98.00	14.68	67.42	43.62	140.00	9.79
		SA	32.61	35.80	105.70	84.22	65.49	42.58	147.40	75.16
		TS	31.61	35.02	105.30	178.48	62.14	42.28	141.50	197.34
		SimE	28.63	34.06	97.40	1.92	60.80	41.65	140.8	4.93
	Zero	FFD_{imp}	47.92	43.57	100.00	19.24	69.75	43.63	145.00	23.11
		LL_{imp}	48.19	43.42	102.00	11.80	69.68	43.53	146.00	10.75
		SA	34.77	36.47	109.25	81.48	67.27	42.76	149.10	75.40
		TS	33.21	35.57	107.60	174.95	65.63	42.58	146.70	217.10
		SimE	30.85	35.10	100.25	2.48	63.91	41.89	145.45	4.60
	Weak −ve	FFD_{imp}	47.27	43.27	103.00	42.04	69.65	43.34	150.00	60.10
		LL_{imp}	46.77	43.47	100.00	17.05	70.07	43.63	147.00	17.46
		SA	33.17	36.05	107.35	81.14	65.58	42.54	148.30	77.40
		TS	32.29	35.44	106.80	168.61	64.74	42.45	147.10	194.34
		SimE	29.05	34.40	97.25	2.46	61.37	42.22	139.85	5.67

Table 2 continued

Reference value	Corr.	Algorithm	50% Loading				70% Loading			
			P_{total}	$\max(T_{\text{in}})$	s	T (s)	P_{total}	$\max(T_{\text{in}})$	s	T (s)
	Strong −ve	FFD_{imp}	49.75	43.26	108.00	45.59	69.48	43.25	150.00	60.33
		LL_{imp}	50.15	43.60	106.00	19.22	71.26	43.62	150.00	19.15
		SA	35.72	36.59	112.75	88.22	67.51	42.87	149.40	87.72
		TS	35.75	36.62	112.70	145.98	68.20	42.98	149.90	228.75
		SimE	22.65	30.24	83.10	2.19	50.24	40.38	127.15	4.69

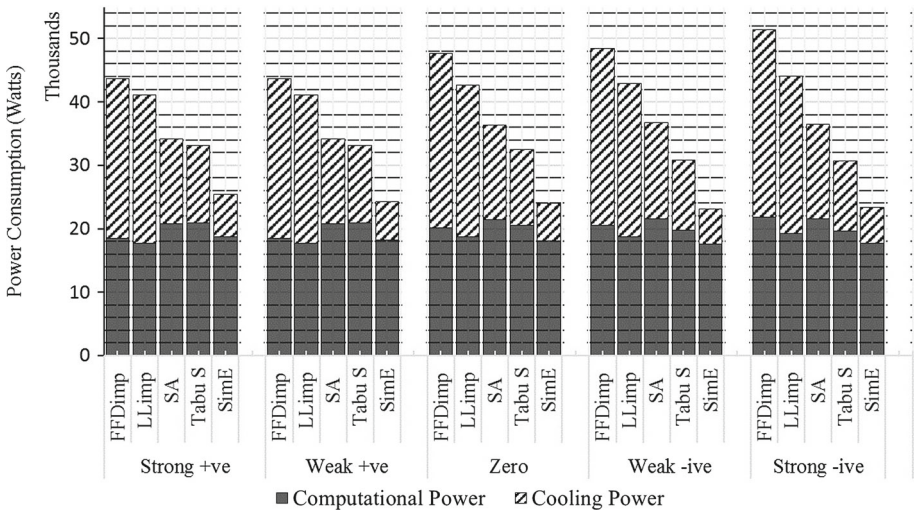


Fig. 10 Power breakdown for different algorithms at various correlations for cases of $\overline{v^c} = \overline{v^m} = 25\%$

per server. On the other hand, SA and TS optimize total power by considering the trade-off between computational and cooling power, but they require significant CPU time. SimE is performing better than TA and SA because SimE uses its goodness function to direct the search, whereas TS and SA use hill climbing, memory, and other features to find the optimal assignment, but they lack intelligent moves. The precise selection of ill-assigned VMs and proper re-allocation plays a key role in improving the solution quality and reducing run time. Although SA, TA, and SimE all are iterative non-deterministic heuristics, SimE is more intelligent and thus requires fewer iterations to converge toward a desirable solution.

Figure 12 shows total power consumption of various schemes at different loadings of the data center. It is reported for 25% reference value and strong-negative correlation. It is evident that SimE is performing better than other schemes for most of the loading span.

Change in total power, max inlet temperature, and number of active servers of SimE with iterations are illustrated in Fig. 13. Without the loss of generality, the case reported has the following values: 50% loading, $R_c = R_m = 25\%$, and strong-negative correlation. However, similar results are obtained for other cases. From graphs in Fig. 13, it is evident that SimE escapes local minima multiple times by making intelligent moves and attempts to reduce total power.

Finally, Fig. 14 shows that the overall average goodness of all movable elements increases with iterations indicating that search in our engineered SimE implementation is intelligently progressing toward a solution where each VM is optimally assigned. Further indication of this is the reduction in the size of the selection set with iterations (Fig. 15).

7 Conclusions and discussion

Computational power and cooling power are significant parts of a data center's operational costs, and they need to be optimize in an integrated fashion. In this work, we presented the engineering of simulated evolution (SimE) for joint optimization of cooling and computational power. We developed a goodness measure that, based on our objectives, enables SimE to intelligently find a near-optimal solution in a short amount of time. We evaluated its performance for a wide range of different problem instances. We showed that our approach is performing better as compared to other schemes published previously.

Modern evolutionary heuristics such as differential evolution (DE), particle swarm optimization (PSO), and Cuckoo Search (CS) are powerful, but implementing them to solve a particular combinatorial optimization problem is relatively very tedious. Further, our initial experiments required larger

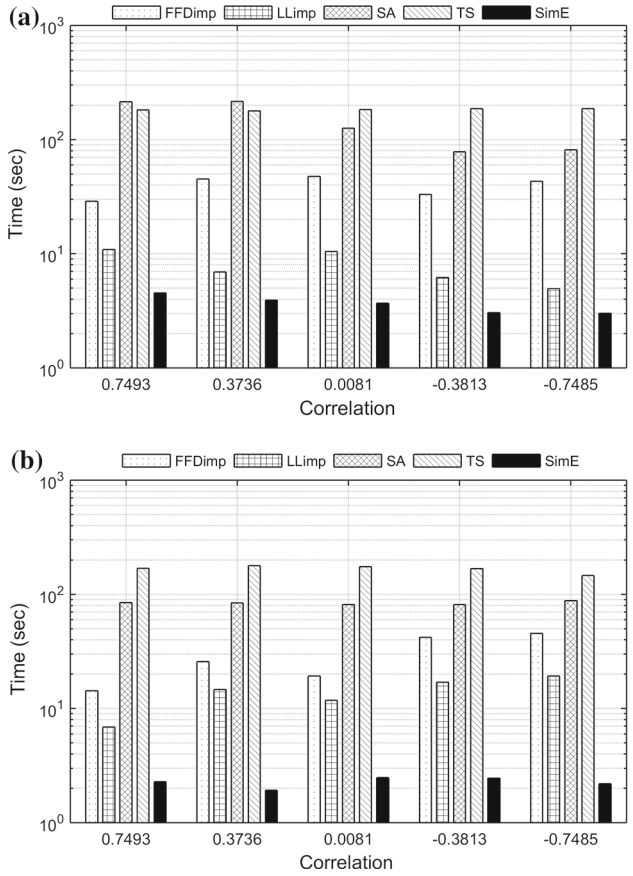


Fig. 11 Run time of FFD_{imp}, LL_{imp}, SA, TS and SimE with 50% loading for cases of **a** $\overline{v^c} = \overline{v^m} = 25\%$ and **b** $\overline{v^c} = \overline{v^m} = 45\%$

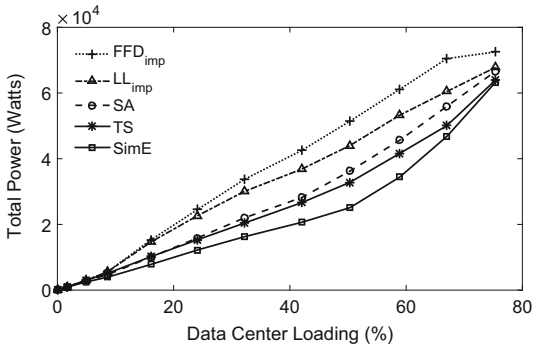


Fig. 12 Comparison of Algorithms for various loadings for cases of $\overline{v^c} = \overline{v^m} = 25\%$

run times; particularly, in the case of DE due to a number of infeasible and inferior solutions generated. The rate of solution improvement is slower in DE due to larger focus on exploration (Zheng et al. 2015). Also, these heuristics do not have a feature that will enable the incorporation of the domain knowledge of the problem being solved. PSO also guides the solutions in search space. However, movements of the solutions are guided by their own best known position in the search space as well as the best known position of

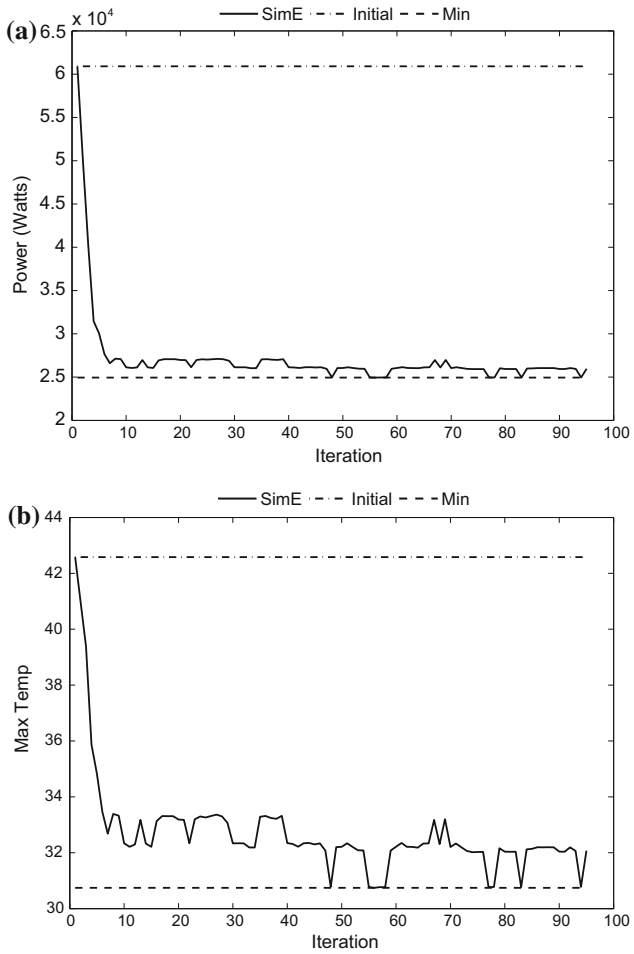


Fig. 13 Change in **a** Total Power consumption P_{total}, **b** max inlet temperature T_{in} with iterations in SimE. **a** SimE: total power versus iterations, **b** SimE: max inlet temperature versus iterations

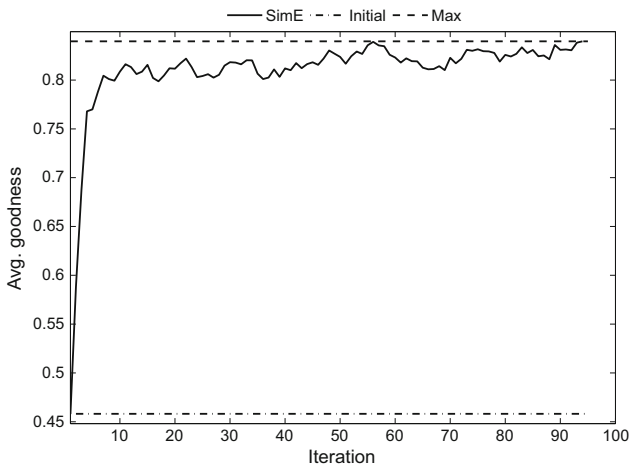


Fig. 14 SimE: change in the average goodness of VMs with iterations

other candidate solutions (Kennedy 1997). PSO does not use domain knowledge to guide the search. It is this powerful feature of SimE that steered the search to obtain excellent

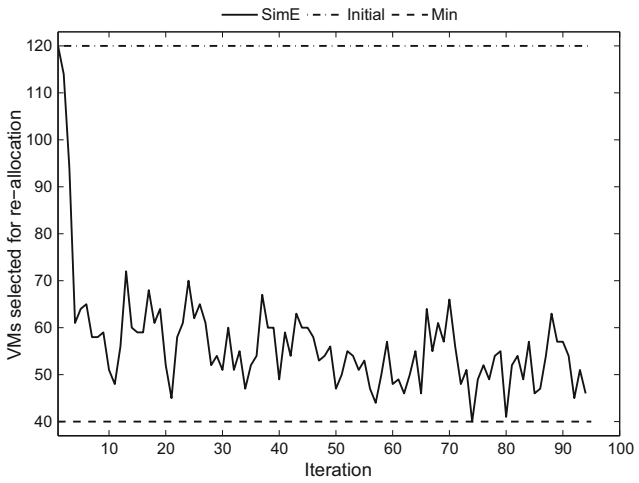


Fig. 15 SimE: change in the number of VMs selected for re-allocation with iterations

solutions in minimal time. It was also shown that the performance of SimE is independent of the correlation between different dimensions of VMs. This feature makes this heuristic desirable for all correlation scenarios.

In this work, we considered the case where all the VM requests are known before placement and the controller allocates them at once, trying to find the optimal allocation in accordance with the objectives and constraints. Such situations arise when a data center starts its operation after a maintenance state or when the data center optimizer/controller takes a decision at the back-end. However, in operational data centers VM requests arrive incrementally over time. In order to address this issue, it is recommended that future studies look into modifications of the algorithm that would work for an online scenario.

Acknowledgements The authors acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for all support. The work was conducted as part of project COE-572132-2.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Ajiro Y, Tanaka A (2007) Improving packing algorithms for server consolidation. In: International of CMG conference, pp 399–406

Al-Qawasmeh AM, Pasricha S, Maciejewski A, Siegel HJ et al (2015) Power and thermal-aware workload allocation in heterogeneous data centers. IEEE Trans Comput 64(2):477–491

Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the art of virtualization. ACM SIGOPS Oper Syst Rev 37(5):164–177

Barroso LA, Hölzle U (2007) The case for energy-proportional computing. IEEE Comput 40(12):33–37

Basmadjian R, Niedermeier F, De Meer H (2012) Modelling and analysing the power consumption of idle servers. In: Sustainable internet and ICT for sustainability (SustainIT). IEEE, pp 1–9

Bohrer P, Elnozahy EN, Keller T, Kistler M, Lefurgy C, McDowell C, Rajamony R (2002) The case for power management in web servers. In: Power aware computing. Springer, Berlin, pp 261–289

Brown DJ, Reams C (2010) Toward energy-efficient computing. Commun ACM 53(3):50–58

Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP (2001) Managing energy and server resources in hosting centers. In: ACM SIGOPS operating systems review, vol 35. ACM, pp 103–116

Doddavula SK, Kaushik M, Jain A (2011) Implementation of a fast vector packing algorithm and its application for server consolidation. In: IEEE third international conference on cloud computing technology and science (CloudCom). IEEE, pp 332–339

Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: ACM SIGARCH Computer Architecture News, vol 35. ACM, pp 13–23

Feng X, Ge R, Cameron KW (2005) Power and energy profiling of scientific applications on distributed systems. In: Parallel and distributed processing symposium, proceedings. 19th IEEE International. IEEE, pp 34–34

Filani D, He J, Gao S, Rajappa M, Kumar A, Shah P, Nagappan R (2008) Dynamic data center power management: trends, issues, and solutions. Intel Technol J 12(1):59–67

Fu Y, Lu C, Wang H (2010) Robust control-theoretic thermal balancing for server clusters. In: IEEE international symposium on parallel & distributed processing (IPDPS). IEEE, pp 1–11

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. J Comput Syst Sci 79(8):1230–1242

Glover F (1989) Tabu search-part I. ORSA J Comput 1(3):190–206

Glover F (1990) Tabu search-part II. ORSA J Comput 2(1):4–32

Hamilton J (2009) Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services. In: Conference on innovative data systems research (CIDR)

Hartmann AK, Weigt M (2006) Phase transitions in combinatorial optimization problems: basics, algorithms and statistical mechanics. Wiley, New York

Helion Eucalyptus Docs Team (2015) Eucalyptus Scheduling Policies. https://www.eucalyptus.com/docs/eucalyptus/4.0.2/install-guide/sched_pol.html. Accessed 5 Nov 2015

Ingber L (1993) Simulated annealing: practice versus theory. Math Comput Model 18(11):29–57

Kennedy J (1997) The particle swarm: social adaptation of knowledge. In: Evolutionary computation. IEEE International conference on. IEEE, pp 303–308

Khan JA, Sait SM, Minhas MR (2002) Fuzzy biasless simulated evolution for multiobjective vlsi placement. In: Evolutionary computation. CEC'02. Proceedings of the 2002 congress on, vol 2. IEEE, pp 1642–1647

Kling RM, Banerjee P (1987) Esp: a new standard cell placement package using simulated evolution. In: Proceedings of the 24th ACM/IEEE design automation conference. ACM, pp 60–66

Kramer HH, Petrucci V, Subramanian A, Uchoa E (2012) A column generation approach for power-aware optimization of virtualized heterogeneous server clusters. Comput Ind Eng 63(3):652–662

Moore JD, Chase JS, Ranganathan P, Sharma RK (2005) Making scheduling "cool": temperature-aware workload placement in data centers. In: USENIX annual technical conference, General Track, pp 61–75

Mukherjee T, Tang Q, Ziesman C, Gupta SK, Cayton P (2007) Software architecture for dynamic thermal management in datacenters. In: 2nd International conference on communication systems software and middleware. IEEE, pp 1–11

Nahar S, Sahni S, Shragowitz E (1989) Simulated annealing and combinatorial optimization. Int J Comput-Aid VLSI Des 1(1):1–23

OpenNebula (2015) OpenNebula Scheduling Policies. http://archives.opennebula.org/documentation:rel4.4:schg. Accessed 5 Nov 2015

Patel CD, Bash CE, Sharma R, Beitelmal M, Friedrich R (2003) Smart cooling of data centers. In: ASME 2003 international electronic packaging technical conference and exhibition. American Society of Mechanical Engineers, pp 129–137

Pinheiro E, Bianchini R, Carrera EV, 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. Barcelona, pp 182–195

Sait SM, Bala A, El-Maleh AH (2016) Cuckoo search based resource optimization of datacenters. Appl Intell. doi:10.1007/s10489-015-0710-x

Sait SM, Shahid KS (2015) Engineering simulated evolution for virtual machine assignment problem. Appl Intell 43(2):1–12

Sait SM, Youssef H (1994) VLSI physical design automation: theory and practice. McGraw-Hill Inc, New York

Sait SM, Youssef H (1999) Iterative computer algorithms with applications in engineering: solving combinatorial optimization problems. IEEE Computer Society Press, Los Alamitos, CA, USA

Sullivan RF (2000) Alternating cold and hot aisles provides more reliable cooling for server farms. Uptime Institute, Santa Fe, New Mexico

Systems IT Governance Research Team (2008) In: Green IT: reality, benefits and best practices. IT Governance, Ely,UK

Tang Q, Gupta SK, Varsamopoulos G (2008) Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach. IEEE Trans Parallel Distrib Syst 19(11):1458–1472

Tang Q, Mukherjee T, Gupta SK, Cayton P (2006) Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In: Fourth international conference on intelligent sensing and information processing (ICISIP). IEEE, pp 203–208

The Climate Group on behalf of the Global eSustainability Initiative (GeSI) (2008) SMART 2020: enabling the low carbon economy in the information age. Climate Group

Wang L, von Laszewski G, Dayal J, Furlani TR (2009) Thermal aware workload scheduling with backfilling for green data centers. In: Performance computing and communications conference (IPCCC), IEEE 28th International. IEEE, pp 289–296

Wang X, Wang Y, Cui Y (2016) An energy-aware bi-level optimization model for multi-job scheduling problems under cloud computing. Soft Comput 20(1):303–317

Xu B, Peng Z, Xiao F, Gates AM, Yu JP (2015) Dynamic deployment of virtual machines in cloud computing using multi-objective optimization. Soft Comput 19(8):2265–2273

Youssef H, Sait SM, Khan SA (2002) Topology design of switched enterprise networks using a fuzzy simulated evolution algorithm. Eng Appl Artif Intell 15(3):327–340

Zheng F, Zecchin AC, Simpson AR (2015) Investigating the run-time searching behavior of the differential evolution algorithm applied to water distribution system optimization. Environ Model Softw 69:292–307