Future Generation Computer Systems 29 (2013) 1901–1908

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

An approximate ϵ-constraint method for a multi-objective job scheduling in the cloud

L. Grandinetti[a,*], O. Pisacane[b], M. Sheikhalishahi[a]

[a] Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Via P. Bucci, 41C, Arcavacata di Rende (CS), Italy

[b] Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Via B. Bianche, 12, Ancona (AN), Italy

HIGHLIGHTS

- We formulate the multi-objective off-line job scheduling problem in the cloud.
- We optimize the makespan, the total average waiting time and the used hosts.
- We generate a set of test instances suitable for the problem.
- We define an approximate ϵ-constraint method.
- We compare the proposed solution approach with the weighted sum method.

ARTICLE INFO

Article history:
Received 13 July 2012
Received in revised form
26 March 2013
Accepted 8 April 2013
Available online 9 May 2013

Keywords:
Jobs scheduling
Operations management
Cloud computing
ϵ-constraint method
Multi-objective optimization

ABSTRACT

Cloud computing is a hybrid model that provides both hardware and software resources through computer networks. Data services (hardware) together with their functionalities (software) are hosted on web servers rather than on single computers connected by networks. Through a device (e.g., either a computer or a smartphone), a browser and an Internet connection, each user accesses a cloud platform and asks for specific services. For example, a user can ask for executing some applications (jobs) on the machines (hosts) of a cloud infrastructure. Therefore, it becomes significant to provide optimized job scheduling approaches suitable to balance the workload distribution among hosts of the platform.

In this paper, a multi-objective mathematical formulation of the job scheduling problem in a homogeneous cloud computing platform is proposed in order to optimize the total average waiting time of the jobs, the average waiting time of the jobs in the longest working schedule (such as the makespan) and the required number of hosts. The proposed approach is based on an approximate ϵ-constraint method, tested on a set of instances and compared with the weighted sum (WS) method.

The computational results highlight that our approach outperforms the WS method in terms of a number of non-dominated solutions.

1. Introduction

Cloud computing is a revolutionary paradigm suitable to change the way of accessing both hardware and software in order to produce, price, provide and deliver services and computational resources to users. Users can run their applications (*jobs*) without paying for software licenses, using well equipped machines (*hosts*) and high performance computational resources.

This paper addresses a multi-objective job scheduling problem in a homogeneous cloud infrastructure considering the minimization of the total average waiting time of the jobs, of the total waiting time of the jobs belonging to the longest working schedule (*makespan*) and the number of used hosts. It takes into account an *off-line* job scheduling scenario and, therefore, the number of jobs to run and their resource requirements are known a-priori. The main contributions are as follows:

- a multi-objective formulation of the off-line job scheduling problem in a homogeneous cloud computing platform;
- an approximate ϵ-constraint method for solving the problem;
- a detailed experimental analysis for evaluating the quality of the proposed approach.

With reference to the last contribution, first we implement an instance generator in order to determine a set of problems considered during the experimental phase. Then, we implement an alternative solution approach based on the *weighted sum* (WS) method. Finally, we compare the two approaches on the set of generated instances.

* Corresponding author. Tel.: +39 3351244747.
E-mail addresses: lugran@unical.it (L. Grandinetti), pisacane@dii.univpm.it (O. Pisacane), alishahi@unical.it (M. Sheikhalishahi).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.04.023

This paper is organized as follows: Section 2 reviews some significant literary contributions, Section 3 provides a high level description of the problem, Section 4 describes the multi-objective mathematical formulation of the problem. Sections 4.1–4.3 detail the two solution approaches taken into account, while Section 5 describes the generated scenarios and discusses the computational results. Finally, Section 6 concludes the work and suggests some future developments.

2. Related work

This section aims at describing some significant literary contributions with reference to the job scheduling problem. Therefore, it is organized as follows: firstly, some of the more significant job scheduling algorithms in computing systems are described; secondly, several multi-objective job scheduling methods are analyzed and finally, some literary contributions based on the ϵ-constraint method are summarized.

In the job scheduling problems in computer systems, the number of machines can be assumed finite and fixed. However, this assumption does not hold for the number of jobs. The scheduling problems in computer systems also differ from system to system (e.g., manufacturing and project planning). For example, the jobs could be dynamic (i.e., their arrival time is not known a-priori as well as their characteristics and duration). Moreover, in computer systems, the aim of these problems is to provide a better resource utilization satisfying the job requirements. A very popular research topic is the *parallel job scheduling*. There are many different ways for scheduling parallel jobs and threads that make them up [1]. However, only a few mechanisms are used in practice and studied in detail. Two approaches that have dominated over the last decade are the *Backfilling* and the *Gang Scheduling*. In particular, the *Backfilling*, introduced in [2], aims at balancing the resource utilization and at maintaining a *First Come First Served* (FCFS) order. It also allows small jobs to move ahead and run on processors that would otherwise remain idle. However, this is subject to some restrictions so that the situations in which the FCFS order is completely violated and some jobs are never run (i.e., *starvation* phenomenon) are avoided. In particular, a reservation for some future times is usually given to jobs that need to wait and its use was included in several early *Batch Schedulers* [3].

In [4], a decentralized dynamic scheduling approach (Community Aware Scheduling Algorithm—CASA) is proposed. CASA consists of a two-phase scheduling solution approach and of some heuristics to achieve optimized performances in the grid/cloud platform. The authors show that it yields a 30%–61% better average job slowdown if compared to the centralized scheduling scheme based on the BestFit meta-scheduling policy. It also yields a 68%–86% shorter average job waiting time if compared to a decentralized scheduling approach without requiring detailed real-time processing information from nodes.

In [5], a hierarchical framework and a job scheduling algorithm (*Hierarchical Load Balanced Algorithm*—(HLBA)) are proposed with reference to a grid platform. The system load is used to determine a balance threshold. The scheduler dynamically adapts the balance threshold according to the system load changes. The aim of the proposed scheduling algorithm is to balance the system load and minimize the *makespan* of jobs. In [6], the authors analyze and investigate the effectiveness of rescheduling using cloud resources in order to increase the job completion reliability. The jobs are scheduled using grid resources and then, cloud resources are used only for rescheduling to cope with a delay in job completion. The computational results demonstrate that the proposed rescheduling guarantees a delay reduction in job completion.

In [7], a credit based scheduling is used to evaluate the entire group of tasks in the job queue and to find their minimal completion time. In [8], a mathematical model for distributing workload among a minimum number of servers is proposed as a set partitioning formulation and two solution approaches are described. In the former, a set of candidate blocks are generated and then composed together for having schedules through an integer programming problem. Instead, in the latter, the set partitioning problem is solved by performing a column generation technique. After performing a test phase, the authors conclude that the second method outperforms the first one.

Moreover, due to the high dynamism of cloud environments that leads to a time-varying resource utilization, cloud providers can potentially accommodate secondary jobs with the remaining resource. In [9], the problem of secondary job scheduling with deadlines under time-varying resource capacity is taken into consideration. However, the scheduling scheme used on many distributed memory parallel supercomputers is *variable partitioning*. In this context, each job receives a machine partition with its desired number of processors [1]. Such partitions are allocated in an FCFS manner to incoming jobs.

While the job scheduling problem in distributed computing has attracted a lot of interest, less attention has been given to the multi-criteria version. In fact, there are a few works to address this special issue. In [10], some novel taxonomies of the multi-criteria grid workflow scheduling problem are proposed, mainly considering workflow, resource and task model, scheduling criteria and process. In [11], a multi-cost scheme of polynomial complexity is proposed in order to perform reservations and select computational resources to execute tasks. This scheme is also used to determine the path to route input data. The authors also describe some multi-cost algorithms with the aim of performing more advance reservations and finding the starting times for data transmission and tasks' execution. In [12], a modular broker architecture is proposed and described. It works with different scheduling strategies in order to optimally deploy virtual services across multiple clouds. These scheduling schemes are mainly based on different criteria to be optimized (either cost or performance optimization). Some user constraints to be taken into consideration (budget, performance, instance types, placement, reallocation or load balancing constraints) together with certain environmental conditions (static vs. dynamic conditions, instance prices, instance types, service workload, etc.). In [13], a particle swarm optimization based heuristic is designed and proposed in order to schedule applications on cloud resources, taking into account both computation cost and data transmission cost.

Nonetheless, one important criticism to be addressed for multi-objective models mainly concerns the definition of an efficient solution approach. In the literature, several alternative solution approaches for multi-objective optimization problems have been proposed. In particular,

- the *weighted global criterion method* in which the objectives are jointly optimized using a weighted function (usually a weighted exponential sum);
- the *weighted sum method* in which the objectives are summed in one function by introducing appropriate weights;
- the *lexicographic method* in which the objectives are arranged in order of importance and relevance.

For an exhaustive study about the multi-objective optimization methods, the reader may refer to [14]. However, an innovative solution approach, the ϵ-constraint method, has recently been introduced in [15]. In [16], a combined procedure of a previously developed single-objective optimization approach together with the ϵ-constraint method is proposed in order to provide an approximation of the Pareto front in multi-objective optimization. In [15], an exact ϵ-constraint method for the bi-objective combinatorial optimization problems with integer objective values is described. The authors also show how the Pareto front can be efficiently detected by applying this method and propose its application to the *Traveling Salesman Problem with Profits*. In [17], some methods for approximating the Pareto set of multi-objective optimization problems are proposed by solving a sequence of constrained single-objective problems. Moreover, the authors also design an adaptive scheme to generate appropriate constraints during the execution. In [18], the authors adopt an ϵ-constraint method to solve the manufacturing cell formation problem minimizing both the inter-cellular movements and the workload unbalance. In [19], the multi-objective congestion management is addressed optimizing the congestion management cost, the voltage security and the dynamic security. Then, the authors describe an ϵ-constraint based solution approach for the problem under examination. The computational results, compared to the one detected by the classical single- and multi-objective approaches, highlight the efficiency of the proposed algorithm.

Other realistic scenarios concern the electricity market. In [20], a multi-objective model is proposed taking under control both the voltage and the dynamic security aspects of the power systems. The proposed method incorporates the lexicographic optimization with the ϵ-constraint method.

However, a recent contribution has been proposed in [21]. The authors describe an approximate ϵ-constraint based heuristic to solve the multi-objective *Undirected Capacitated Arc Routing Problem*. The computational results highlight that this method detects high quality Pareto front approximations. In this paper, this heuristic method has been exploited to solve the multi-objective job scheduling problem in a homogeneous cloud computing platform.

3. Job scheduling in cloud computing platforms

This section describes the problem introducing assumptions and notations.

A scheduling problem is usually modeled as either an optimization or a decision problem [22]. Usually, the former is harder than the latter and aims at identifying the best solutions (i.e., the ones that optimize a specific goal) among a given set of feasible solutions. On the other hand, the decision problem aims at establishing whether a given feasible solution achieves or not its objective (i.e., *yes* or *no* answer). In this paper, the job scheduling problem is formulated as a decision problem with reference to three specific objectives.

The job scheduling problem on a cloud computing platform is usually studied at two different levels: at a *user-level* and at a *system-level*. The user-level aims at managing the services' provision among the providers and the customers while the system-level aims at managing the resources within a data center. In this last case, the aim is to assign the user jobs to the hosts of the platform considering some specific impacts on the data center.

In this paper, the following notations are used:

- J is the set of n jobs to execute;
- H is the set of m homogeneous hosts of the platform;
- w_j is the workload of job $j \in J$ in terms of amount of resource requirements;
- u_h is the capacity of the host $h \in H$ in terms of amount of available computational resources. In particular, since the platform is assumed to be homogeneous, it will be denoted by u for all hosts;
- t_j is the processing time required by the job $j \in J$ on each host.

It is worth noting that both the job workload and the host capacity are expressed in the same unit and a resource (either provided or required) can be represented by CPU time, memory, storage and/or network bandwidth. Moreover, by $\bar{H} \subseteq H$ the set of allocated hosts in the final solution is denoted and $S_{\bar{h}}$ represents a working schedule on the host $\bar{h} \in \bar{H}$. Therefore, $S_{\bar{h}}$ can be seen as the working sequence of the jobs of the sub-set $J_{\bar{h}}$ assigned to the host $\bar{h}$.

Moreover, a feasible schedule $\widetilde{S}_{\bar{h}}$ is defined as follows:

$$\widetilde{S}_{\bar{h}} = \left\{ \bar{j} \in J_{\bar{h}} : \sum_{\bar{j} \in J_{\bar{h}}} w_{\bar{j}} \le u \right\}. \tag{1}$$

Among all the working sequences, only the feasible ones will be considered and they will populate the set Ω formally defined by

$$\Omega = \{\widetilde{S}_{\bar{h}} \forall \bar{h} \in \bar{H}\}. \tag{2}$$

This problem can be seen as a special case of the routing application described in [23,21]. In fact, the hosts of the cloud platform represent the vehicles while the jobs are the customers. All the feasible routes are here replaced by all the feasible working schedules.

In a high level multi-objective formulation, the objective function $f(x)$ is a β-vector (where β denotes the number of objectives to be optimized) of the following form:

$$\text{Minimize}\, f(x) = (f_1(x), f_2(x), \ldots, f_\beta(x)),$$

where x is the vector of the decision variables.

Each evaluation of the vector $f(x)$ is denoted by $\widetilde{\phi}$ and the value of the i-th objective is represented by ϕ_i. Therefore, the objective space is represented as

$$\Phi = \{\widetilde{\phi} = (\phi_1, \phi_2, \ldots, \phi_\beta) : \phi_i = f_i(\widetilde{x}) \\ \forall \widetilde{x} \in X,\ i = 1, 2, \ldots, \beta\},$$

where X is the feasible region of the problem.

Throughout the paper, the following definitions will be considered.

Definition 1 (*Dominance*)**.** Let $\widetilde{\phi}, \widetilde{\phi}' \in \Phi$. Then $\widetilde{\phi}$ dominates $\widetilde{\phi}'$ (i.e., $\widetilde{\phi} \succ \widetilde{\phi}'$) if and only if $\widetilde{\phi}_i \le \widetilde{\phi}'_i$, $\forall i = 1, 2, \ldots, \beta$ and at least one of them is a strict inequality.

Definition 2 (*Pareto Efficiency*)**.** Let $\widetilde{x} \in X$ be a feasible solution of the problem. It is a Pareto efficiency solution if and only if there is not a solution $\widetilde{x}' \in X$ such that $f(\widetilde{x}') \succ f(\widetilde{x})$.

Definition 3 (*Efficiency Solutions*)**.** The set $\Pi = \{\widetilde{x} \in X\}$ is defined as the set of the Pareto efficiency solutions if and only if each $\widetilde{x}$ is a Pareto efficiency solution.

Definition 4 (*Pareto Front*)**.** The Pareto front F is defined by $F = \{f(\widetilde{x}) : \widetilde{x} \in \Pi\}$.

Since different feasible solutions could match the same point in Φ, the Pareto front can be approximated for giving a set of equivalent solutions to the decision maker. Therefore, the following two definitions are introduced.

Definition 5 (*Ideal Point*)**.** The vector $\widetilde{\phi}^{\text{IDEAL}} = (\phi_1^{\text{IDEAL}}, \phi_2^{\text{IDEAL}})$ such that

$$\phi_1^{\text{IDEAL}} = \min_{\widetilde{\phi} \in \Phi}\{\phi_1\}, \qquad \phi_2^{\text{IDEAL}} = \min_{\widetilde{\phi} \in \Phi}\{\phi_2\}$$

represents the *Ideal point*.

Definition 6 (*Nadir Point*)**.** The vector $\widetilde{\phi}^{\text{NADIR}} = (\phi_1^{\text{NADIR}}, \phi_2^{\text{NADIR}})$ such that

$$\phi_1^{\text{NADIR}} = \min_{\widetilde{\phi} \in \Phi}\{\phi_1 : \phi_2 = \phi_2^{\text{IDEAL}}\},$$

$$\phi_2^{\text{NADIR}} = \min_{\widetilde{\phi} \in \Phi}\{\phi_2 : \phi_1 = \phi_1^{\text{IDEAL}}\}$$

represents the *Nadir point*.

4. A multi-objective job scheduling in cloud computing

This section describes the proposed multi-objective mathematical formulation of the job scheduling problem on a homogeneous cloud platform. With reference to the assumptions and notations introduced in Section 3, three objectives are optimized: the total average waiting time of the jobs, the *makespan* (denoted by δ) and the number of used hosts. The mathematical model uses the following variables and input data:

- $x_\omega, \forall \omega \in \Omega$ that denotes a binary decision variable equal to 1 is the feasible schedule $\omega \in \Omega$ belongs to the final solution, 0 otherwise;
- $A \in R^{|J| \times |\Omega|}$ that represents a binary matrix whose element $a_{j\omega}$ is equal to 1 if the job j is processed in the feasible working schedule ω, 0 otherwise;
- c_ω that is the cost of the feasible working schedule ω evaluated in terms of the total average waiting time of its jobs.

Thus, the three-objective ($\beta = 3$) mathematical model is the following.

Minimize

$$\delta \tag{3}$$

Minimize

$$\sum_{\omega \in \Omega} c_\omega x_\omega \tag{4}$$

Minimize

$$\sum_{\omega \in \Omega} x_\omega \tag{5}$$

subject to

$$\sum_{\omega \in \Omega} a_{j\omega} x_\omega = 1 \quad \forall j \in J \tag{6}$$

$$\delta \geq c_\omega x_\omega \quad \forall \omega \in \Omega \tag{7}$$

$$\delta > 0 \tag{8}$$

$$x_\omega \in [0, 1] \quad \forall \omega \in \Omega. \tag{9}$$

The objective functions ((3)–(5)), to be minimized, take into account the makespan, the total average waiting time and the number of used hosts, respectively. Moreover, the constraint (6) guarantees that each job j has to be assigned to only one working schedule ω, the constraint (7) imposes that δ is greater or equal to the average waiting time of each selected working schedule ω and, finally, the constraints (8) and (9) are the non-negative and binary conditions on δ and x-variables, respectively.

However, since it is always possible to detect an upper bound $\overline{m} > 0$ and a lower bound $\underline{m} > 0$ on the number of used hosts, the aforementioned formulation can be transformed into a bi-objective model ($\beta = 2$) that aims at minimizing δ and the total average waiting time. The number of used hosts m is properly varied in $[\underline{m}, \overline{m}]$ where $\underline{m}$ is determined as follows:

$$\underline{m} = \left\lceil \frac{\sum_{j \in J} w_j}{u} \right\rceil$$

and $\overline{m}$ is set equal to $|J|$ (i.e., one host for each job). Thus, the bi-objective model is obtained removing Eq. (5) from the formulation (3)–(9) and introducing the following constraint:

$$\sum_{\omega \in \Omega} x_\omega \leq m \tag{10}$$

that imposes that the number of used hosts is less than or equal to m. Varying m in $[\underline{m}, \overline{m}]$, several bi-objective formulations are solved and then an approximation of F can be determined.

4.1. The weighted sum method

The weighted sum method is one of the most popular and simple methods for solving multi-objective optimization problems.

It belongs to the class of a posteriori methods in which the set of all the Pareto optimal points is firstly generated. Then, the decision maker will choose the solution that better meets its requirements.

In this solution approach, the objective functions to optimize are expressed as a linear combination introducing the weights $\text{weight}_j \geq 0$ with $j = 1, \ldots, \beta$ (such as one weight for each $f_j(x)$). Therefore, the new objective function can be written as follows:

$$\sum_{j=1}^{\beta} \text{weight}_j f_j(x). \tag{11}$$

Moreover, the weights are assumed to be normalized such as

$$\sum_{j=1}^{\beta} \text{weight}_j = 1. \tag{12}$$

For each of the possible values of m, this method is applied to the bi-objective formulation of the job scheduling problem. Therefore, the objective function is expressed as follows:

$$\text{weight}_1 \delta + \text{weight}_2 \sum_{\omega \in \Omega} c_\omega x_\omega. \tag{13}$$

Moreover, according to Eq. (12), weight_2 can be obtained as $1 - \text{weight}_1$ and therefore, the objective function (13) can be transformed into

$$\text{weight}_1 \delta + (1 - \text{weight}_1) \sum_{\omega \in \Omega} c_\omega x_\omega. \tag{14}$$

Varying weight_1 in the range [0, 1], several no dominated solutions can be determined.

In Algorithm 1, the outline of the weighted sum based algorithm implemented for solving the multi-objective job scheduling problem in a cloud computing platform is provided, where Λ is a parameter set to 0.5.

```
input  : $\Omega$, $c_\omega \forall \omega \in \Omega$, $w_j \forall j \in J$, $u$, $\Lambda$
output: The Pareto front $F$
$weight_1 = 0$;
Set the iteration counter $ic = 1$;
Set $F := \emptyset$;
while $weight_1 \leq 1$ do
    Solve the single-objective model (14),(6)-(9), (10) and
    let $x^{ic}$ be the optimal solution;
    Set $F := F \bigcup \{x^{ic}\}$;
    Set $weight_1 = weight_1 + \Lambda$;
    Set $ic = ic + 1$;
end
Remove dominated solutions from $F$;
```

Algorithm 1: The weighted sum based algorithm.

4.2. Approximate ϵ-constraint approach

This section describes the ϵ-constraint method applied to the job scheduling problem in a homogeneous cloud platform. It solves a set of ϵ-constraint problems $P_j(\epsilon)$, one for each objective function f_j, for $j = 1, 2, \ldots, \beta$. Each of these sub-problems optimizes only one objective and imposes upper bounds on the remaining ones. In the bi-objective case, the ranges for ϵ_1 and ϵ_2 are $[\underline{\epsilon_1}, \overline{\epsilon_1}]$ and $[\underline{\epsilon_2}, \overline{\epsilon_2}]$, respectively, where $\underline{\epsilon_1} = \phi_1^{\text{IDEAL}}$, $\overline{\epsilon_1} = \phi_1^{\text{NADIR}}$, $\underline{\epsilon_2} = \phi_2^{\text{IDEAL}}$ and $\overline{\epsilon_2} = \phi_2^{\text{NADIR}}$.

Then, for each value of ϵ_2, the following mathematical model is solved:

$P_1(\epsilon_2)$

Minimize

$$f_1(x) \tag{15}$$

subject to

$$x \in X \tag{16}$$

$$f_2(x) \leq \epsilon_2. \tag{17}$$

While for each ϵ_1, the following mathematical model is solved:

$P_2(\epsilon_1)$

Minimize

$$f_2(x) \tag{18}$$

subject to

$$x \in X \tag{19}$$

$$f_1(x) \leq \epsilon_1. \tag{20}$$

Therefore, the methodology introduces as many ϵ values as the number of objectives to be optimized. Moreover, it is worth noting that the efficient solutions can always be found by solving ϵ-constraint problems as long as either the ϵ_2 values make $P_1(\epsilon_2)$ feasible or the ϵ_1 values make $P_2(\epsilon_1)$ feasible. On the other hand, for each efficient solution x, an ϵ_j can be detected such that x solves $P_1(\epsilon_2)$ or $P_2(\epsilon_1)$. Generally speaking, solving as many ϵ-constraint problems as the values of ϵ_j aims at generating at least one solution for each point belonging to F and therefore, it could generate the exact Pareto front. The method defines a finite sequence of ϵ-constraint problems through a progressive reduction of the values of the critical parameters ϵ_j. With reference to the bi-objective job scheduling formulation, two sub-problems can be formulated applying the ϵ-constraint methodology introducing ϵ_1 as the upper bound for the total average waiting time and ϵ_2 as the upper bound for δ. The first formulation (Scheduling$_1(\epsilon_2)$) minimizes (4), subjected to the constraints (6), (7)–(9), (10) and

$$\delta \leq \epsilon_2. \tag{21}$$

The second formulation (Scheduling$_2(\epsilon_1)$) minimizes (3), subjected to the constraints (6), (7)–(9), (10) and

$$\sum_{\omega \in \Omega} c_\omega x_\omega \leq \epsilon_1. \tag{22}$$

The solution approach can be summarized as outlined in Algorithm 2.

```
input  : $\Omega$, $c_\omega \forall \omega \in \Omega$, $w_j \forall j \in J$, $u$, $\epsilon_1$, $\epsilon_2$, $\Delta$
output: The Pareto front $F$
$a = 1$ and $b = 2$ or $a = 2$ and $b = 1$;
Set $F := \emptyset$;
Compute Nadir and Ideal points: $\Phi^{IDEAL}$, $\Phi^{NADIR}$;
Set $F := F \bigcup \{(\phi_a^{IDEAL}, \phi_b^{NADIR})\}$;
Set $\epsilon_b := \phi_b^{NADIR} - \Delta$;
while $\epsilon_b \geq \phi_b^{IDEAL}$ do
    Solve $Scheduling_a(\epsilon_b)$;
    Set $F := F \bigcup \{(\phi_a^*, \phi_b^*)\}$;
    Set $\epsilon_b = \epsilon_b - \Delta$;
end
Remove dominated solutions from $F$;
```

Algorithm 2: ϵ-constraint algorithm.

Δ is a positive constant for reducing ϵ_b at each iteration and thus spanning the solution space. Usually, $\Delta = 1$, since the difference between two integer objective values is at least equal to 1. This parameter setting ensures that a sequence of ϵ-constraint problems generates one feasible solution for each point of F (Theorem 3 in [15]).

4.3. Generating the Ω set

One relevant issue concerns the generation of feasible working schedules (i.e., Ω set). It plays a key role in the approach and represents the input of Algorithm 2. A heuristic procedure is usually used for generating Ω. The heuristic procedure implemented in this work is summarized in Algorithm 3. An optimal working schedule satisfies Eq. (1) and minimizes the average waiting time of its jobs.

A heuristic for detecting a feasible working schedule $\widetilde{s}$ is to sort its jobs by increasing processing times. For example, in the case of 6 jobs ($|J| = 6$), host capacity $u = 8$ time unit (TU), job workloads $w_1 = 5$ TU; $w_2 = 3$ TU; $w_3 = 3$ TU; $w_4 = 2$ TU; $w_5 = 2$ TU and $w_6 = 5$ TU and processing times (expressed in time slots (TS)) $t_1 = 3$; $t_2 = 10$; $t_3 = 4$; $t_4 = 5$; $t_5 = 7$ and $t_6 = 8$; all the feasible working schedules (according to (1)) are:{1}; {2}; {3}; {4}; {5}; {6}; {1, 2}; {1, 3}; {1, 4}; {1, 5}; {2, 3}; {2, 4}; {2, 5}; {2, 6}; {3, 4}; {3, 5}; {3, 6}; {4, 5}; {4, 6}; {5, 6}; {2, 3, 4}; {2, 3, 5}; {2, 4, 5}; {3, 4, 5}. For example, the notation {2, 3, 4} denotes a feasible schedule because, according to $w_2 = 3$ TU, $w_3 = 3$ TU and $w_4 = 2$ TU, and to $u = 8$ TU, the jobs 2, 3 and 4 can be processed on the same host: $w_2 + w_3 + w_4 = 3 + 3 + 2 = 8$. On the other hand, according to the relative workloads and to $u = 8$ TU, the schedule {1, 2, 3, 4, 5, 6} cannot be considered as feasible: $w_1 + w_2 + w_3 + w_4 + w_5 + w_6 = 5 + 3 + 3 + 2 + 2 + 5 = 20 > 8$. This numerical example shows how both u and the job workloads affect the feasibility of a schedule.

Focusing attention on the feasible working schedule {2, 3, 4} and sorting its jobs according to increasing processing times, the best way to handle them is {3, 4, 2} with an average waiting time equal to $(0+4+9)/3 = 4.33$TS. Therefore, this sequence is added to Ω.

It is worth noting that the cardinality of Ω affects the solution quality. In fact, if many feasible working schedules are generated, the procedure selects solutions that better meet the criteria; on the other hand, a high cardinality of Ω determines more complex formulations (Scheduling$_1(\epsilon_2)$, Scheduling$_2(\epsilon_1)$) that usually require long computational times. Therefore, in this paper, all the feasible working schedules generated by the heuristic are taken into consideration only in small and medium scale scenarios; on the contrary, in large scale scenarios, its cardinality is limited by a control parameter ξ, properly set by the user.

```
input  : $J$, $H$, $w_j \forall j \in J$, $u$
output: The set $\Omega$
Set $\Omega := \emptyset$;
$S$ := set containing all the possible working schedules to
handle the jobs in the set $J$;
$\bar{S} \subseteq S$ set containing only the feasible working schedules;
for $\bar{s} \in \bar{S}$ do
    $\widetilde{s}$ := optimal working schedule for handling the jobs
    belonging to $\bar{s}$;
    Set $\Omega := \Omega \bigcup \{\widetilde{s}\}$;
end
```

Algorithm 3: Generating Ω.

5. Experimental phase

This section aims at analyzing the proposed ϵ-constraint approach comparing it with the WS method. Since in the literature testing instances do not exist for this specific problem, the first criticism to address concerns the generation of consistent and realistic scenarios (Section 5.1). The second criticism concerns properly finding the bounds on the total average waiting time and the

Table 1
Ideal and Nadir points.

Problem	$\underline{\epsilon_2}$	$\overline{\epsilon_2}$	$\underline{\epsilon_1}$	$\overline{\epsilon_1}$
1	0.38	1.40	0.38	2.02
2	0.38	1.00	0.96	1.49
3	0.00	2.31	0.00	2.31
4	0.37	0.76	1.88	3.10
5	0.37	0.56	1.96	3.59
6	0.00	1.40	0.00	2.02
7	0.00	1.23	0.00	2.28
8	0.00	0.82	0.00	2.12
9	0.15	0.82	0.15	2.35
10	1.55	5.23	1.80	15.68
11	0.21	0.75	0.72	2.77
12	2.14	7.51	7.17	27.71
13	0.37	0.56	1.96	3.59
14	0.37	0.56	1.96	3.60
15	3.66	5.60	19.65	35.89
16	0.37	0.80	1.96	4.96
17	0.96	11.66	0.96	11.66
18	0.62	0.84	4.28	7.31
19	0.00	0.01	0.06	0.09
20	0.00	0.08	0.00	1.20

Table 3
Numerical comparisons II.

Problem	$\Sigma^{\text{epsilon}}_{\text{spread}}$	$\Sigma^{\text{WS}}_{\text{spread}}$	$\Sigma^{\text{epsilon}}_{\text{spacing}}$	$\Sigma^{\text{WS}}_{\text{spacing}}$
1	0.51	0.56	0.07	0.11
2	0.65	0.63	0.13	0.15
3	0.51	0.74	0.09	0.12
4	0.9	0.8	0.02	0.03
5	1.16	0.86	0.02	0.03
6	0.47	0.5	0.08	0.09
7	0.54	0.52	0.04	0.03
8	0.55	0.48	0.03	0.03
9	0.93	0.66	0	0.01
10	0.6	0.57	0.16	0.19
11	0.96	0.55	0	0.01
12	1.11	0.6	0.01	0.09
13	1.11	0.87	0.02	0.03
14	1.17	0.86	0.01	0.03
15	1.33	0.93	0.08	0.21
16	0.97	0.77	0.02	0.02
17	0.67	–	0.96	–
18	0.62	0.4	0.04	0.19
19	0.36	0.67	0	0.02
20	0.29	–	0.01	–

makespan in order to run the proposed approach (Section 5.2). Finally, in order to evaluate the quality of our approach, some numerical results are presented. This last part of the work aims also to compare the proposed approach with the WS method in terms of quality of the Pareto fronts (Section 5.3).

In particular, the numerical results are analyzed with reference to δ, the total average waiting time and the number of the used hosts.

The two solution approaches have been implemented in *Java*, in *Eclipse* environment. Moreover, *Cplex* (release 9.0, [24]) has been used as an optimizer.

5.1. Generating consistent scenarios

This section aims at generating a set of consistent and realistic scenarios in order to conduct experiments and to do comparisons. To this purpose, an instance generator has been implemented in *Java*, in *Eclipse* environment.

The workloads are assumed to be distributed according to either a normal probability function ($w_j \sim N(\mu, \sigma), \forall j \in J$) or a continuous uniform distribution ($w_j \sim U(\underline{r}, \overline{r}), \forall j \in J$). The job processing times are instead assumed to be distributed according to either a Poisson probability function ($t_j \sim P(\lambda), \forall j \in J$) or an exponential distribution ($t_j \sim E(\lambda), \forall j \in J$). These are the probability distribution functions traditionally used in the literature for representing the workloads and the processing times.

The host capacity is assumed to be fixed at most to 100 while ξ varies from 1000 up to 10 000, $|H|$ is set equal to {5, 10, 15, 20, 40, 50} and $|J|$ varies from 5 up to 100.

5.2. Lower and upper bounds

In order to carry out experiments, significant and consistent lower and upper bounds have been found on both the total average waiting time and the makespan (as reported in Table 1).

5.3. Computational results

In this section, numerical comparisons are presented between the proposed approach and the WS method. Table 2 highlights the extreme points of the Pareto fronts and the number of non-dominated solutions for each solution approach and problem test. In particular, E_1 and E_2 are the extreme points of the Pareto fronts found by our approach while WS_1 and WS_2 are the extreme points of the Pareto fronts obtained by the WS method. Moreover, η_E and

Table 2
Numerical comparisons I.

Problem			E_1			E_2			WS_1			WS_2	η_E	η_{WS}
1	2	1.21	2.02	5	0.38	0.38	2	1.21	2.02	5	0.38	0.38	8	6
2	4	0.53	1.49	5	0.52	0.96	4	0.53	1.49	5	0.52	0.96	4	4
3	2	0.80	1.57	5	0.00	0.00	1	2.31	2.31	5	0.00	0.00	7	7
4	8	0.65	3.10	10	0.55	1.88	8	0.65	3.10	10	0.55	1.88	9	8
5	7	0.55	3.59	10	0.44	1.96	7	0.55	3.59	10	0.44	1.96	11	9
6	2	1.21	2.02	6	0.00	0.00	2	1.21	2.02	6	0.00	0.00	8	7
7	3	0.87	2.18	8	0.00	0.00	3	0.80	2.28	8	0.00	0.00	12	12
8	3	0.80	2.04	9	0.00	0.00	3	0.76	2.12	9	0.00	0.00	12	11
9	3	0.82	2.35	10	0.15	0.15	3	0.82	2.35	10	0.15	0.15	30	19
10	5	4.89	15.59	10	1.80	1.80	5	3.83	15.68	10	1.80	1.80	13	11
11	4	0.71	2.77	10	0.37	0.72	4	0.71	2.77	10	0.37	0.72	42	18
12	4	7.23	27.66	10	3.60	7.20	4	7.08	27.71	10	3.68	7.17	70	20
13	7	0.55	3.59	10	0.44	1.96	7	0.55	3.59	10	0.44	1.96	11	9
14	7	0.55	3.60	10	0.44	1.96	7	0.55	3.60	10	0.44	1.96	13	9
15	7	5.58	35.59	10	4.21	19.75	7	5.55	35.89	10	4.39	19.65	16	10
16	11	0.65	4.36	15	0.44	1.99	10	0.71	4.96	15	0.45	1.96	13	15
17	14	0.96	11.66	15	0.94	10.70	14	0.96	11.66	14	0.96	11.66	2	1
18	10	0.84	7.31	20	0.64	1.50	10	0.84	7.31	14	0.62	4.28	15	5
19	25	0.01	0.09	40	0.00	0.01	25	0.01	0.09	28	0.01	0.06	9	2
20	35	0.08	1.20	41	0.06	0.75	35	0.08	1.20	35	0.08	1.20	7	1

Table 4
Nomenclature.

Symbol	Meaning
n	Number of jobs to be executed
J	Set of jobs
m	Number of available homogeneous hosts
H	Set of available homogeneous hosts
w_j	Workload of $j \in J$
u_h	Capacity of $h \in H$
t_j	Processing time required by $j \in J$
Ω	Set of feasible working schedules
x_ω	Binary decision variable
δ	Makespan
A	Binary matrix
c_ω	Cost of $\omega \in \Omega$
ξ	Parameter that controls the cardinality of Ω

η_{WS} denote the number of no dominated solutions detected by our approach and the WS method, respectively.

From Table 2, it is evident that our approach outperforms the WS method in terms of number of no dominated solutions. In fact, over all the problem tests, the average number of no dominated solutions detected by our solution approach is equal to 16 while the WS method reaches an average number equal to 9.

However, in order to assert about the quality of our approach, two significant metrics are also introduced and used: *Spacing Metric* (Σ_{spacing}) (see [25]) and *Spread Metric* (Σ_{spread}) (see [26]).

In particular, Σ_{spacing} relies on the evaluation of the Euclidean distance d_i between consecutive efficient solutions and the average of distance $\bar{d}$:

$$\Sigma_{\text{spacing}} = \sum_{i=1}^{\eta-1} \frac{|d_i - \bar{d}|}{\eta - 1} \tag{23}$$

where η represents the number of no dominated solutions. Because it does not take into account the spread, Σ_{spread} is also considered:

$$\Sigma_{\text{spread}} = \frac{d_f + d_l + \sum_{i=1}^{\eta-1} |d_i - \bar{d}|}{d_f + d_l + (\eta - 1)\bar{d}}, \tag{24}$$

where d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of the efficient set; $\bar{d}$ represents the average of all distances d_i, $i \in [1, \eta - 1]$. The smaller the value of Σ_{spread}, the better the diversity of the efficient set.

Table 3 clearly highlights that both the two approaches detect well uniformly distributed solutions with an average Σ_{spacing} equal to $0.1 \ll 1$. However, the WS method finds sets of no dominated solutions that are slightly more diversified than the ϵ-constraint approach (i.e., $\Sigma_{\text{spread}}^{\text{WS}} = 0.7 < \Sigma_{\text{spread}}^{\text{epsilon}} = 0.8$).

6. Conclusions and future work

This paper addresses a multi-objective off-line job scheduling problem on a homogeneous cloud computing platform. For that, an approximate ϵ-constraint method is designed, implemented and evaluated. Moreover, the numerical results are compared with the WS method traditionally used for solving multi-objective optimization problems. The solution qualities are estimated in terms of the number of no dominated solutions, their distribution and diversification introducing two metrics: Σ_{spacing} and Σ_{spread}.

Further developments will be carried out on the implementation of alternative heuristic approaches used to populate Ω in order to detect more diverse sets of solutions. Moreover, an on-line job scheduling version of the problem will be examined and studied and also heterogeneous cloud platforms will be taken into consideration. In particular, a redefinition of the mathematical model will be necessary in order to explicitly include the costs for data transfer.

Appendix

See Table 4.

References

[1] D.G. Feitelson, A survey of scheduling in multiprogrammed parallel systems, Research Report RC 19790, 87657, IBM T. J. Watson Research Center, USA, 1994.

[2] D. Lifka, The ANL/IBM SP scheduling system, in: D.G. Feitelson, L. Rudolph (Eds.), Job Scheduling Strategies for Parallel Processing, in: Lect. Notes Comput. Sci., vol. 949, Springer-Verlag, 1995, pp. 295–303.

[3] Intel Corp., iPSC/860 Multi-User Accounting, Control, and Scheduling Utilities Manual, Order number 312261-002, 1992.

[4] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, B. Hirsbrunner, Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm, Future Generation Computer Systems (2011). http://dx.doi.org/10.1016/j.future.2011.05.006.

[5] Y.H. Lee, S. Leu, R.S. Chang, Improving job scheduling algorithms in a grid environment, Future Generation Computer Systems 27 (8) (2011) 991–998.

[6] Y.C. Lee, A.Y. Zomaya, Rescheduling for reliable job completion with the support of clouds, Future Generation Computer Systems 26 (8) (2010) 1192–1199.

[7] M. Paul, G. Sanyal, Task-scheduling in cloud computing using credit based assignment problem, International Journal on Computer Science and Engineering 3 (10) (2011) 3426–3430.

[8] V. Borovskiy, J. Wust, C. Schwarz, W. Koch, A. Zeier, A linear programming approach for optimizing workload distribution in a cloud, in: CLOUD COMPUTING 2011: The Second International Conference on Cloud Computing, GRIDs, and Virtualization, 2011, pp. 127–132.

[9] S. Chen, T. He, H.Y.S. Wong, K.W. Lee, L. Tong, Secondary job scheduling in the cloud with deadlines, IPDPS Workshops, 2011, pp. 1009–1016.

[10] M. Wieczorek, A. Hoheisel, R. Prodan, Towards a general model of the multi-criteria workflow scheduling on the grid, Future Generation Computer Systems 25 (3) (2009) 237–256.

[11] T. Stevens, M. De Leenheer, C. Develder, B. Dhoedt, K. Christodoulopoulos, P. Kokkinos, E. Varvarigos, Multi-cost job routing and scheduling in Grid networks, Future Generation Computer Systems 25 (8) (2009) 912–925.

[12] J.L. Lucas-Simarro, R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente, Scheduling strategies for optimal service deployment across multiple clouds, Future Generation Computer Systems (2012). http://dx.doi.org/10.1016/j.future.2012.01.007.

[13] S. Pandey, L. Wu, S.M. Guru, R. Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments, in: AINA'10 Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, 2010, pp. 400–407.

[14] R.T. Marler, J.S. Arora, Survey of multi-objective optimization methods for engineering, Structural and Multidisciplinary Optimization 26 (6) (2004) 369–395.

[15] J.F. Berube, M. Gendreau, J.Y. Potvin, An exact ϵ-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits, European Journal of Operational Research 194 (2009) 39–50.

[16] R.L. Becerra, C.A. Coello Coello, Epsilon-constraint with an efficient cultured differential evolution, in: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, 2007, pp. 2787–2794.

[17] M. Laumanns, L. Thiele, E. Zitzler, An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method, European Journal of Operational Research 169 (3) (2006) 932–942.

[18] M. Boulif, K. Atif, An exact multiobjective epsilon-constraint approach for the manufacturing cell formation problem, in: 2006 International Conference on Service Systems and Service Management, Vol. 2, 2006, pp. 883–888.

[19] M. Esmaili, N. Amjady, H.A. Shayanfar, Multi-objective congestion management by modified augmented ϵ-constraint method, Applied Energy 88 (3) (2011) 755–766.

[20] J. Aghaei, N. Amjady, H.A. Shayanfar, Multi-objective electricity market clearing considering dynamic security by lexicographic optimization and augmented epsilon constraint method, Applied Soft Computing 11 (4) (2011) 3846–3858.

[21] L. Grandinetti, F. Guerriero, D. Laganá, O. Pisacane, An optimization-based heuristic for the multi-objective undirected capacitated arc routing problem, Computers and Operations Research 39 (2012) 2300–2309.

[22] J. Blazewicz, Scheduling in Computer and Manufacturing Systems, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[23] L. Grandinetti, F. Guerriero, D. Laganá, O. Pisacane, An approximate ϵ-constraint method for the multi-objective undirected capacitated arc routing problem, in: P. Festa (Ed.), Proceedings of the 9th International Symposium on Experimental Algorithms, SEA'10, in: Lecture Notes in Computer Science, vol. 6049, Springer Verlag, 2010, pp. 214–225.

[24] http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[25] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, in: Parallel Problem Solving from Nature, Vol. 6, 2000, pp. 849–858.

[26] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, first ed., Wiley, Chichester, UK, 2001.

Lucio Grandinetti. Since 1986, Lucio Grandinetti has been a full-time professor at the Faculty of Engineering, University of Calabria (UNICAL), Italy. At the same University, he currently holds the position of Vice Rector (since 1999) and coordinator of the Ph.D. Degree Program in the area of Management Science and Engineering. At the above mentioned University, he was Director of the Department of Electronics, Informatics, and Systems for ten years, as well as member of the University Administration Council and President of the Management Engineering Degree Course. His scientific background is in Electronic Engineering and Systems Science. He is a graduate of the University of Pisa, Italy, and the University of California at Berkeley. He has also been a post-doc fellow at University of Southern California, Los Angeles, and Research Fellow at the University of Dundee, Scotland. He has been a member of the IEEE Committee on Parallel Processing, and European Editor of a book series of MIT Press on Advanced Computational Methods and Engineering. Currently he is a member of the Editorial Board of three international journals. He is the author of many research papers in well-established international journals and Editor or co-Editor of several books on algorithms, software, applications of Parallel Computing, HPC, Grids. Since 1994, he has been part of several Evaluation Panels of European Projects within various ICT and Infosociety Programs; he has also been a reviewer of many EU Projects in the above Programs. He has been a recipient and scientific leader of many European-Commission-Funded projects since 1993 (e.g. Molecular Dynamics Simulations by MPP Systems, EUROMED, HPC Finance, WADI). He has been a recipient and scientific leader of several national projects (CNR progetti finalizzati, Grid.it, PRIN). He is one of the founders of the Consortium SPACI (Southern Partnership for Advanced Computational Infrastructures), a pioneering Grid infrastructure experiment in Italy. Currently he is the Director of the Centre of Excellence on HPC established at the University of Calabria by the Italian government; co-managing director of a Supercomputing Centre jointly established by the University of Calabria and NEC Corporation; recipient and scientific leader of some European-Commission-Funded projects (among others, BEINGRID); Project leader of a node of the most important European Grid Infrastructure for e-science named EGEE.

Ornella Pisacane was born in Catanzaro (Italy) on August 11th 1981. In 2004, she obtained the Laurea degree in Computing Engineering at the University of Calabria (UNICAL) defending a thesis whose title was *Parallel Algorithms of Simulated Annealing for simulation-optimization*. She obtained the maximum score of 110/110 with honor. She was an assistant professor in Optimization Laboratory and Logistics course at UNICAL. From 2005 to 2006, she was a research fellow at the Departement d'Informatique et de Recherche Operationnelle-*Universitè* De *Montrèal*-(Canada). In 2008, she completed her Ph.D. in Operations Research at UNICAL, defending a thesis whose title was *Agent Scheduling in a Multi-skill Call Center*. From April until September 2008, she collaborated at the CIRRELT-*Universitè De Montrèal*-(Canada). From December 2007 until December 2012, she was a post-doctoral researcher at the Department of Electronics, Informatics and Systems of UNICAL. She is currently a post-doctoral researcher at the Information Engineering Department of Universitá Politecnica delle Marche. Her main research areas concern: *Grid Computing, Simulation Optimization, Multi-objective Programming*. She is the author of scientific publications, technical reports and the co-author of books.

Mehdi Sheikhalishahi is currently a Ph.D. student in Computer Science at University of Calabria in Italy working on critical trends in cloud computing; more specifically Mehdi's Ph.D. Thesis is on energy efficient computing in Infrastructure-as-a-Service cloud model. Mehdi's major research interests lie in the fields of promising computing technologies such as High Performance Computing, Grid and Cloud Computing with theoretical consideration.