Cost Effective, Reliable and Secure Workflow Deployment over Federated Clouds

Zhenyu Wen, Jacek Cała, Paul Watson, and Alexander Romanovsky

Abstract—The significant growth in cloud computing has led to increasing number of cloud providers, each offering their service under different conditions – one might be more secure whilst another might be less expensive or more reliable. At the same time user applications have become more and more complex. Often, they consist of a diverse collection of software components, and need to handle variable workloads, which poses different requirements on the infrastructure. Therefore, many organisations are considering using a combination of different clouds to satisfy these needs. It raises, however, a non-trivial issue of how to select the best combination of clouds to meet the application requirements. This paper presents a novel algorithm to deploy workflow applications on federated clouds. First, we introduce an entropy-based method to quantify the most reliable workflow deployments. Second, we apply an extension of the Bell-LaPadula Multi-Level security model to address application security requirements. Finally, we optimise deployment in terms of its entropy and also its monetary cost, taking into account the cost of computing power, data storage and inter-cloud communication. We implemented our new approach and compared it against two existing scheduling algorithms: Extended Dynamic Constraint Algorithm (EDCA) and Extended Biobjective dynamic level scheduling (EBDLS). We show that our algorithm can find deployments that are of equivalent reliability but are less expensive and meet security requirements. We have validated our solution through a set of realistic scientific workflows, using well-known cloud simulation tools (WorkflowSim and DynamicCloudSim) and a realistic cloud based data analysis system (e-Science Central).

Index Terms—Cloud computing, reliability, security, workflow, scheduling, cost

1 Introduction

With the advent of cloud computing, owners of large-scale computing infrastructure (specifically, the cloud providers) can rent out their computation power or storage to individual users. The success of this approach is demonstrated by the numerous cloud providers worldwide. For example, CloudRFP.com [1] has compiled a database of cloud hosting and related service providers, which includes over 190 companies. In addition, users also may have access to their own, private cloud infrastructure. Importantly, all these various infrastructure providers offer services of different quality. For example, considering cloud security, data centres must follow the privacy policy of the country where the centre is located. That can sometimes force users to use more expensive, or lower quality, providers for the sake of compliance with their security requirements. Furthermore, some cloud providers offer different type of clouds with different security level; e.g., Microsoft Azure provides more secure service called a private cloud.

With regards to the cloud reliability users have no easier choice. Despite the fact that the SLA (service-level agreement) for cloud computing is usually at the level of 99.95 percent availability for the compute service, cloud providers are not always able to this level. In practice, this level of availability means that the service can only be offline for about 20 minutes in a month, or only about 250 minutes per year. However, in early 2011 several high-profile technical companies were landed in trouble when Amazon's EC2 service suffered an outage [2] which lasted for almost 11 hours. In a month it gives only 98.47 percent of availability and in a year still only 99.87 percent. Similarly, the outage at the GoDaddy cloud provider took down millions of web sites.[1]

With all this variety and uncertainty, application developers who decide to host their system in the cloud face the issue of which cloud to choose to get the best operational conditions in terms of price, reliability and security. And the decision becomes even more complicated if their application consists of a number of distributed components, each with slightly different requirements. An answer to this need might be combining together different cloud providers.

Cloud federation offers the ability to distribute a single application on two or more cloud platforms, so that the application can benefit from the advantages of each of them [3]. The key challenge in attempting to realise this opportunity is how to find the best distribution (or deployment) of application components, which can yield the greatest benefits. In this paper we tackle this problem and propose an algorithm to partition a workflow-based application over federated clouds in order to exploit the strengths of each cloud. Our algorithm splits an application structured as a workflow such that the security and reliability requirements of each part are met, whilst the overall cost of execution is minimised.

• *The authors are with the School of Computing Science, Newcastle University, United Kingdom. E-mail: {z.wen, jacek.cala, paul.watson, alexander.romanovsky}@newcastle.ac.uk.*

Manuscript received 9 June 2015; revised 30 Dec. 2015; accepted 3 Mar. 2016. Date of publication 17 Mar. 2016; date of current version 8 Dec. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2016.2543719

1. K. Finley, "Godaddy outage takes down millions of sites, anonymous member claims responsibility," http://techcrunch.com/2012/09/10/godaddy-outage-takes-down-millions-of-sites.

For the purpose of the algorithm we extended our security model presented previously in [4] and adapted it to meet the new, multi-criteria requirements. The model, based on the Bell-LaPadula Multi-Level Security model [5], [6], may be used to address the confidentiality of systems, i.e., the absence of unauthorised disclosure of information [7]. In this work we use the model as a driver to partition workflows over a federated clouds to meet their security requirements.

The basis for the new algorithm is a new method to quantify the most reliable workflow deployments, which applies Shannon's information theory [8]. Using reliability information for each application component and the underlying cloud platform, the method calculates the entropy of a workflow deployment. The entropy value is then used as a constraint in the optimisation problem. We argue that by using entropy we can reduce the overall risk of workflow failures caused by a small number of components being deployed on less reliable clouds.

As there may be a number of deployments that meet our security and reliability requirements, we search for the option that minimises the monetary cost. Finding the optimal deployment is, however, a NP-hard problem [9], and so we need a heuristic algorithm to solve it. The two most common approaches are: (1) to linearise the problem by assigning weights to the criteria and then optimising the weighted sum [10], or (2) to optimise one criterion and keep the others constraints within predefined thresholds. In the first approach the difficulty is not only in defining the weights properly but also in the limitation of the simple, linear model which may not be able to accurately represent the complexity of the problem. Hence in this work we use the second approach.

To handle this multi-criteria and NP-hard problem we generate a valid initial solution and then apply a set of refinement methods to approach the optimum. At the same time we keep the time complexity of the algorithm as polynomial.

The contributions of this paper as follows:

- A scheduling algorithm that minimises the monetary cost of deployment of a workflow application across federated clouds. The algorithm takes into account security and reliability requirements as well as the monetary cost incurred from three main sources in the cloud: computation, data transfer and data storage.
- A novel method to quantify the reliability of a workflow deployment using entropy, which can reduce the overall risk of workflow failures caused by a small number of components being deployed on less reliable clouds.
- An extension of our previous Bell-LaPadula Multi-Level Security model to guarantee the security of deploying a workflow over federated clouds.
- An evaluation of our work with both randomly generated workflows simulated in a combination of WorkflowSim [11] and DynamicCloudSim [12] environments, and also an existing scientific workflow running on e-Science Central [13], a cloud-based data analysis platform.

The remainder of the paper is structured as follows. We first introduce the notation and models used to represent the reliability and security requirements and to calculate the monetary cost of deployment. Next, in Section 3 we describe the optimisation problem as described by the models. Then we present a scheduling algorithm to search efficiently for a suitable deployment option is presented. In Section 5 we discuss the evaluation of our work. Finally, future work is outlined and conclusions are drawn.

2 Problem Description

With the increasing availability of public and private cloud resources it is easy to deploy instances of the same service in multiple places. We observe this tendency with our e-Science Central data analysis system which, depending on the use case, has been deployed in a variety of locations including private clouds at universities in Spain and Brazil[2] and public cloud resources such as Amazon AWS and Microsoft Azure [13]. Each of these clouds has its own advantages and thus we focus in this paper on how a single workflow application might be deployed over a federated clouds. By a federated clouds we consider in this paper a set of workflow execution environments (such a e-Science Central) running in different clouds, which we can manage and use to run applications. Our goal is to partition a workflow application in such a way that it can benefit from the "best" combination of these environments.

The following sections define the three concepts that form the basis for our algorithm: the measure of reliability, the security rules and the cost model. Further, Table 1 shows the key notations used through the paper.

2.1 Reliability

In this paper the target clouds consist of a set of PMs (physical machines), and can be geographically distributed. A single physical machine is able to contain a set of VMs (virtual machines). Each VM can run 0 or more instances of WP (workflow execution platform). Last, a WP can run a number of services/workflow tasks (to put it simply, we use service in the rest of paper) concurrently. We are not considering fault tolerance [14] as a means for improving reliability; this is outside of the scope of this paper.

2.1.1 Reliability Computing

By reliability we mean the readiness for correct service, for example, the amount of time a device or service is actually operating as the percentage of total time it should be operating. Therefore, reliability can be defined as: $REL = \frac{\text{Fault}-\text{FreeTime}}{\text{TotalTime}}$.

In this paper, we assume that a service itself is implemented fault-free [15] and it is executed completely and reliably on a workflow execution platform, iff the physical machine, VM and WP which the service is running on are fault-free during the service execution time.

Let t be a random variable which represents a point in time when a failure happens, while f denotes the probability density function of t. We assume that failures are

2. This work was conducted within the EUBrazil Cloud Connect project; http://www.eubrazilcloudconnect.eu.

TABLE 1
Notations

Symbol	Meaning
Workflow	
w	a workflow application
S	a set services of workflow w
s_i	ith service in a workflow application
D	a set of data of workflow w
$d_{i,j}$	data dependency between s_i and s_j
$\mathcal{E}$	the union of data dependencies, services and clouds
e	one of the element of set $\mathcal{E}$
wp_i	ith cloud of a set of clouds WP
$d_{i,j}^{wp_i}$, $s_j^{wp_i}$	a set to represent $d_{i,j}$ or s_j, allocated in cloud wp_i
Φ	possible deployment solutions
ϕ	one of the deployments solution of Φ
Reliability Model	
REL	required reliability of each service
R_P	reliability of a deployment solution
R_E	entropy value of a deployment solution
R_i	the reliability rate of service s_i
R_{max}	the maximum reliability rate of the services
Security Model	
$\mathcal{D}$	a finite set of ordered pairs
$\prod L'(e)$	the upper bound of e
$\coprod L'(e)$	the lower bound of e
Cost Model	
$T_{i,j}$	storage time of data $d_{i,j}$
$Store_{wp_i}$	cost of storing data on wp_i in GB per hour
OUT	the outgoing data dependencies of a cloud
Com_{wp_i,wp_j}	cost of transferring 1 GB of data between clouds
IN	the incoming data dependencies of a cloud
$T_j^{wp_i}$	execution time of s_j on wp_i
$Exec_{wp_i}$	cost of using compute resources on wp_i for one hour
$SCOST$	data storage cost
$CCOST$	communication cost
$ECOST$	execution cost
$COST$	total cost of a workflow deployment

occurring randomly in time and can be modelled by an exponential distribution. Therefore, the exponential probability density function of WP is $f_{wp}(t) = \lambda_{wp} e^{-\lambda_{wp} t}$, where λ_{wp} is the failure rate of workflow platform wp. Then, following [15], we can define the reliability function of wp as:

$$R_{wp}(t) = 1 - \int_0^t f_{wp}(t)dt = e^{-\lambda_{wp} t}. \quad (1)$$

We assume that the execution time of service s_i is t_i and T_{wp_i} is the time since wp_i is available but just before s_i is launched. Thus we can have the following corollary.

Corollary 1. *The reliability of execution of s_i in wp_i is $R_i(t_i) = R_{wp_i}(t_i + T_{wp_i})$.*

Proof. From the discussion above, reliability $R_{vm_i}(t)$ of vm_i at time t depends on the reliability of the VM instance itself, $R'_{vm_i}(t)$, and the reliability of its host machine $R_{pm_i}(t + T_{pm_i})$, where T_{pm_i} is the time since pm_i has been available but just before vm_i is started. We can denote that as: $R_{vm_i}(t) = R'_{vm_i}(t) \cdot R_{pm_i}(t + T_{pm_i})$. Similarly, the reliability of wp_i is $R_{wp_i}(t) = R'_{wp_i}(t) \cdot R_{vm_i}(t + T_{vm_i})$, where T_{vm_i} is the time since vm_i has been available but just before wp_i is started. Consequently, we can state that the reliability of running service s_i in the Cloud is $R_i(t) = R'_i(t) \cdot R_{wp_i}(t + T_{wp_i})$. However, earlier we assume that service s_i is implemented fault-free and so $R_i(t) = R_{wp_i}(t + T_{wp_i})$. □

2.1.2 Measure of Workflow Reliability

We assume that workflow w consists of n services $s_1, , s_n$, and the reliability of each service is R_i. Moreover, the communication between related WPs and services are fault-free. Therefore, the possibility that w is failure-free is:

$$R_P(\phi) = \prod_{i=1}^{n} R_i = e^{\sum_{i=1}^{n} -\lambda_{wp_i}(t_i + T_{wp_i})}, \quad (2)$$

where ϕ is a deployment of w, λ_{wp_i} is the failure rate of workflow platform which is used to deploy s_i (we assume that λ_{wp_i} already includes the failure rate of the underlying layers), and $t_i + T_{wp_i}$ is the execution time of the WP since it was started and until s_i finished. R_P (power method thereafter) is the most common way to measure the reliability of the execution of a workflow application [16], [17] and [18].

However, in this paper we propose an *entropy-based* method to measure workflow reliability. Entropy [8] is a widely used approach to capture the degree of dispersal or concentration of a random variable. For a discrete random variable X with probabilities $p(x_i)$ it is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i). \quad (3)$$

We deem $p(x_i)$ as reliability of service s_i and use the entropy value as a measure of workflow reliability. Thus, the reliability of workflow w can be calculated as:

$$R_E(\phi) = -\sum_{i=1}^{n} R_i \log R_i. \quad (4)$$

In Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSC.2016.2543719, we illustrate that entropy method has more advantages than the traditional power method. Furthermore, we propose a constraint on the entropy value which can guarantee not only the reliability of a workflow deployment, but also reduces the risk that a workflow includes a service with relatively low reliability. If we assume that the required reliability of deploying workflow w is $\mathcal{R}$, then using the power method we need $R_P(\phi) \geq \mathcal{R}$. However, the corresponding entropy constraint is: $R_E(\phi) \leq -R_M \log \mathcal{R}$, where $R_M = MAX(R_i)$ is the maximum reliability of the workflow services.

2.2 Security Rules

A security model is needed to determine whether a deployment of services and data to a set of clouds meets organisation's security requirements. In this paper, we present a novel security model which extends our previous work [4],

building on the Bell-LaPadula model [5] and incorporating the security levels of the clouds, data and services. For clarity, from now on we assume that each cloud runs only a single instance of a workflow platform. This does not limit our security and cost models but improves readability.

To model security we follow [19] and use security lattice $\mathcal{L} = (L, \leq)$ to represent the security levels of any entity $e \in \mathcal{E} =$ (clouds, data and services). $\mathcal{L}$ denotes that set L has a partial order relation (i.e., $\leq$). For this the lattice has at least one least upper bound l ($l \sqcup l' \in L$) and one lower bound l' ($l' \sqcap l \in L$). Furthermore, the subset L' of L has a strict partial order (i.e., $(L', <)$), noting $\prod L'$ (upper bound) and $\coprod L'$ (lower bound).

We assign the security levels to each object $e \in \mathcal{E}$, for example, $L'(e) \in L$ which means e is assigned multiple security levels from $\prod L'(e)$ to $\coprod L'(e)$. A workflow w, including D data and S services, is going to be deployed on a set of WP. Φ is all possible distributions of objects: $\Phi = \mathcal{E} \times L$. Moreover, the dependency relation of data and services, for example, is data $d_{i,j}$ which is generated from service s_i and consumed in service s_j:

$$\boxed{s_i} \rightarrow \boxed{d_{i,j}} \rightarrow \boxed{s_j}$$

(where arrow $\rightarrow$ is used to show data dependency, and each block – service or data – is uniquely identified by the subscript).

$\mathcal{D}$ is a finite set of ordered pairs, thus our security model consists of three rules.

Rule 1. "no-write-down": denotes that a service cannot write data which has lowest security level (required security level) that is lower than service own slef,

$$\forall (s_i, d_{i,j}) \in \mathcal{D} \coprod L'(s_i) \leq \coprod L'(d_{i,j}).$$

Rule 2. "no-read-up": means a service cannot read data if the data's lowest security level is higher than the service's highest security level,

$$\forall (d_{i,j}, s_j) \in \mathcal{D} \coprod L'(d_{i,j}) \leq \prod L'(s_j).$$

Rule 3. "security in cloud computing": defines that the lowest security level of a cloud must be greater than or equal to the lowest security level of any service or data hosted by this cloud,

$$\forall \left(d_{i,j}^{wp_i}, s_j^{wp_i}\right) \in \mathcal{D} \coprod L'(d_{i,j}) \leq \coprod L'(wp_i) \text{ and } \coprod L'(s_j) \leq \coprod L'(wp_i),$$

Where set $d_{i,j}^{wp_i} = \{d_{i,j}, wp_i\}$ and $s_j^{wp_i} = \{s_j, wp_i\}$ represent $d_{i,j}$ and s_j as allocated in wp_i respectively. The lowest security level of both must be less than or equal to wp_i's lowest security level.

Based on the definition of the three security rules (noting W thereafter), a secure workflow deployment can be defined as:

$$\phi \in \sum(\Phi, W). \tag{5}$$

Where ϕ denotes a deployment of workflow w on a set of WP which meets the security requirements of W, noting $\phi = \{s_1^{wp_i}, \ldots, s_n^{wp_j}\}$. $\sum$ represents a set of deployments which can meet the security rules W. In Appendix B, available in the online supplemental material, we have demonstrated how to apply the security rules to a simple pipeline workflow.

2.3 Cost Model

In this section we introduce the cost model that plays a key role in our algorithm. We assume that clouds are linked in a fully connected topology and data can be freely transferred between them. Additionally, a wp can run several services at the same time.

To calculate the cost of executing a service in the federated clouds we define a set of cost functions. First is the data storage cost:

$$SCOST\left(s_i^{wp_i}\right) = \sum_{d_{i,j} \in OUT} d_{i,j} \times T_{i,j} \times Store_{wp_i}, \tag{6}$$

Where $s_i^{wp_i}$ means service s_i is deployed on cloud platform wp_i. OUT is a set of data dependencies, representing that data are generated by s_i and transferred to its immediate successor s_j which is not deployed on platform wp_i (note that if all immediate successors of s_i are on wp_i, $OUT = \emptyset$). $d_{i,j}$ represents the amount of data (in GB) which is generated by s_i and consumed by s_j. $T_{i,j}$ denotes storage time of data $d_{i,j}$, which is the time from the data being generated until completion of the workflow execution (in hr). Finally, $Store_{wp_i}$ is the cost of storing 1 GB of data for 1 hour on wp_i (in \$/GB/hr).

Only a few researchers consider the data storage cost when deploying workflows over a federated clouds. However, the following four reasons describe why it is worth taking into account: 1) it makes the cost model more complete because cloud providers usually charge not only for compute resources and data transfer but also for data storage; 2) for data intensive workflows the storage cost becomes considerable, especially when data needs to be transmitted between two clouds; 3) storing the output of a workflow partition acts as a checkpoint which can reduce the loss in the event of an outage in any clouds running the following partitions; 4) such data checkpointing mechanisms are implemented by some workflow management platforms. For example, the e-Science Central cloud platform [13] implicitly stores the data that need to be transferred between partitions.

Next function, $CCOST$, is used to estimate the communication cost for services:

$$CCOST(s_j^{wp_i}) = \sum_{d_{i,j} \in IN} d_{i,j} \times Com_{wp_i, wp_j}. \tag{7}$$

It is the cost of the data transferred from the immediate predecessors of service s_j (denoted as IN). Com_{wp_i,wp_j} represents the unit cost of transferring 1 GB of data from wp_i to wp_j (in \$/GB). Note that if two services are deployed on the same platform, the unit cost is zero, i.e., $\forall wp_i = wp_j : Com_{wp_i,wp_j} = 0$.

Finally, $ECOST(s_j^{wp_i})$ indicates the execution cost of service s_j on wp_i. It is defined as:

$$ECOST(s_j^{wp_i}) = T_j^{wp_i} \times Exec_{wp_i}. \quad (8)$$

Where $T_j^{wp_i}$ is the execution time of s_j on platform wp_i (in hr), and $Exec_{wp_i}$ represents the cost of using compute resources on wp_i for 1 hour (in \$/hr).

Based on these three functions, we can define the total cost of deploying a workflow over a set of clouds:

$$COST(\phi) = \sum_{s_i^{wp_i} \subset \phi} SCOST(s_i^{wp_i}) + CCOST(s_i^{wp_i}) + ECOST(s_i^{wp_i}). \quad (9)$$

Where ϕ is one of the deployment solutions, $\phi \in \Phi$.

3 The Challenges

3.1 Multi-Objective Optimisation Problem

For a given ϕ which is a secure deployment of workflow w over federated clouds WP, we propose to optimise two parameters: i) minimise the value of entropy $H(\phi)$ which results in a distribution that maximises the reliability of the workflow, ii) minimise the value of $COST(\phi)$ to obtain deployments with low cost of execution. We express this problem as:

$$\textbf{minimise}\ (COST(\phi), (H(\phi))$$
$$\textbf{subject to}\ \exists \phi \in \sum(\Phi, W)$$
$$\Phi = D \times S \times WP \times L.$$

Finding a solution which optimises both $COST(\phi)$ and $H(\phi)$ is challenging. Note that if we simplify our problem to only one objective optimisation, e.g., finding deployment ϕ which minimises cost, it can be directly mapped to the *skewed* graph partitioning [20] which is a variant of classic NP-complete graph partitioning [9].

In the skewed graph partitioning each vertex i has a desire to be in set k denoted by $d_k(i)$. Given that, the problem tries to minimise the cut edges (flow of data between sets) and maximise the desires. If $w(e_{ij})$ is the weight associated to the edge between vertices i and j and $s(i)$ is the set to which the vertex i is assigned, the problem tries to find mapping s which minimises objective function:

$$\textbf{minimise} \sum_{e_{ij}} \begin{cases} w(e_{ij}), & \text{if } s(i) \neq s(j) \\ 0, & \text{otherwise} \end{cases} - \sum_{i=1}^{n} d_{s(i)}(i).$$

In our case vertices denote workflow services, weights associated with edges represent the cost of data transfer between services, set $s(i) = wp_i$ is the workflow platform assigned to execute service s_i and, finally, desire $d_{s(i)}(i)$ is the negative value of the execution and data storage cost associated with running service s_i on platform wp_i.

3.2 Trade-off Problem

To have a better understanding of the trade-off between entropy and the monetary cost of workflow deployments we ran two sets of experiments. First we used the NCF (Not Cheapest First) algorithm from our previous work [21] to optimise deployment by cost. NCF is an approximate greedy algorithm which can quickly find deployments of workflows with security constraints. Then we implemented another greedy algorithm to optimise deployment by its entropy value. Finally, we ran these two algorithms to deploy 300 different randomly generated workflows which included from 2 to 30 services. Fig. 1 shows the result.

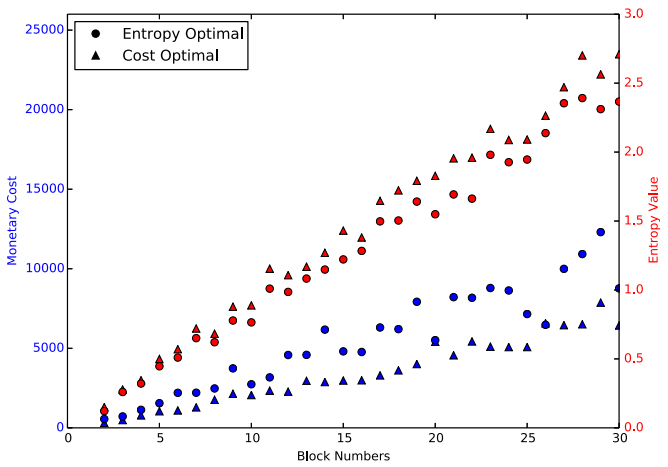

Fig. 1. Cost and entropy optimal workflow deployments.

Triangles represent the deployments which minimise the monetary cost, whereas circles are deployments with minimal entropy. Colours are related to axes with blue showing the monetary cost of a deployment and red showing its entropy. Clearly, the cost optimised deployments are cheaper than the entropy optimal ones. However, they have also greater entropy value. Conversely, the entropy optimal deployments have lower entropy but result in more expensive deployments. Also, as may be seen, the gap between cost optimal and entropy optimal deployments increases with the increasing number of services in a workflow, which clearly indicates the trade-off between these values.

4 Scheduling Algorithm

The scheduling algorithm we propose can optimise the deployment of a workflow over a set of clouds. It takes into account user requirements against multiple criteria, namely security, reliability and cost. The optimisation part of the algorithm is an extension of the multiple-choice knapsack problem (MCKP) [22].

Overall, our algorithm is executed following the three steps: 1) setting a boundary on one of the two objectives, 2) searching for a deployment which minimises the other objective while the first objective is within the boundary set and 3) traversing the available options to optimise the deployment found in step 2.

In the first step we set a bound $\mathcal{C}$ on entropy such that:

$$R_E(\phi) \leq -nR_{max} \log \sqrt[n]{\frac{R_P(\phi^L) + R_P(\phi^M)}{2}}, \quad (10)$$
$$= -R_{max} \log \frac{R_P(\phi^L) + R_P(\phi^M)}{2} = \mathcal{C}$$

where ϕ^M is the deployment which has the maximum reliability rate (see equation (2)), and ϕ^L is the lower bound

deployment of monetary cost without considering reliability constraint. Therefore, $R_P(\phi^L)$ and $R_P(\phi^M)$ are the values of the power method. Where R_{max} is the service which has the maximum reliability rate, deploying over federated clouds and also meet the security requirements. n represents that the target workflow includes n number of services. We could also make a bound on cost, however, we need to guarantee that the reliability rate is acceptable. We also do not set the boundary on security because all candidate solutions we generated meet the security requirements W.

As mentioned above, our optimisation is an extension of MCKP. MCKP is used to optimise a set of decisions $SET = \{set_1, \ldots, set_n\}$ within a defined constraint. In our scenario, SET represents the set of deployments Φ, where set_i is a mapping of each workflow object e onto a cloud platform wp, $set_i = \{e_1^{wp_i}, \ldots, e_m^{wp_j}\}$. Our algorithm aims to find the optimised set_i.

To realise this goal, first, we generate the ϕ^L (lower bound) and ϕ^M (most reliable) deployments that both satisfy security requirements. Then, we set $\mathcal{C}$ as the reliability constraints and invoke a set of optimisation mechanisms to improve ϕ^M. Thus, our algorithm consists of two phases: initial deployment and deployment optimisation.

4.1 Initial Deployment

The goal of the initial deployment is to generate quickly, in linear time, a solution which can be a seed to find the optimal deployment. First, we produce the cost optimised deployment ϕ^L by applying the NCF algorithm from our previous work [21]. That algorithm works as follows: (1) a greedy-based function determines a deployment which accounts for the costs related to running on a cloud, with each service considered in isolation; (2) the algorithm then takes into account direct links between services and redeploys services such that the influence of the data transfer costs is minimised. In both steps services' security requirements are guaranteed.

Given ϕ^L generated by NCF, we compare its entropy value with constraint $\mathcal{C}$. If $R_E(\phi^L) < \mathcal{C}$, ϕ^L will be the optimal solution returned by our algorithm. Otherwise, we apply a greedy algorithm to nd the most reliable solution ϕ^M using the power method. The algorithm takes a list of the workow services and assigns each service to the most reliable cloud that meets the security constraints of that service. Once all services are assigned, ϕ^M becomes the seed for the next phase.

Note that ϕ^M is optimal, and can be generated by a greedy algorithm because calculating the reliability of a workow using the power method has the optimal substructure property if we assume fault-free connections between services; by maximising the reliability of a single service we obtain the optimal solution if the reliability of remaining services is also maximised. For the proof please see Appendix C, available in the online supplemental material.

4.2 Deployment Optimisation

If ϕ^L does not meet the reliability constraint, we use ϕ^M as the initial solution and the seed to find less costly options (cf. lines 1–2 in Algorithm 1). We limit the search to find t deployment options. Each new deployment is generated by function $Change(\phi, Cloud, M)$ (see Algorithm 3) and stored in M (lines 4-6). Once the loop is finished, M is sorted in the descending order by the cost and thus the best solution found is the last on the list.

Algorithm 1. Deployment Optimisation

Input:
- ϕ^M the power method optimal deployment;
- t the maximum number of deployments to search for;
- $Cloud$ a two dimensional array of clouds available for each service;

Output:
- M valid deployments, cheapest last;

1: $M[0] \leftarrow \phi^M$
2: $\phi \leftarrow \phi^M$
3: **for** $i \in 1...t$ **do**
4: $\quad \phi^{NEW} \leftarrow Change(\phi, Cloud, M)$;
5: $\quad \phi \leftarrow \phi^{NEW}$
6: $\quad M[i] \leftarrow \phi^{NEW}$
7: sort M by cost, descending
8: **return** $M[t]$

Finding t valid options was the key challenge in designing the algorithm because of the huge search space. To address it we observe the contribution of each cloud for each service taking into account its predecessors. To calculate the cost of deploying a service on a specific cloud we use the COD function (Eq. (11)). COD is calculated by adding the computing cost of service s_i and the transmission and storage cost of data sent from all s_i immediate predecessor services that are not in the same cloud,

$$COD(s_i^p) = SCOST(s_i^p) + CCOST(s_i^p) + ECOST(s_i^p). \tag{11}$$

We use the COD value in Algorithm 2 to rank valid clouds for each service; clouds are sorted by COD in the ascending order.

Algorithm 2. Rank Clouds by COD

Input and variables:
- ϕ a deployment of workflow w;
- $Cloud$ a two dimensional array of clouds available for each service in w;
- Tmp a list of key-value pairs $\langle cloud, COD \rangle$, one for each cloud available for a service;

Output:
- $Cloud$ the array of clouds with the reordered list of clouds for each service;

1: **function** Rank($\phi, Cloud$)
2: **for** $s_i \in topsort(\phi)$ **do**
3: $\quad$ **for** $p \in Cloud[s_i]$ **do**
4: $\quad\quad Tmp[p] \leftarrow \langle p, COD(s_i^p) \rangle$
5: $\quad$ sort Tmp in the ascending order by the COD value
6: $\quad Cloud[s_i] \leftarrow keys(Tmp)$
7: **return** $Cloud$

Function $topsort(\phi)$ in line 2 returns a topological order of workflow w. Lines 3-5 are used to calculate an ordered list of COD values for clouds available for service s_i. Finally,

TABLE 2
Number of Services in Each Pegasus Workflow for the Three Different Sizes

Workflow app.	Medium	Large	Very large
CyberShake	30	100	1,000
Montage	25	100	1,000
LIGO	30	100	1,000
Epigenomics	24	100	997

in line 6 the $Cloud$ array is updated with an ordered list of clouds available for service s_i (keys from the Tmp list), and returned in line 7.

The $Rank$ function is used by $Change$ (shown in Algorithm 3) to alter the given deployment. It moves a randomly selected service s (line 4) to another cloud (line 5). The choice of the cloud is also random but with the help of the $Benford$ function. This function is a transformation of Benford's law [23] defined as:

$$\begin{aligned} Benford(s) &= \left\lfloor \frac{1}{10^b - 1} \right\rfloor \\ b &= random\left(\log_{10}\left(1 + \frac{1}{len(Cloud[s])}\right), log_{10}2\right), \end{aligned} \tag{12}$$

where $len(Cloud[s])$ denotes the number of valid clouds for service s.

We used randomisation in the selection of the cloud because the optimal option is not always composed of the clouds which minimise the COD value. By applying the $Benford$ function, the clouds on the front of the ranking have higher chances to be selected but we also avoid the situation that some clouds are never chosen.

Algorithm 3. Change Deployment

Input:
- ϕ a deployment of workflow w;
- $Cloud$ a two dimensional array of clouds available for each service;
- M an array of valid deployments;

Output:
- ϕ^{NEW} a new deployment of w;

1: **function** Change($\phi, Cloud, M$)
2: $Cloud \leftarrow Rank(\phi, Cloud)$
3: **while** true **do**
4: $s \leftarrow$ {Choose a random service from ϕ}
5: $\phi^{NEW} \Leftarrow replace(\phi, s, Cloud[s][Benford(s)])$;
6: **if** $\phi^{NEW} \notin M$ **and** ϕ^{NEW} is secure **and** $H(\phi^{NEW}) \leq \mathcal{C}$ **then**
7: **return** ϕ^{NEW}

The last two lines of the algorithm (6-7) break the loop and return new deployment ϕ^{NEW} iff it is unique among deployments already found (array M) and meets the criteria with regards to security and reliability.

5 Results of the Experiments and Evaluation

In order to evaluate our algorithm we set up two experiments. First, we conducted a series of simulation experiments on a number of real scientific workflow applications.

TABLE 3
Cloud Pricing and Security Levels

VM	Loc.	Exec (/hour)	Store (/hour/GB)	In (/GB)	Out (/GB)	Start-up Time (hour)
C1	0	0.40	0.10	0	0.02	5.0
C2	2	2.20	0.60	0.03	0.01	3.0
C3	1	1.23	0.30	0.14	0.07	4.5
C4	2	3.70	0.60	0.10	0.05	2.5
C5	3	4.50	0.90	0.14	0.05	1.5
C6	4	5.50	1.30	0.14	0.13	0.5

In the second experiment we applied our algorithm over federated clouds to deploy a scientific workflow from one of the projects we have been involved in. Note that in this work, workflows are defined as directed acyclic graphs where vertices represent services and arcs between them represent data dependencies.

5.1 Simulation Environment

In this work, the experiments require to be repeatable in order compare and analyse different types of algorithms. Therefore, to evaluate our algorithm we combined together two simulation environments WorkflowSim and DynamicCloudSim.

WorkflowSim can simulate execution of workflows of the Pegasus workflow management system [24], whereas DynamicCloudSim adds to the cloud simulation instability and service failures. This combination can more realistically simulate workflow execution in dynamic cloud environments.

5.1.1 Experimental Setup

To evaluate our algorithm we consider four workflow applications available in the Pegasus project and in WorkflowSim: CyberShake (earthquake risk characterisation), Montage (generation of sky mosaics), LIGO (detection of gravitational waves), and Epigenomics (bioinformatics).[3] For each of these applications we selected three sizes: medium, large and very large, which determine the number of services the workflows include (Table 2). The full characterisation of these workflow applications can be found in [25]. In our simulation, however, we only need to consider the execution time, input and output data, and security level of each service. As the data privacy information for the workflows is unavailable, we randomly generated the security levels for the services.

To represent a federation of workflow platforms we created six VMs in six different data centres each with a random security level in range 0-4. Note that usually more secure clouds are also more expensive. Table 3 shows details of the platform setup with the cloud security level (*Location*) and cost of computation, storage, and incoming and outgoing communication. Additionally, *Start-up Time* indicates a randomly generated time when each WP has become available.

We compare our algorithm EMCK (extended multiple-choice knapsack) with DCA [26] and BDLS [17] which

3. The XML description files of the workflows are available via the Pegasus project pages at 'https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator'.

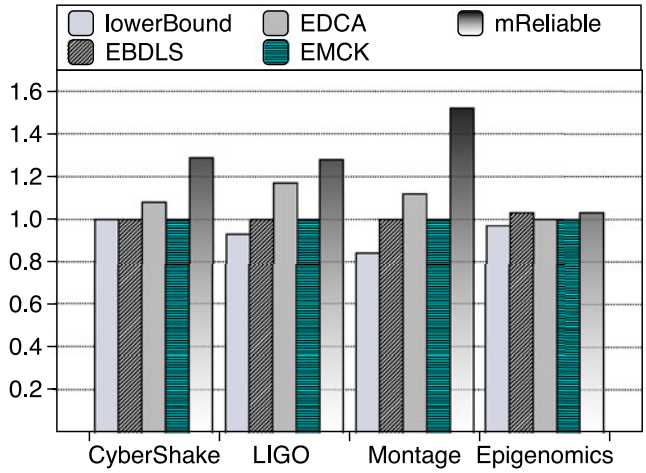

Fig. 2. Cost for the medium size workflows.

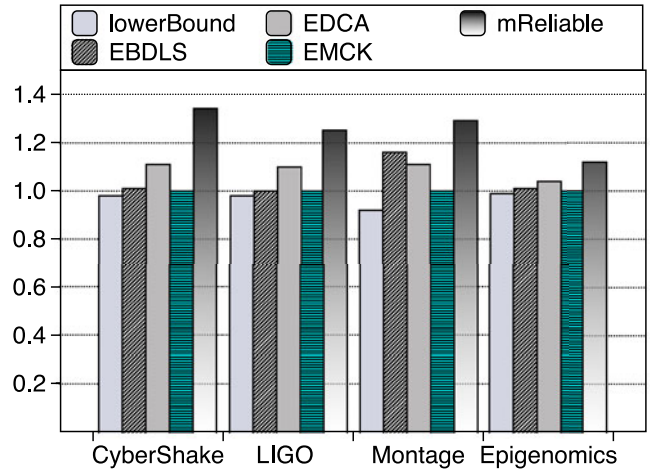

Fig. 4. Cost for the very large size workflows.

represent two existing and widely used multi-criteria scheduling algorithms. DCA and our algorithm both extend MCKP. However, DCA is focused on a single computing resource, and does not take into account the cost of data transfer and storage, so we adapted DCA to our model and called the extended version EDCA. BDLS is a list scheduling algorithm that schedules services according to a priority list of service-resource pairs. For the purpose of the experiment, we had to extend and adapt BDLS as follows: (1) we applied the ranking mechanism similar to Algorithm 3 to build a cost list; (2) we added another list with entropy constraint for each service. As a result the services using extended BDLS (EBDLS) were assigned to the cloud which minimised the cost and also met the entropy requirement.

The experiment results presented below are the average values of 1,000 deployments generated by each algorithm. The observed outputs are normalised against the results of the EMCK algorithm.

5.1.2 Monetary Cost and Time Complexity Evaluation

Figs. 2, 3 and 4 depict the normalised monetary cost for all twelve application setting with four types of workflows in three sizes mentioned previously. There are five different deployment solutions: EMCK, EBDLS, EDCA, lowerBound and mReliable. EMCK, EBDLS and EDCA represent the deployment solutions generated by the corresponding algorithms, whereas lowerBound (ϕ^L) and mReliable (ϕ^M) are cost optimised and most reliable deployments, respectively.

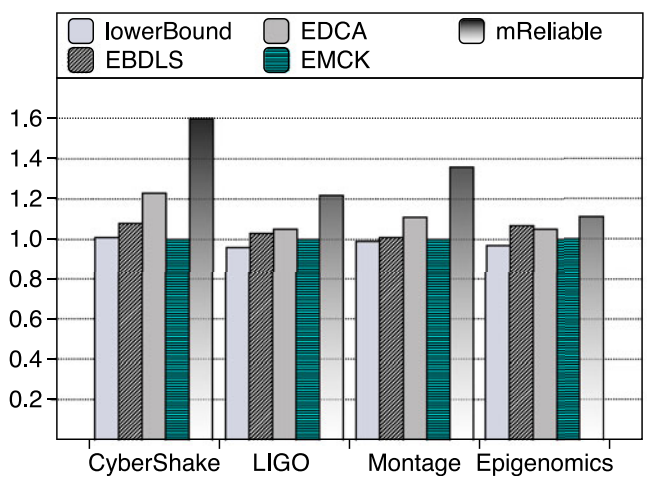

Fig. 3. Cost for the large size workflows.

The results show that EMCK almost always generates the cheapest deployment when compared with the other two algorithms. For one case (the large CyberShake workflow) it even produced deployments which were cheaper than our lowerBound result. The reason is that NCF, the algorithm used to generate the lowerBound solutions, is a heuristic-based algorithm [27] that can generate deployments based on cost and security yet it cannot guarantee the cheapest solution.

The performance of EBDLS is significantly influenced by the type and size of the workflow. For small and medium size workflows the cost of deployment solutions generated by EMCK and EBDLS is very close. However, for large CyberShare and Epigenomics workflows EMCK can save nearly 10 percent of cost when compared with EBDLS. Furthermore, for the very large Montage workflow EMCK yields 18 percent cost savings.

Fig. 5 shows the normalised time consumption of generating a deployment solution for different algorithms. EBDLS is hundreds times faster than EMCK and with the increasing size of the workflow the gap is also increasing. Similarly, EDCA is tens times faster. Although our algorithm is much slower than the two others, the time to generate a deployment, even for very large workflows, is less than one minute, which is acceptable for our use.

5.1.3 Reliability Evaluation

In order to simulate failures during workflow execution we combined WorkflowSim with DynamicCloudSim. DynamicCloudSim introduces a basic failure model which simulates the service fail during service execution. Whenever a service is assigned to a VM and its execution time is computed, DynamicCloudSim determines whether the service is bound to succeed or fail. This decision is based on the average rate of failure specified by the user [12].

Our experiment was set up as follows. First, we set the initial reliability rate of each cloud to 99.95 percent (The authors in [28] observed that a large proportion of fail events occur within 2 days in Google Cloud environment).

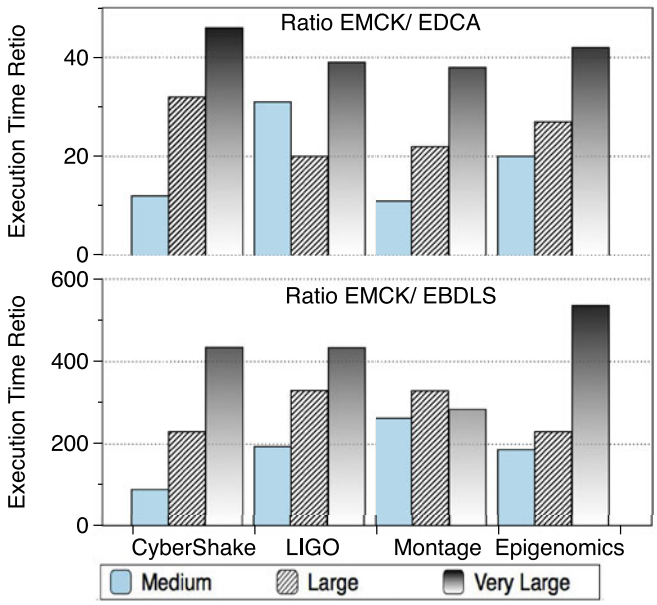

Fig. 5. Time ratio of EMCK to the other two algorithms.

Next, the VMs were initialised with security and cost parameters as shown in Table 3. The table includes also the start-up time of each VM, which is the time when a platform was started and is needed to calculate its reliability value at the moment when a workflow deployment occurs. Given the start-up time, the reliability rate of service s_i (or service in DynamicCloudSim) was $e^{-0.0005 \cdot t}$, where t was the start-up time of the VM assigned to execute s_i plus the execution time of s_i.

In this experiment we ran 1,000 executions of the Pegasus workflows deployed using EMCK and NCF which generates cost optimised deployments ("lowerBound"). The results are presented in Fig. 6 as the ratio of successful executions between these two algorithms. As shown, deployments generated by EMCK were always more reliable than the cost optimised ones. Furthermore, as workflows with more services have more chances to fail, with the increasing size of the workflows the ratio was more in favour of EMCK. For example, for the very large size of the "Epigenomics" workflow, EMCK avoided 80 percent of failures when compared with deployments generated by NCF.

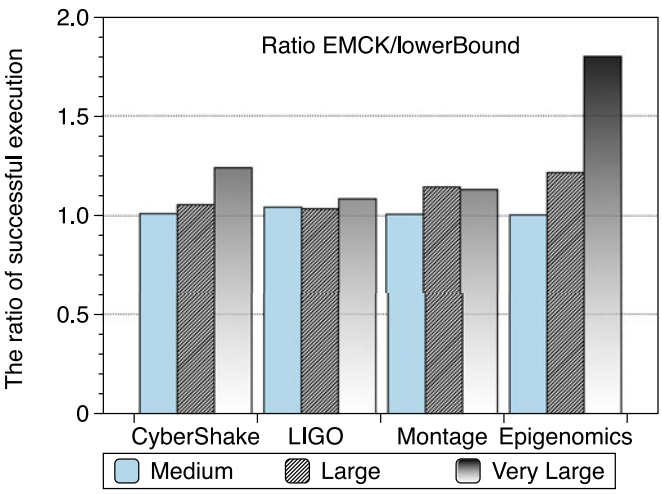

Fig. 6. Comparing successful executions between two types of deployments of workflows for different size.

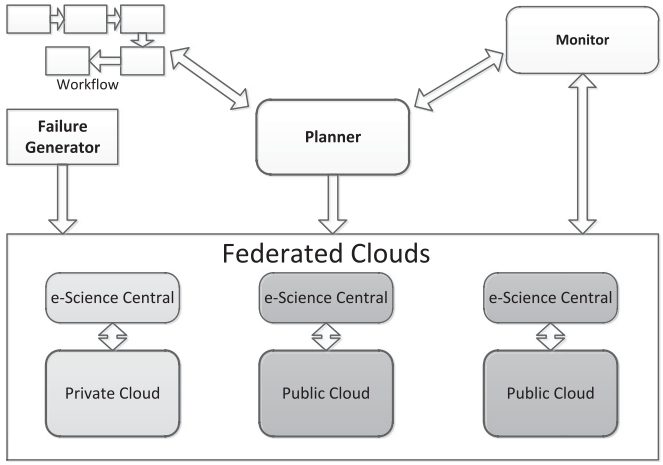

Fig. 7. The architecture of the deployment tool.

Importantly, workflow reliability was also influenced by the structure of the workflow. The "LIGO" workflow includes mostly short running services executed in parallel, whereas "Epigenomics" consists of chains of long running services. As a result, "Epigenomics" has much more chance to fail and so for this workflow the benefits of using EMCK were most prominent.

5.2 Realistic System

To evaluate our algorithm in conditions closer to a production use we applied it to schedule scientific workflows in e-Science Central [13]. e-Science Central (e-SC) is a cloud-based data analysis system that can run on a range of public and private clouds including Amazon AWS, Microsoft Azure and OpenShift. e-SC is a SaaS and PaaS, it offers a web user interface but also provides a range of APIs to allow users to control the system from code. For example, the storage subsystem API allows users to upload, download and manipulate data, whereas the workflow API enables them to execute, terminate and monitor workflow invocations. We used the APIs to create a tool that can orchestrate invocations of a single workflow partitioned over a number of e-SC instances.

5.2.1 Tool Design

Fig. 7 shows the architecture of our deployment tool. It consists of four core components: *Planner* assigns workflow partitions to *Federated Clouds* using the algorithm discussed above. The *Federated Clouds* consists of a set of e-Science Central instances running in different clouds. *Monitor* observes the status of each instance, detects failures, and provides the information about available instances to the *Planner*. Finally, *Failure Generator* is used to simulate failures by shutting down the e-SC instances with a predefined probability.

5.2.2 Experiment Setup

To verify our algorithm we selected one of the workflows used in the Cloud e-Genome project [29] (see Fig. 8). The project implements a whole exome sequencing pipeline

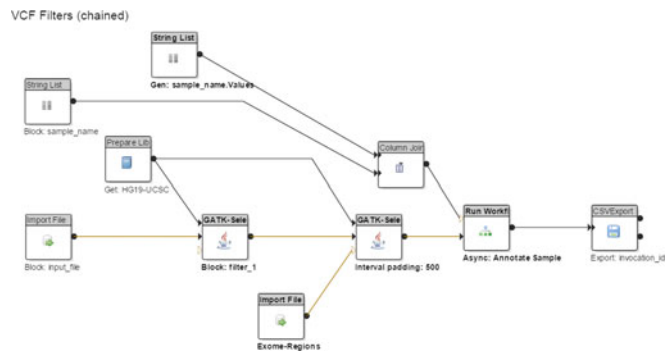

Fig. 8. A selected workflow from the Cloud e-Genome project.

TABLE 4
Services Representation and Security and Execution Time

Service	Name	Clearance	Location	Time(/h)
Sample_Name	S_1	1	0	1
ImportFile	S_2	1	0	1.5
Sample_Vaule Info	S_3	1	0	3
*HG*19	S_4	1	0	0.1
Filter	S_5	2	0	10
$Exome - Regions$	S_6	1	0	7
Intervalpadding	S_7	0	0	20
ColumnJoin	S_8	2	0	0.1
AnnotateSample	S_9	2	0	5
Export	S_{10}	1	0	0.3

TABLE 5
Data Security and Size

Data	Location	Size (GB)
$S_{1,8}$	1	0
$S_{2,5}$	0	1.1
$S_{3,8}$	2	0.01
$S_{4,5}$	0	0.005
$S_{4,7}$	0	0.005
$S_{5,7}$	0	6.2
$S_{6,7}$	0	10.3
$S_{7,9}$	1	3.6
$S_{8,9}$	0	0
$S_{9,10}$	0	0.05

using e-Science Central workflows deployed on Microsoft Azure.

Guaranteeing that human genomic data can be securely processed on the cloud is very important. Therefore, we modelled the security requirements of a selected Cloud e-Genome workflow by assigning security levels as shown in Tables 4 and 5. Note that the size of data transferred between blocks and the execution time of each block are actual values taken from provenance logs collected by e-SC. Other work, e.g., [30], [31], [32], describe how the size and time may be estimated using execution logs and workflow input data.

Table 5 shows data sizes in GB, where 0 denotes less than 1 MB of data. The pricing shown in Table 6 is collected from two major cloud providers and is based on the equivalent VM configurations. Also one of the providers offered a private cloud service and we include its pricing in the table.

TABLE 6
Basic Attributes of the Three Clouds Used in the Experiment: Security Level, Cost of Computing Resources, Cost of Data Transfer Between Clouds (e.g., $Pr1 \rightarrow Pu1 = 0.1$)

Cloud	Pr1	Pu1	Pu2
Security	2	1	0
CPU	3.41(/h)	2.40(/h)	1.28(/h)
Pr1	0	0.1	0.11
Pu1	0.13	0	0.09
Pu2	0.07	0.02	0

TABLE 7
Two Deployments

Service	lowerBound	EMCK
S_1	Pr1	Pu1
S_2	Pu2	Pu2
S_3	Pr1	Pr1
S_4	Pu2	Pu2
S_5	Pu2	Pu1
S_6	Pu2	Pu2
S_7	Pu2	Pu2
S_8	Pr1	Pr1
S_9	Pu2	Pu2
S_{10}	Pu2	Pr1

To simulate this environment we set up three virtual machines each running a single instance of the e-SC system. VM1 was hosted on a personal PC and represented the private cloud. Two other VMs were hosted in our University virtualised environment and played the role of public cloud providers *Pu1* and *Pu2*.

As previously, to test our algorithm the platform's start-up time must be defined. We set it for VM1, VM2 and VM3 as 1.7, 2.4 and 3.5 h, and their initial reliability as 0.95, 0.93 and 0.90, respectively. In this experiment failure rates had to be unrealistically high because we wanted workflows to fail frequently. Otherwise, we would need to run the experiments for a prohibitively long time. Also, for the purpose of the experiment we reduced the execution time of given workflow to about 30 seconds by scaling down the amount of input data shown in Table 5 by a factor of 2,400.

5.2.3 Results and Discussion

Based on the presented experiment setup we generated two deployments: "lowerBound" produced by the NCF algorithm and one generated by EMCK (Table 7). The cost of the workflow execution was 69.832 for the "lowerBound" and 128.897 for EMCK. Although nearly twice as costly as the cost optimised solution, the deployment produced by our algorithm improved reliability by about 25 percent (Fig. 9).

One may challenge that the failure rate for both deployments were too high (about 40 percent). This is, however, the result of the high initial failure rate set for each VM. If the initial rate of each VM was set as the realistic value, there are lack of failures during the execution of the given workflow. This is not only because of the size of workflow is too small, but also the start-up time of each VM is just a few hours. However, the experiment in which we simulate more realistic reliability conditions were presented in the

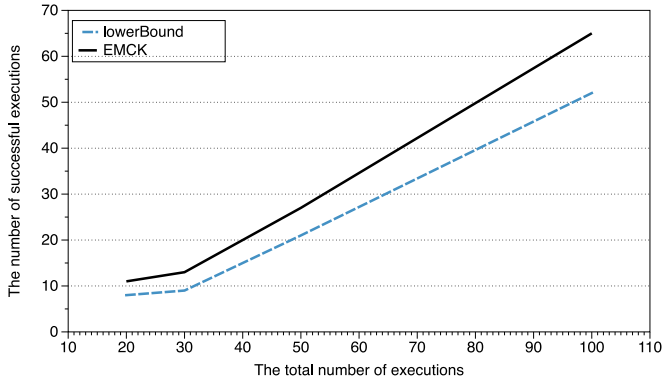

Fig. 9. The execution results of two deployments.

previous section, whereas in this experiment we show that our algorithm can be adapted to the real system and can generate correct deployments.

6 Related Work

Workflow scheduling has been a classic research topic for decades and has developed together with changes in the technology. In the last decade, most work has focused on workflow mapping problems using DAG scheduling heuristics such as [33], [34] and [35], to name just a few. However, these algorithms are all based on single computing resources such as "free" grid resources, and thus aim to minimise makespan. Furthermore, there are a few works which consider other scheduling with other objectives such as security, reliability and performance. Song et al. introduces in [36] a version of a genetic algorithm to assign jobs based on risk-resilient strategies to provide security assurance of trusted Grid computing. The algorithm in [37] tolerates processor failure by means of primary/backup techniques to allocate two copies of each service to different processors.

In [38] Xie and Qin model the security needs of the real-time application on clusters, design and implement a scheduling algorithm, including the security-aware heuristic strategy. In contrast, in our work we consider a problem with three objectives in which we optimise monetary cost, whilst reliability and security are maintained according to the user requirements.

Multi-objective optimisation has been explored previously in heterogeneous computing. The authors of paper [18] minimise the makespan and failure probability by combining both objectives into a single cost function to form a new matching and scheduling algorithm. In paper [39], the authors distinguish the reliability maximisation and execution time minimisation mappings and then provide a method for converting workflow scheduling heuristics on heterogeneous clusters into heuristics that take reliability into account. The work in [17] presents two algorithms to address the trade-off between makespan and reliability. One is based on a dynamic level scheduling algorithm and the other is a version of a genetic algorithm. Instead, our work presents a technique to deploy workflows on federated clouds where the monetary cost plays a crucial role, and none of the mentioned algorithms considers this.

Research related to federated clouds or multi-cloud environments is still new with little literature available. For example, in [40] the authors introduced a pricing model and truthful mechanism for scheduling workflow applications to different clouds. They consider a scheduling problem with respect to two objectives: makespan and monetary cost. However, the reliability and security are not taken into account. Moreover, Tang et al. addressed in [41] the reliability of a distributed workflow application. It uses a linear utility function to represent the problem and then optimises this function. However, as we mention in Introduction, we follow the other approach: to optimise one criteria and keep others at an acceptable level.

In [42] Fard et al. attempt to optimise more than two objectives but, similarly to BDLS, their algorithm uses a heuristic based on list scheduling. Using priority list is very effective when applied to a set of independent tasks. However, for a workflow of interdependent tasks it shows limitations (cf. Section 5.1.2 and Figs. 2, 3, 4). Additionally, the authors use the power method to measure reliability, whereas we introduce and utilise a more advantageous entropy-based method.

Finally, Web Services has been developed for about two decades and resulted in a number of service selection algorithms that are related to our work, such as [14], [43] and [44] to list a few. Although service selection and workflow deployment are similar problems, the former focuses on grouping business processes, whereas we address deployment of scientific workflows. Scientific workflows often require large amount of data to be transferred and thus factors such as the cost of data transfer between clouds require careful consideration. Moreover, our security model inspired by the Bell-LaPadula model makes it different from other work in this area.

7 Conclusion

Driven by the popularity of moving to cloud, increasing number of workflow based applications are hosted in cloud. However, the variety and dynamics of cloud environments with security, reliability and price, which make the users struggle in choosing a best cloud for deploying their applications. In this paper, we presented a new algorithm to address the problem of deploying workflow applications over federated clouds meeting the reliability, security and monetary cost requirements. We have shown the trade-off between reliability and cost, and the optimisation problem is NP-hard problem. Our algorithm guarantees that the security and reliability constraints are met while optimising the monetary cost. The algorithm has been evaluated using realistic scientific workflows on both simulated environment and a real world cloud based platform. Experimental results show that our solutions can guarantee the security and reduce 25 percent failures while generating the cheapest solution comparing with other algorithms.

Future work will address remaining opportunities including: how to include performance requirements, and how to benefit from new cloud resources that become available during workflow execution? Performance and scalability are among the main factors that attract users to cloud computing, and so incorporating mechanisms and algorithms that address these aspects of cloud is our next focus.

ACKNOWLEDGMENTS

This work was supported by the RCUK Digital Economy Theme [grant number EP/G066019/1 - SIDE: Social Inclusion through the Digital Economy].

REFERENCES

[1] (2015). Cloudrfp.com. [Online]. Available: https://www.cloudrfp.com/company/browse
[2] (2011). Summary of the amazon ec2 and amazon rds service disruption in the us east region. [Online]. Available: http://aws.amazon.com/message/65648/
[3] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, May 2010, pp. 513–517.
[4] P. Watson. (2012). A multi-level security model for partitioning workflows over federated clouds. *J. Cloud Comput.* [Online]. *1(1)*, pp. 1–15. Available: http://dx.doi.org/10.1186/2192-113X-1-15
[5] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," MITRE Corporation, Bedford, MA, Tech. Rep. MTR-2547, vol. 1, Mar. 1973.
[6] D. Hollingsworth, "Workflow management coalition - The workflow reference model," Workflow Management Coalition, 2 Crown Walk Winchester Hampshire, UK, Tech. Rep. TC00-1003, 1995.
[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.
[8] C. E. Shannon. (2001, Jan.). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.* [Online]. *5(1)*, pp. 3–55. Available: http://doi.acm.org/10.1145/584091.584093
[9] M. R. Garey and D. S. Johnson, *Computers Intractability: Guide Theory NP-Completeness*. New York, NY, USA: Freeman, 1979.
[10] Y. Klein and G. Langholz, "Multi-criteria scheduling optimization using fuzzy logic," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1998, vol. 1, pp. 445–450.
[11] W. Chen and E. Deelman. (2012). Workflowsim: A toolkit for simulating scientific workflows in distributed environments. in *Proc. IEEE 8th Int. Conf. E-Sci.*, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/eScience.2012.6404430
[12] M. Bux and U. Leser, "DynamicCloudSim: Simulating heterogeneity in computational clouds," *Future Gener. Comput. Syst.*, vol. 46, no. 0, pp. 85–99, 2015.
[13] H. Hiden, S. Woodman, P. Watson, and J. Cała, "Developing cloud applications using the e-Science Central platform," *Roy. Soc. London. Philosoph. Trans. A. Math., Phys. Eng. Sci.*, vol. 371, p. 20120085, 2013.
[14] Z. Zheng and M. Lyu, "Selecting an optimal fault tolerance strategy for reliable service-oriented systems with local and global constraints," *Computers*, vol. 64, no. 1, pp. 219–232, Jan 2015.
[15] E. L. Rhys Lewis, *Introduction Reliability Engineering*. Hoboken, NJ, USA: Wiley, Nov. 15, 1995.
[16] M. Hakem and F. Butelle, "Reliability and scheduling on systems subject to failures," in *Proc. Int. Conf. Parallel Process.*, Sep. 2007, pp. 38–38.
[17] A. Doğan and F. Özgüner. (2005, May). Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems*. *Comput. J.* [Online]. *48(3)*, pp. 300–314. Available: http://dx.doi.org/10.1093/comjnl/bxh086
[18] A. Dogan and F. Ozguner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 308–323, Mar. 2002.
[19] D. E. Denning. (1976, May). A lattice model of secure information flow. *Commun. ACM* [Online]. *19(5)*, pp. 236–243. Available: http://doi.acm.org/10.1145/360051.360056
[20] B. Hendrickson, R. Leland, and R. V. Driessche. (1997). Skewed graph partitioning. in *Proc. 8th SIAM Conf. Parallel Process. Sci. Comput.* [Online]. Available: http://www.sandia.gov/~leland/documents/Skewed-Graph-Partitioning.pdf
[21] Z. Wen, J. Cała, and P. Watson, "A scalable method for partitioning workflows with security requirements over federated clouds," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Tech. Sci.*, Dec. 2014, pp. 122–129.
[22] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Berlin, Germany: Springer-Verlag, 2004.
[23] D. Jang, J. U. Kang, A. Kruckman, J. Kudo, and S. J. Miller, "Chains of distributions, hierarchical Bayesian models and benfords law," *ArXiv e-print*, May 2008, http://adsabs.harvard.edu/abs/2008arXiv0805.4226J
[24] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger. (2014). Pegasus, a workflow management system for science automation. *Future Gener. Comput. Syst.* [Online]. *46*, pp. 17–35. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167739X14002015
[25] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, 2013.
[26] R. Prodan and M. Wieczorek, "Bi-criteria scheduling of scientific grid workflows," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 2, pp. 364–376, Apr. 2010.
[27] B. Shirazi, M. Wang, and G. Pathak, "Analysis and evaluation of heuristic methods for static task scheduling," *J. Parallel Distrib. Comput.*, vol. 10, no. 3, pp. 222–232, 1990.
[28] P. Garraghan, P. Townend, and J. Xu, "An empirical failure-analysis of a large-scale cloud computing environment," in *Proc. IEEE 15th Int. Symp. High-Assurance Syst. Eng.*, Jan. 2014, pp. 113–120.
[29] J. Cała, Y. X. Xu, E. A. Wijaya, and P. Missier, "From scripted HPC-based NGS pipelines to workflows on the cloud," in *Proc. C4Bio Workshop, Co-Located 2014 CCGrid Conf.*, 2014, pp. 694–700.
[30] M. Dobber, R. van der Mei, and G. Koole, "Effective prediction of job processing times in a large-scale grid environment," in *Proc. 15th IEEE Int. Symp. High Perform. Distrib. Comput.*, 2006, pp. 359–360.
[31] T. Miu and P. Missier, "Predicting the execution time of workflow activities based on their input features," in *Proc. High Perform. Comput., Netw., Storage Anal.*, Nov. 2012, pp. 64–72.
[32] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, and T. Fahringer, "A hybrid intelligent method for performance modeling and prediction of workflow activities in grids," in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, May 2009, pp. 339–347.
[33] H. Topcuouglu, S. Hariri, and M.-Y. Wu. (2002, Mar.). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* [Online]. *13(3)*, pp. 260–274. Available: http://dx.doi.org/10.1109/71.993206
[34] I. Ahmad and Y.-K. Kwok, "A new approach to scheduling parallel programs using task duplication," in *Proc. Int. Conf. Parallel Process.*, Aug. 1994, vol. 2, pp. 47–51.
[35] B. Kruatrachue and T. Lewis, "Grain size determination for parallel processing," *IEEE Software*, vol. 5, no. 1, pp. 23–32, Jan. 1988.
[36] S. Song, K. Hwang, Y. kwong Kwok, and S. Member, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling," *Computers*, vol. 55, no. 6, pp. 703–719, Jun. 2006.
[37] X. Qin and H. Jiang. (2006, Jun.). A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Comput.* [Online]. *32(5)*, pp. 331–356. Available: http://dx.doi.org/10.1016/j.parco.2006.06.006
[38] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *Computers*, vol. 55, no. 7, pp. 864–879, Jul. 2006.
[39] J. J. Dongarra, E. Jeannot, E. Saule, and Z. Shi. (2007). Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. in *Proc. 19th Annu. ACM Symp. Parallel Algorithms Archit.*, pp. 280–288. [Online]. Available: http://doi.acm.org/10.1145/1248377.1248423
[40] H. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1203–1212, Jun. 2013.
[41] X. Tang, K. Li, M. Qiu, and E. H.-M. Sha. (2012). A hierarchical reliability-driven scheduling algorithm in grid systems. *J. Parallel Distrib. Comput.* [Online]. *72(4)*, pp. 525–535. Available: http://www.sciencedirect.com/science/article/pii/S0743731511002346
[42] H. M. Fard, R. Prodan, and T. Fahringer. (2014). Multi-objective list scheduling of workflow applications in distributed computing infrastructures. *J. Parallel Distrib. Comput.* [Online]. *74(3)*, pp. 2152–2165. Available: http://www.sciencedirect.com/science/article/pii/S0743731513002384
[43] H. Zo, D. L. Nazareth, and H. K. Jain, "Measuring reliability of applications composed of web services," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2007, pp. 1–10.
[44] S. Ran. (2003, Mar.). A model for web services discovery with qos. *SIGecom Exch.* [Online]. *4(1)*, pp. 1–10. Available: http://doi.acm.org/10.1145/844357.844360

Zhenyu Wen received the BE degree in computer science and technology from Zhejiang Gongshang University, Zhejiang, China, in 2009, and the MS and PhD degrees in computer science from Newcastle University, Newcastle Upon Tyne, United Kingdom, in 2011 and 2015, respectively. He is currently a PostDoctoral researcher with the School of Informatics, the University of Edinburgh, Edinburgh, United Kingdom. He has authored a number of research papers in the field of cloud computing. His current research interests include multi-objects optimisation, crowdsources, artificial intelligent and cloud computing.

Jacek Cała received the PhD degree in computer science from the AGH-University of Science and Technology in 2010. He is a research associate in the School of Computing Science at Newcastle University. Recently, he has been involved in development of workflow-based systems using e-Science Central, private, and public cloud platforms. His main interests include cloud-based, component-based and distributed solutions driving workflows and e-Science. Previously, he was a teaching and research assistant at the AGH-University of Science and Technology in Kraków, Poland. There he was one of the architects and key developers of TeleDICOM, a system which supports medical teleconsultations in more than 20 hospitals and medical centres in the South of Poland.

Paul Watson graduated in 1983 with a BSc in computer engineering from Manchester University, followed by the PhD degree on parallel graph reduction in 1986. He is a professor of computer science and a director of the Digital Institute at Newcastle University United Kingdom. He also codirects the EPSRC Centre for Doctoral training in cloud computing for big data. In the 80s, as a lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990 to 2005, he was with ICL as a system designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995, he moved to Newcastle University, where he has been an investigator on research projects in the area of scalable information management. He is a chartered engineer, a fellow of the British Computer Society, and a member of the UK Computing Research Committee. He received the 2014 Jim Gray eScience Award.

Alexander Romanovsky is a professor with the School of Computing Science (Newcastle University, United Kingdom) and the leader of the School's Secure and Resilient Systems Group. He coordinates the EPSRC platform grant on Trustworthy Ambient Systems and the RSSB SafeCap+ grant on railway integrated optimum capacity, safety and energy strategies. He is a coinvestigator of the EPSRC PRiME programme grant on power-efficient, reliable, many-core embedded systems. Before this, he coordinated the major EU FP7 DEPLOY IP that developed the Rodin tooling environment for formal stepwise design of complex dependable systems using Event-B. Rodin is now widely used by companies in Europe, China, Japan, USA, Canada, and Brazil. His main research areas are system dependability, fault tolerance, safety, modelling, and verification.