Business-driven Service Placement for Highly Dynamic and Distributed Cloud Systems

Mauro Tortonesi, *Member, IEEE,* Luca Foschini, *Member, IEEE*

Abstract—The emergence of large-scale Cloud computing environments characterized by dynamic resource pricing schemes enables valuable cost saving opportunities for service providers that could dynamically decide to change the placement of their IT service components in order to reduce their bills. However, that requires new management solutions to dynamically reconfigure IT service components placement, in order to respond to pricing changes and to control and guarantee the high-level business objectives defined by service providers. This paper proposes a novel approach based on Genetic Algorithm (GA) optimization techniques for adaptive business-driven IT service component reconfiguration. Our proposal allows to evaluate the performance of complex IT services deployments over large-scale Cloud systems in a wide range of alternative configurations, by granting prompt transitions to more convenient placements as business values and costs change dynamically. We deeply assessed our framework in a realistic scenario that consists of 2-tier service architectures with real-world pricing schemes. Collected results show the effectiveness and quantify the overhead of our solution. The results also demonstrate the suitability of business-driven IT management techniques for service components placement and reconfiguration in highly dynamic and distributed Cloud systems.

Index Terms—Optimization of Services Systems; Services Management; Bridging Business and IT Architecture; Installation Management; Simulation, Modeling, and Visualization.

1 Introduction

NOVEL Cloud computing infrastructures, consisting of worldwide fully interconnected data centers offering their computational resources on a pay-per-use basis, are opening brand new challenges and opportunities to develop sophisticated services and IT management systems on a global scale. These complex Cloud systems typically involve three main types of actors: *service users*, *service providers*, and *Cloud providers*. Service users are the final clients that require access to specific online services. Service providers seize the opportunity to build new services in order to increase their economical revenue and externalize the execution of their own services to avoid the deployment of costly private IT infrastructures. Finally, Cloud providers are usually big players, such as Amazon, Google, and IBM, that offer service providers all the system resources needed to execute their services on a pay-per-use basis.

In recent years, many Cloud providers, such as Amazon Elastic Compute Cloud (EC2) Spot Instance Service[1], have introduced dynamic resource pricing schemes. In fact, Cloud providers typically benefit of underutilized resources at their different data centers, which they attempt to sell by implementing temporary price reduction and bidding schemes. As a consequence, resource prices are not only different between different Clouds, but also fluctuate over time depending on the different data center, even for the same Cloud provider. Those new dynamic pricing schemes would pave the way to significant operation money savings if service providers were able to dynamically and self-adaptively (re-)place and operate their services to the data center that, at any time, grants needed service levels and the best economic advantages.

Among the several open management issues that have to be solved to take full advantage of those new possibilities, this paper originally focuses on the problem of enabling placement computation of complex services that require the placement of multiple components and virtual resources (e.g., Virtual Machines - VMs, storage, networking, etc.) in highly dynamic and large-scale Cloud environments. This specific problem has already been addressed by some other works that all share the common purpose of balancing the Cloud provider internal objectives, namely minimizing infrastructure/hardware resource usage and granting Service Level Agreement negotiated with Service Providers (SLA_{SP}), and the external objectives of service providers using the Cloud, typically cost minimization and fulfillment of SLA signed with their Service Users (SLA_{SU}). However, most of the works available in the literature tend to focus more on internal objectives with different goals: to minimize energy consumption either in a single data center or in distributed multiple ones [1, 2, 3]; to balance incoming load to prevent resource shortages [4, 5]; and to reorganize service schemes in order to deliver agreed SLAs and to let service

- *M. Tortonesi is with the Engineering Department, University of Ferrara, Via Saragat 1, 44122, Ferrara, Italy. E-mail: mauro.tortonesi@ unife.it.*
- *L. Foschini is with the Department of Computer Science and Engineering, University of Bologna, Mura Anteo Zamboni 7, 40126, Bologna, Italy, E-mail: luca.foschini@unibo.it.*

[1] Amazon EC2 Spot Instance Service is available at: http://aws.amazon.com/ec2/spot-instances/

providers express specific placement constraints, usually considering traditional IT performance metrics [6, 7].

The optimization of external objectives, instead, apart a few specific seminal studies [8, 9], is still widely unexplored. In this area, we will focus on the service provider perspective to address two specific business cases that have motivated our project. The first one is aimed to help service provider executives by assisting them in the optimization of service allocation, though minimization of operational and business-related service costs. Those charges include: the cost of each Cloud component, automatically accountability according to the pay-per-use cost model; SLA_{SU} violation penalties as in the contract agreed with the service users; and possible delays when the service is moved and operated elsewhere for the sake of cost savings. The second one, instead, focuses on service provider business managers with the goal to provide them with tools for calculating and verifying the alignment of IT service infrastructure with business criteria. That evaluation involves, on the one hand, risk management aspects, such as, fault-tolerance and reliability, tendency to under-dimension IT services, and likelihood of SLA_{SU} violations; on the other hand, even intangible aspects, such as, service user satisfaction, and vendor lock-in [10].

These use-cases raise a series of challenges regarding business-driven IT management, namely, the practice in IT service management that attempts to evaluate and optimize the IT infrastructure according to business criteria, in Cloud systems:

- How can we model services operated by the service provider in the Cloud?
- How do we keep up with the highly dynamic resource pricing schemes offered by Cloud providers so to minimize service provider operational costs?
- How do we model service provisioning and user demand to evaluate feasible and realistic service placement for different possible deployment scenarios?
- How can we enable fast computation of future service placement for fast alignment of service deployment to actual operational costs, SLAs, and business criteria?

To answer those open technical challenges, this paper proposes a novel adaptive and business-driven service placement solution for Cloud computing environments that significantly enhances our recent previous work in this field [11], and shows three main original technical aspects. First, it considers *business-driven metrics* in order to evaluate the alignment of service placement (for all interacting components) with the business objectives set by the IT management of the service provider; in particular, with respect to [11] it presents an much in-depth discussion of the employed service modeling and allows to identify the configuration with the lowest business impact. Second, it adopts a *simulative approach* to reenact IT services under different configurations, thus providing a much better capability to accurately capture peculiar behavior of real-life IT services than analytic methods. Third, it leverages on meta-heuristics based on *Genetic Algorithm (GA) optimizations* to enable robust and resilient exploration of the large and dynamically-changing space of possible IT service component placement configurations by also granting fast and effective reactions to changes, such as demand load and VM cost variations.

In order to better underline the benefits and original aspects of the proposed solution and to demonstrate the effectiveness of our solution, the paper presents a thorough experimental evaluation, much more wider and detailed than the one in [11], based on a realistic 2-tier service scenario and on real costs for a large-scale Cloud computing environment implemented on top of 3 different Amazon EC2 data centers. Let us stress that this scenario represents a very significant business use case because it allows to capture the simple yet effective distributed architecture adopted by many service providers, including several successful startups using the Cloud to operate their infrastructure, as in the case of the seminal Instagram architecture that in 2012 was essentially based on 2 tiers[2].

2 Business-Driven Service Placement

The possibility to quickly increase or decrease the pool of virtual resources used to implement an IT service enables service providers to dynamically reconfigure the architecture of their IT systems, and to retune their performance. That presents many opportunities to rescale IT services to match both Cloud providers' offers of virtual resources and users' demand, enabling service providers to reap significant savings by allocating virtual resources when needed and at the best possible cost.

So far, most approaches in IT service configuration optimization in Cloud environments focused on achieving IT-level objectives, typically for SLA_{SP} enforcement purposes. This effectively means evaluating IT service configurations *from an internal IT perspective*, a conceptually straightforward but in practice very difficult task which requires to consider many IT-level performance metrics. Since those metrics are often very difficult to aggregate, this calls for the adoption of relatively sophisticated, and possibly brittle, multi-objective optimization methods.

The pay-per-use pricing schemes of Cloud computing, instead, facilitate the accurate estimation of IT costs, thus presenting the natural opportunity to evaluate IT service architectures *from the external business perspective* of service providers. This fosters service providers to reconfigure their IT services in order to minimize IT costs, and consequently to maximize revenues. However, this represents a straightforward, but rather naïve evaluation criterion that does not consider business aspects such as risk management and other intangibles, thus enforcing an excessively shallow business-level perspective [12, 13].

To this end, *business impact analysis* techniques represent a significantly better criterion for service providers to adopt for the performance optimization of their (virtual) IT infrastructures. Business impact-driven optimization aims at aligning IT service configurations with the business objectives of service providers, possibly even

[2] Instagram high scalability architecture, article available at: http://highscalability.com/blog/2012/4/9/the-instagram-architecture-facebook-bought-for-a-cool-billio.html

through the adoption of business management-specific techniques such as the Balanced Scorecard or the Analytic Hierarchy Process [14, 15].

Indeed, business impact represents a very convenient criterion from the optimization perspective, because it enables to evaluate all different aspects of IT service configurations through the same (monetary cost) metric in the optimization process. That enables an easier and more robust aggregation of different measured values, and allows the adoption of significantly simpler and more efficient optimization algorithms than complex multi-objective methods. Finally, business impact analysis does not require to consider explicit constraints on allowed IT service configurations (e.g., a maximum threshold for service response times) because undesired service configurations will have a high business impact and therefore will be automatically ruled out by the optimization process.

Worldwide Cloud systems raise several complexities from the configuration optimization perspective because services are typically deployed on a global scale. In addition, there is the need to consider an accurate model of virtual resource allocation that takes into account multiple service users and multiple Cloud data centers with different pricing schemes. Finally, it should consider the geographical distance between the instantiated virtual resources and the corresponding service users' location and the possibility to migrate service components between different Cloud data centers.

This raises the opportunity to design and develop sophisticated Cloud-based IT service optimization solutions capable of reenacting alternative IT service configuration through what-if scenario analysis, of evaluating their business-level performance, and of identifying the most convenient one. To this end, in this paper we propose *Business-Driven Management as a Service (BDMaaS)*, an adaptive and business-driven service placement and reconfiguration solution for IT services; Fig. 1 shows how BDMaaS framework integrates with existing Cloud systems and actors.

BDMaaS agents (deployed at provider data centers) are in charge of gathering data center resource availability, pricing schemes, and resource costs proposed by Cloud providers and of activating VMs at the target data centers. *BDMaaS engine* analyses the business impact of possible service placement adjustments and takes deployment decisions. *BDMaaS REpresentational State Transfer (REST) based Application Programming Interfaces (APIs)* are offered by our support to service providers to let them specify all needed input data, such as service and demand profile models, business goals and costs, etc., and to verify the alignment of adopted deployments with their business criteria. Finally, BDMaaS bus glues together all main BDMaaS components by enabling the necessary communication substrate. In the remainder of this article, we focus mainly on the BDMaaS engine that represents the core and more complex component of the whole BDMaaS architecture.

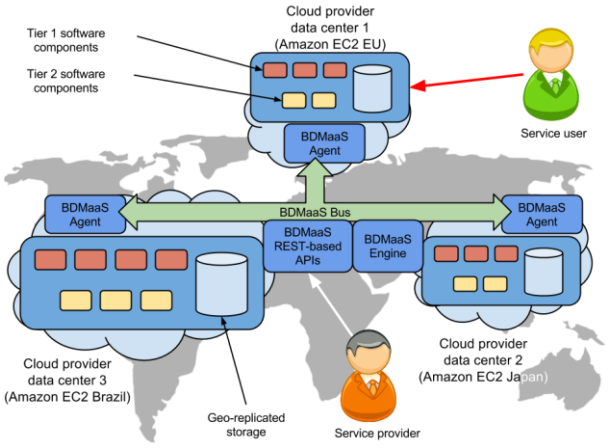

Fig. 1. BDMaaS distributed architecture and integration with existing Cloud systems and actors.

3. The BDMaaS Framework

The overall goal of the BDMaaS framework is the evaluation of the service placement that minimizes the business impact of the deployment with respect to all service provider operational and non-operational business criteria.

For the sake of simplicity, and without hindering the generality of the proposed approach, BDMaaS currently focuses on VMs as the basic building blocks for the realization of complex IT services. In other words, BDMaaS conceptually operates at the Infrastructure-as-a-Service (IaaS) level with the main goal of finding the best placement configuration of the VMs in the distributed Cloud environment.

Fig. 2 better details the internal architecture of the BDMaaS framework. Service providers interact with the BDMaaS engine using the BDMaaS REST-based APIs that includes two components, namely, Configuration Management and Policy Management. These components allow service providers to enter a configuration of the Cloud computing environment (e.g., number of data centers, service model, etc.) and to select the optimization policies to apply (e.g., business objectives, parameters for the optimization algorithm, etc.).

Focusing on the BDMaaS engine, it consists of three main stages that work in a pipeline, namely, modeling, optimization, and decision making. At the modeling stage, Demand Model and Service Model are the two main components that provide, respectively, the functions for building the models of the service user service request arrival process (e.g., customers' locations, distributions of service request inter-arrival times, etc.) and of the IT service execution (e.g., service time distribution, current service component placement, etc.).

The optimization stage consists of the Optimization macro-component and represents the core part of BDMaaS. It is in charge of reenacting the Cloud computing IT service and of evaluating possible alternative service placement configurations according to optimization policies selected by service providers and to the service and demand models provided by the previous stage. First, the Service Placement Simulation component mim-

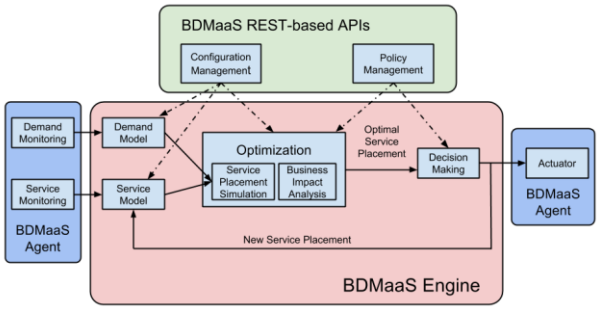

Fig. 2. The internal architecture of the BDMaaS framework.

ics possible service placements among those generated by the modeling stage. Then, the Business Impact Analysis component assigns an overall cost (namely, business impact) to each of these possible configurations.

At the third stage, the Decision Making component is in charge of selecting the best IT service placement configuration, namely, the one minimizing the business impact, according to the user preferences and the output data provided by the Optimization component. Finally, BDMaaS was designed to be easily integrated with existing Cloud-based IT services through lightweight BDMaaS agents installed at each data center. Each agent includes three relatively simple and implementation-specific "connector" components: Demand Monitoring, Service Monitoring, and Actuator, depicted in green in Fig. 2.

The Demand Monitoring component continuously analyzes service user request logs and feeds the Demand Model component with accurate and up-to-date data. Similarly, the Service Monitoring component integrates with real-life application performance and virtual resource cost monitoring tools to update the parameters of the service execution model used in the simulations. Finally, the Actuator component is capable of automatically putting the new service configuration in place as required by the Decision Making component.

4. System and Service Model in BDMaaS

Large-scale Cloud-based IT services are complex systems, formed by a large number of interconnected software components, and their modeling is a very challenging task that requires to consider tradeoffs in model complexity. This section provides a detailed description of all main assumptions and motivations of modeling choices adopted in the modeling stage of our BDMaaS engine.

In the literature, some recent works model Cloud computing services as composed of fully independent components [8]. That assumption allows to significantly limit the complexity of the service placement framework through the use of analytic methods to identify the optimal service placement. However, that model does not suit well the complexity of Cloud ecosystems that typically require 2- or 3-tier architectures.

A different approach, stemming from research on traditional, i.e., non-Cloud-based, Web services proposes to model both the single software component behavior, leveraging on closed queuing networks [16] or service time approximation methods [17], as well as their interactions within service sessions. However, these models are not always applicable to the service placement optimization of Cloud-based infrastructures. In fact, in modern large-scale Cloud-based IT service deployments the database function is often implemented by a geo-replication system between the different data centers [18]. This means that for large scale deployments, the traditional 3-tier architecture of Web services translates to a 2-tier architecture plus an additional database layer that, for the service placement optimization purpose, could be considered as a commodity available at each Cloud data center.

In addition, complex Cloud-based services present dynamic aspects that cannot be ignored in the modeling process, especially with regards to the demand modeling phase. In fact, while demand modeling research in Web services focuses on the accurate reenactment of requests within service, i.e., browsing sessions [16, 17], demand modeling in Cloud-based services typically focuses on the reenactment of an aggregate flow of requests, that also considers daily and weekly patterns in request loads. More specifically, in order to correctly reenact the workload on the Cloud computing IT service, demand models should accurately capture the inter-arrival time patterns in service requests. In fact, service user requests typically have highly dynamic and non-trivial patterns, and their proper reenactment calls for the adoption of sophisticated modeling techniques based on non-parametric statistics.

To address all these challenging issues, we need new service and demand models able not only to capture the relationships between service components (each one deployed to a single VM), but also to measure the impact of a component reallocation to a different data center on the whole service performance as better described in the following subsections.

4.1. IT service model

BDMaaS service model enables to reenact complex Cloud-based IT services offered on a global scale; in the following, we first detail how we model complex distributed Cloud services, and then we focus on the modeling of single service components.

Starting with distributed modeling aspects, BDMaaS model evolves usual Web services models [16, 17]; it considers a Cloud service as a set of software components deployed on multiple Cloud data centers and using geo-replicated database layer available at each Cloud data center, thus resulting in a *2-tier architecture* deployment.

Moreover, our service model has been designed to capture the execution of the multiple service components (VMs) at different geographically distributed data centers, considering also the fine grained characteristics of VM types available for instantiation with corresponding set of virtual resources (CPU, storage, etc.) and pricing schemes. That closely represents offer of all more popular public IaaS Cloud environments, such as Amazon EC2.

Without loss of generality, each tier of the IT service is formed by a number of (equivalent and replicated) service components of the same type, so to model the adoption of horizontal scalability techniques usual of Cloud environments. Software components of each tier run in-

dependently, and interact only with software components from other tiers as service requests flow through them.

When a service request arrives at a Cloud data center, it is initially dispatched to a tier-1 software component (more specifically, to the corresponding VM) according to the tier-1 *load balancing policy* implemented by the IT service. Our model includes several types of load balancing policies, such as "random component" and "least loaded component" selection, designed to capture the behavior of load balancing support typically available by Cloud IaaS providers, such as Amazon EC2[3]. The same policies are used to model the request forward to tier-2 components.

Focusing on single service component modeling, we map service components to VMs, and assume that each VM runs a single software component that can be instantiated only in the subset of available VM types that satisfy its resource requirements.

As for the *execution model*, we opted for an open queuing network model because it represents a very flexible model capable of accurately capturing the behavior of a large class of real-life software components. In particular, we model tier-1 and tier-2 software components as multi-server $G/G/s_i$ First Come First Served (FCFS) queue, namely, single servers where both request inter-arrival times and service times have a (different) General distribution (i.e., G/G) and where s_i is the number of service processes concurrently running in the i-th service component. $G/G/s_i$ FCFS queues, supporting a wide range of stochastic models for service times from parametric probability distributions to empirical probability distributions built from log traces, effectively model both simple and very sophisticated service component behaviors within an easy-to-understand conceptual framework [19]. In addition, they allow to mimic different performance levels that software components exhibit when running in different VM types by tuning the set of model parameters $\theta_{i,j}$ for each couple of software component *i* and VM type *j*.

4.2. Modeling service user requests and SLA_{SU}

Modeling the request arrival process is also a particularly challenging task that requires considering multiple customers with different locations and service usage profiles that might change over time (daily and weekly) and with highly dynamic behaviors. For each service user, we consider a different, dynamic request generation profile designed to accurately capture the time-varying nature of his requests. Researchers have shown that dynamic workload on Cloud-based systems can be rather accurately predicted, for instance using AutoRegressive Integrated Moving Average (ARIMA) [20]. As a result, the varying-intensity request generation process for the *i*-th service user can be modeled through non-stationary stochastic processes, such as Non-Homogeneous Poisson Processes (NHPP), with a time-varying intensity $\lambda_i(t)$ [21]. In turn, the $\lambda_i(t)$ function could be inferred from the analysis of historical service request log data, through a predictive tool, e.g., ARIMA.

Alternatively, high-variance stationary stochastic processes, based on power law distributions such as Pareto, also represent a very interesting approach for service request reenactment. In fact, they are often proposed in research literature to model inter-arrival times of service requests in non-stationary conditions, as they enable to capture the time-varying behavior of service requests through a relatively simple stationary stochastic process, thus achieving a remarkably attractive tradeoff between model accuracy and complexity [21, 22].

We also assume that service user requests are automatically forwarded to the closest Cloud data center among those ones involved in the IT service deployment. This is a rather realistic assumption, as most Cloud providers nowadays offer that geographical request routing support, such as Amazon Route 53[4]. However, the accurate performance evaluation of Cloud computing services also requires to calculate the communication latencies involved in service user requests, that depend from the relative position of service user location and reference Cloud data center. Accordingly, we model communication latencies using stochastic processes.

From a business management perspective, instead, our model is very expressive and enables to define several SLA_{SU}s for an IT service, to associate them with a specific set of service users, and to set the corresponding violation penalties. BDMaaS provides service providers with a domain-specific language for the definition of IT- and business-level Key Performance Indicators (KPIs), of sophisticated SLA_{SU} verification checking functions on top of those indicators, and of the corresponding violation penalties calculation functions.

5. GA-based Service Placement Optimization

This section delves into the realization details of the optimization and decision making stages of our BDMaaS engine. First, it introduces the main phases to estimate the business impact (i.e., monetary cost), then it focuses on the proposed optimization technique, and finally it reports some details about the decision making approach.

5.1. Business Impact Analysis

We formalize the Business Impact (BI) analysis of a Cloud based IT service operating with configuration x through the definition of a function *BI*, whose evaluation is divided in two consecutive stages: an IT-level one and a business-level one. We then have:

$$\mathrm{BI}(x) = \mathrm{BIA}(\mathrm{ILM}(x)) \qquad (1)$$

The IT-Level Metrics (*ILM*) and the Business Impact Analysis (*BIA*) functions evaluate the performance of the IT service through IT-level and business level Key Performance Indicators (KPIs) respectively. In BDMaaS, those functions are respectively implemented by the Service Placement Simulation and Business Impact Analysis components. More specifically, ILM returns both the resource consumption of the IT service and its performance

[3] Amazon Web Services Elastic Load Balancing is available at: http://aws.amazon.com/elasticloadbalancing/

[4] Amazon Route 53 is available at: http://aws.amazon.com/route53/

measured through a set of IT-level metrics defined by the service provider, obtained from a simulation of the IT service operating in configuration x. These values are then provided to BIA that uses them to perform the business impact analysis of configuration x.

BIA consists of 3 subcomponents, involving different phases: new IT service configuration and deployment cost evaluation, SLA_{SU} violation penalties estimation, and (where applicable) service reorganization cost assessment. Formally, it is:

$$\mathrm{BIA(m)} = \mathrm{C}_{OPS}(\mathrm{m}) + \mathrm{C}_{CNTR}(\mathrm{m}) + \mathrm{C}_{DRIFT}(\mathrm{m}) \qquad (2)$$

where we apply the variable substitution m = ILM(x) for convenience.

In the first phase, corresponding to the C_{OPS} component, our framework calculates the operational costs caused by running system with configuration x according to the fees for all considered Cloud data centers. These costs are based on real data available at all major Cloud providers (such as Amazon, Microsoft, IBM, etc.) and change over time and for different geographic locations, depending on data center location and being influenced by several factors (energy cost, local security conditions, etc.).

The second phase, instead, considers the cost of operating the IT service in configuration x from the contracting perspective. The corresponding C_{CNTR} component evaluates whether switching to the new IT service configuration x would cause the service provider to incur in violation penalties, more formally:

$$C_{CNTR}(m) = \sum_{i=1}^{N} violations(i, m) \qquad (3)$$

where, for each σ_i (with $1 \leq i \leq N$) defined in the set SLA_{SU} of Service Level Agreements stipulated with the service users, the *violations* function calculates the corresponding business-level KPIs K_i, checks whether they fall within the corresponding target range τ_i stipulated with the service user, and, if not, applies the corresponding penalties.

Finally, the third phase, C_{DRIFT}, calculates the costs related to performance regressions, risk management, and reconfigurations for operating the IT service in configuration x with respect to the current configuration x_0. These effects are not fully captured by the C_{OPS} and C_{CNTR} components alone, so we define a function to specifically address them:

$$C_{DRIFT}(m) = f_{DRIFT}(m, ILM(x_0)) \qquad (4)$$

where f_{DRIFT} is usually defined as a penalization function over one or more IT-level KPIs.

C_{DRIFT} is an essential component in our model, that addresses multiple concerns. First, it effectively enables the introduction of risk management aspects that counterbalance the tendency of automated optimization solutions to select under-provisioned IT service configurations. In fact, assigning IT services the exact amount of virtual resources they are expected to require could leave them under-dimensioned and under-performing in case of unexpected load spikes, potentially leading to more SLA_{SU} violations. In addition, an optional and application-specific task, triggered only when necessary, is to calculate the costs related to IT service reorganization from the current configuration x_0 to the new configuration x. For instance, migrating VMs between different data centers might result in traffic costs as well as costs related to performance loss; moreover, it might be required to stop the service during the migration, and that would also contribute to increase the costs related to temporary service unavailability, and so forth.

5.2. Business Impact Optimization

After the first cost evaluation phase, BDMaaS chooses the solution that optimizes the BI, by minimizing it. Hene, from a theoretical perspective we formalize the service placement problem as the following optimization problem:

$$\min_{sbjct\ to\ x \in S_{PC}} BI(x) \qquad (5)$$

where the variable x represents the IT service configuration; the set S_{PC} represents the space of possible IT service configurations to explore; and *BI* is the function (described in the previous subsection) that evaluates the business impact of the IT service configuration x.

The *BI* function is very complex, thus limiting the applicability of traditional mathematical optimization algorithms, such as those based on gradient evaluation and descent techniques. In fact, since the gradient of the *BI* function is unknown, the calculation of approximate values for the gradients would require a large number of evaluations of *BI*, thus significantly slowing down the convergence time of the optimization process [23].

Because the space to explore is typically very wide, we in BDMaaS we decided to use meta-heuristics designed for large-scale optimizations. In fact, we believe that the complex nature of this class of optimization problems calls for the adoption of GA-based optimization techniques.

GAs not only have the very desirable property of being resilient to changes in the optimization function, as it may occur in our problem, but they are also able to promptly react to changes in the operating conditions by hasting the optimization process. In fact, (a part of) the fit individuals from the last population generation can be used as a starting/priming point for the next round of optimization [24]. In addition, the GA population size parameter can be modified to control the number of times that the optimization function is evaluated for every optimization cycle. Finally, GAs can be easily parallelized and it is possible to take advantage of parallel execution frameworks in the Cloud, such as MapReduce-based solutions [25].

As regards our customized GA, we map the typical concepts of GA-based techniques, namely, fitness to the environment, inheritance, selection, mutation, and recombination [24], to our optimization problem as follows:

- the space of candidate solutions (in our case, S_{PC}) has a genetic representation – that is, a candidate solution is represented as the genotype, i.e., the genet-

ic material, of a specific individual living in a natural environment;
- the function to optimize (in our case, BI) is a measure of the "fitness" of an individual (in our case x) to the natural environment;
- time is divided in epochs (or generations) in which only a (fixed-size) population of individuals can exist;
- the "fittest" individuals (for each generation) will be selected for breeding, through a process of mutation and recombination of their genetic material, thus generating an offspring that will form the next-generation population.

Within the family of GAs, a specific GA might differ from the others for the specific genetic representation of candidate solutions, or the selection, mutation, and recombination procedures that it adopts [26, 27]. We adopted a rather straightforward integer vector representation for the search space, namely, the service component allocation that in the GA metaphor takes the name of *genotype*. More specifically, our algorithm considers a genotype representation composed of as many *chromosomes* as the number of data centers to consider. In turn, each chromosome consists of two *genes*, each one containing the number of VMs allocated for the corresponding service tier, respectively, for Tier 1 and Tier 2, at that data center; hence, the content of each gene is an integer greater or equal to 1.

Moreover, our GA implements a binary tournament selection phase to select the candidates to reproduce, and a reproduction phase based on random geometrically-distributed mutations and intermediate recombination. While binary tournament and intermediate recombination are traditional mechanisms often adopted in GAs, the choice to adopt a geometric distribution-based mutation procedure is original and driven by the peculiar characteristic of this problem.

In fact, the geometric probability distribution is usually adopted to model the number of negative outcomes in a sequence of trials before achieving a positive outcome. For instance, the geometric distribution can be used to calculate the probability that a sequence of k+1 coin tosses will result in a sequence of k heads, when p_m is the probability of a single coin toss resulting in a tail.

In our experiments, we found that using values sampled from a geometric distribution as modifiers for the genes' integer representations makes the mutation procedure a particularly effective tool in exploring the search space, enabling it to strike a very good tradeoff between local and global exploration. This is consistent with what other researchers reported in GA-related scientific literature. For instance, in her seminal paper on hypermutation in GAs, Cobb notes that a mutation phase based on the sampling from a power law distribution represents a relatively simple procedure that leads to very effective results [28].

The mutation probability parameter p_m controls the shape of the distribution from which variations to the current individual's genes are sampled, thus effectively modulating the impact of the mutation phase in the GA-based optimization process. Higher values of p_m will result in lower average mutation sizes and lower values of p_m will lead to higher average mutation sizes. As we will demonstrate later, the choice of the mutation probability parameter is critical for the GA efficiency, as an excessively high value could significantly slow down the convergence process and an excessively low value could cause too much variations between a generation of individuals and the following one - thus effectively disregarding the memory of the algorithm, tilting the GA search process towards a mostly explorative behavior, and ultimately bringing the GA to operate in a sort of unstable state that severely harms the convergence process.

The adoption of the integer vector representation and of a mutation phase based on sampling from a geometric distribution brought the GA version presented in this paper to achieve a significantly improved efficiency compared to more traditional versions based on bitstring genotype representation, such as the one adopted in our previous work [11].

5.3. Decision Making Phase

The evaluation of different configurations and the detection of the optimal one needs to be followed by a decision making phase with the purpose of deciding whether the new configuration has to be applied or not. In fact, service reconfigurations are time- and resource-consuming, and should be performed as seldom as possible.

As a result, there is the need to put in place management precautions that limit the frequency of service placement reconfigurations. To avoid frequent reconfiguration and to avoid ping-pong effects, BDMaaS currently allows service reconfigurations only after a minimum time interval τ_{SLR} has elapsed since the last reconfiguration or when the difference between the (expected) business impact of the current and the new configuration exceeds a pre-configured threshold, thereby effectively implementing a hysteresis-based reconfiguration process.

6 Experimental Evaluation

We realized a prototype implementation of the BDMaaS engine in the JRuby platform. The prototype adopts the Simulator for IT Service in Federated Clouds (SISFC) component and the ruby-mhl optimization library, which we developed in the context of this research project[5] and released as open source.

We used our prototype to evaluate the behavior of our service placement framework in a limited but significant test scenario that attempts to capture the most critical aspects of service placement in large-scale Cloud computing environments with multiple data centers.

6.1 Service Model and Simulation Configuration

As introduced before, we focus on the problem of optimally placing in the Cloud the components of a complex IT service, considering 2-tier services modeled according

[5] For more information about the SISFC software, installation, configuration language, preferences, and used genotype representations, we refer the reader to project home page: http://endif.unife.it/dsg/sisfc

to what described in Section 4. We consider a deployment in multiple Cloud data center operated by Amazon to illustrate the opportunities in using BDMaaS even with a single Cloud provider scenario. In addition, we believe the single Cloud provider case to be the most common one for real-life IT service deployments at the moment of this writing, due to non-negligible difficulties in migrating service components from one Cloud provider to another. In particular, we consider three Amazon EC2 Cloud data centers: US (East), Japan, and Brazil.

In our experiments, we model service components as G/M/1 FCFS queues. This is a specialization of the general G/G/s_i FCFS model described in Section 4.1, in which inter-arrival times have a general distribution and service times for each job have an exponential distribution and a single service process. We adopt this model to highlight the generality of our approach that can lead to very interesting saving opportunities even when applied to a generic IT service with a roughly sketched service execution model. In fact, the G/M/1 model captures the behavior of a rather large part of software components with an acceptable trade-off between model complexity and accuracy.

As for model parameterization, we respectively selected a mean duration of 9 and 12ms for tier-1 and tier-2 service times, to emulate a larger processing at tier-2. We also assume that the times spent in tier-1 and tier-2 components of our service execution model include the access times to the geo-replicated database layer.

To generate request inter-arrival times, we adopted the model discussed in Section 4.2 based on the sampling from a random variable with a Pareto distribution:

$$F(x) = \begin{cases} 1 - \left(\frac{x_m}{x}\right)^{\alpha} & x \geq x_m \\ 0 & x < x_m \end{cases} \qquad (6)$$

with a scale parameter x_m equal to 1.2E-4s and a shape parameter α equal to 5.0, resulting in an incoming service request rate of 6666.67 per second (400,000 per minute).

Moreover, we assume that tier-1 components can fit into an Amazon EC2 x1.medium VM, while tier-2 component require an x1.large VM[6]. We consider the operational costs for those VM types detailed in Table I, which represent real market prices in the corresponding Amazon EC2 Cloud data centers.

Finally, we consider a service user with global presence, and assume the service user's requests to be uniformly distributed among the three considered Cloud data centers. Besides, to account for all possible communication latencies, such between service user and Cloud data centers and for geo-replicated database layer access, we assign to each request an additional aggregate amount of time estimated as a random time penalty sampled from a truncated Gaussian distribution with a mean of 100ms, a standard deviation of 25ms, and a minimum value of 20ms.

Each configuration is evaluated by the BDMaaS Service Placement Simulation component, implemented by the SISFC simulator, for a total (simulated) time of 40s. In particular, we did not consider the first 10s of (simulated) warm-up time not to bias results from observing the system in a cold start state, this leaves an effective (simulated) time of 30s, during which approximately 200,000 requests are evaluated. In our tests, this has proved to be an adequate interval of time to evaluate the system performance, while keeping the computational cost of the simulations manageable.

TABLE I. PRICING FOR MEDIUM AND LARGE SIZE VMS IN AMAZON EC2 DATA CENTERS (FIGURES IN $/HOUR).

	AMAZON US (EAST)	AMAZON JAPAN	AMAZON BRAZIL
MEDIUM	0.160	0.184	0.230
LARGE	0.320	0.368	0.460

Finally, we consider a fixed 1,500$ SLA_{SU} violation penalty in case the Mean Response Time (MRT) performance indicator raises above the 200ms threshold, and we impose a performance regression penalty in case the VM allocation fails to deliver the performance parameter set by IT managers. More precisely, we consider a *target* value SR_0=200,000 of requests served, and we adopted the following formula:

$$f_{DRIFT}(SR, SR_0) = \frac{10{,}000\ \$}{\frac{\pi}{2}} * \tan^{-1}\left(\frac{SR_0 - SR}{SR_0/8}\right) \qquad (7)$$

where the Served Requests (SR) metric represents the number of requests actually served, so to penalize IT service configurations that did not provide performance levels as expected by service providers. For SR values bigger than 200,000 no penalization is applied.

Let us note that the choice of an appropriate value for the scale parameter (10,000 $ * 2 / π in equation 7) of the f_{DRIFT} function represents a very important aspect, which the service provider should dedicate special attention to when inserting the description of an IT service to be optimized in BDMaaS. In fact, the scale parameter of f_{DRIFT} modulates the impact of the C_{DRIFT} component of equation (2) with respect to the the C_{OPS} and C_{CNTR} ones.

6.2 GA-based Optimization Assessment

The first set of experiments evaluates the effectiveness of the proposed GA in optimizing the service component placements. Initially, we selected 128 as the population size parameter of the GA through empirical trial-and-error process, which is usually recommended as a best practice when working with GAs [29].

We then tuned the mutation probability parameter of the GA running various experiments for 5 different values of mutation probability parameter: 0.1, 0.2, 0.3, 0.4, and 0.5 – corresponding to an average mutation of 9, 4, 2.33, 1.5, and 1 units respectively. We also assessed an adaptive control method of the mutation probability parameter p_m of the GA. More specifically, we implemented a modified version of the Rechenberg control algorithm, that modulates the mutation probability parameter value p_m according to the performance of the last G generations in the GA, increasing p_m of a factor α if more than 20% of the last G generations were more successful than their

[6] Information on VM instance types offered by Amazon EC2 is available at: http://aws.amazon.com/ec2/instance-types/

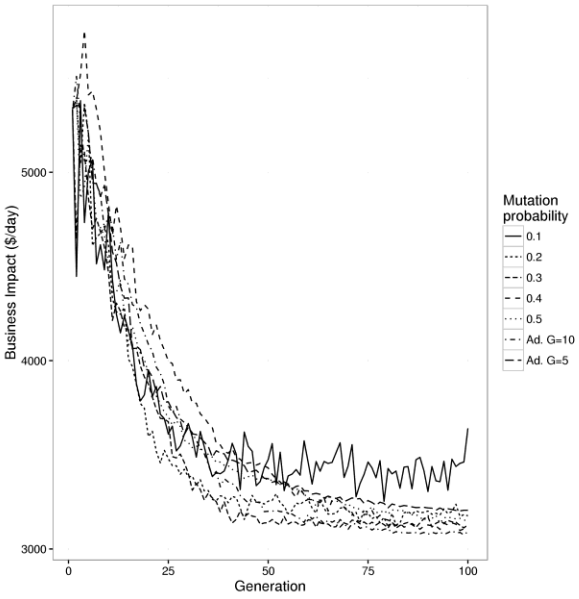

Fig. 3. Convergence speed of the GA using different mutation operators.

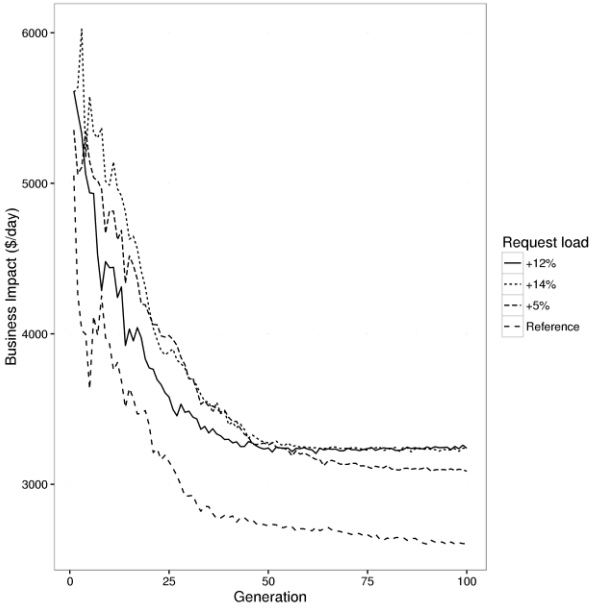

Fig. 4. Convergence speed of the GA with different request loads.

preceding generation in finding a fitter solution, and decreasing p_m of α instead [29, 30]. This adaptive control of p_m enables to dynamically tune the GA, steering it towards a more explorative or a more exploitative behavior, according to the (recent) results of the search space exploration. We tested our adaptive control algorithm using an initial value of p_m equal to 0.3, $\alpha = 0.1$, and in two different configurations for the number of generations to consider in the control procedure: G=5 and G=10 respectively. In all the experiments presented in this paper, we adopted the recommended value of 0.25 for the intermediate recombination probability parameter [24].

Fig. 3 shows on the y-axis the value of the BI function, in $/day, when calculated on the best individual of the generation indicated on the x-axis. The $p_m = 0.1$ curve

TABLE II. OPTIMAL VM PLACEMENT FOR EXPERIMENT 1.

	AMAZON US	AMAZON JAPAN	AMAZON BRAZIL
MEDIUM	25	25	24
LARGE	33	31	31

TABLE III. OPTIMAL VM PLACEMENT FOR EXPERIMENT 2.

		AMAZON US	AMAZON JAPAN	AMAZON BRAZIL
+5%	MEDIUM	28	27	25
	LARGE	31	34	32
+12%	MEDIUM	28	27	29
	LARGE	34	34	33
+14%	MEDIUM	33	27	27
	LARGE	36	34	37

corresponds to the GA variant with the highest impact of the mutation phase and, consequently, an aggressively explorative (instead of exploitative) behavior. In this case, the GA exhibits a steep drop in business impact at first, but it soon enters an "oscillating" operating condition which the algorithm keeps moving between a series of local minima. The best convergence curve is the one obtained from the adaptive control of the p_m parameter with G=10. In fact, the adaptive control GA variant achieves an optimal tradeoff between explorative and exploitative behavior, thus helping the GA to work around uninteresting parts of the search space (plateaus and local minima) so to focus on more interesting ones.

The best configuration that we obtained in the experiment resulted in a cost of 3074.64 $/day (1208.35 $/day for running the VMs and 1866.29 $/day for performance related penalties). Table II shows the number of VMs allocated to each data center in this configuration.

6.3 Dynamic Adaptiveness to Request Load Variations

The second set of experimental results verifies the capacity of the BDMaaS framework to adapt to a dynamic request load. To this end, we simulated three request load variations, with an increase of 5%, 12%, and 14% with respect to the reference request load used in the previous experiment. We changed the penalization model accordingly, by setting the target value SR_0 in equation (7) to 210,000, 224,000, and 228,000, so to match the increased number of expected service requests.

We used the same parameters adopted in the previous experiment (population size 128, adaptive control of mutation probability parameter starting from an initial value of 0.3 and re-evaluating the mutation probability every 10 generations), and we consider as baseline the curve obtained for the reference workload (see Fig. 4).

The curves for the +12% and +14% workload conditions exhibit a better convergence rate than the one for the +5% one. This behavior can be traced back to the different target functions considered in the optimization runs (SR_0 assumes a different value in each run) and to the stochastic nature of the GA. In any case, let us point out that these are all valid outcomes for the GA, and that continu-

TABLE IV. SERVICE USER CHARACTERIZATION.

SERVICE USER	OPERATING IN	DATA CENTER	REQUEST SHARE	SERVICE CLASS
SU1	ASIA	AMAZON JAPAN	10%	GOLD
SU2	SOUTH AMERICA	AMAZON BRAZIL	15%	GOLD
SU3	ASIA	AMAZON JAPAN	30%	SILVER
SU4	NORTH AMERICA	AMAZON US	22.5%	SILVER
SU5	NORTH AMERICA	AMAZON US	22.5%	BRONZE

TABLE V. OPTIMAL VM PLACEMENT FOR THE DIFFERENTIATED SERVICES EXPERIMENT.

	SU1	SU2	SU3	SU4	SU5
MEDIUM	39	7	34	42	30
LARGE	7	11	43	30	21

ous optimization tool like BDMaaS, that are designed for systems with strongly dynamic behaviors, are better served by GA variants that exhibit a smooth, steep, and reliable descent, instead of GA variants that reach the lowest plateau in a high number of generations.

The new configurations selected by our prototype resulted in a cost of 3086.66 $/day (1256.40 $/day for running the VMs and 1830.26 $/day for performance related penalties) for the +5% run, of 3204.72 $/day (1312.56 $/day for running the VMs and 1892.16 $/day for performance related penalties) for the +12% run, and of 3215.22 $/day (1380.24 $/day for running the VMs and 1834.98 $/day for performance related penalties) for the +14% run. This represents a cost increase of 18.71%, 23.25%, and 23.65% respectively with regards to the reference configuration. Table III shows the number of VMs allocated to each data center for the three different levels of request loads evaluated. These results confirm that our prototype is very effective in finding good VM placement configurations under different operating conditions.

6.4 Differentiated Service Placement

In the next experiment, we considered a more complex scenario to demonstrate the effectiveness of BDMaaS in dealing with multiple service users with different agreed SLA_{SU}s. In particular, we considered 5 service users grouped in 3 different service classes, namely Gold, Silver, and Bronze, with the characteristics in Table IV.

We defined for each service class specific penalties so to obtain differentiated guarantees depending of the expected business impact. Gold customers have a 4,000$ SLA_{SU} violation penalty in case MRT in serving requests rises above the 200ms threshold. Silver service users have a 2,000$ SLA_{SU} violation penalty in case MRT overcomes the 250ms threshold. Bronze customers have a 1,000$ SLA_{SU} violation penalty in case MRT trespasses the 300ms threshold. In addition, we assume that SU1 and SU3 address their request to the Amazon Japan data center, SU2 to the Amazon Brazil data center, and SU4 and SU5 to the Amazon US data center. The fourth column of Table IV reports the share of the total number of requests for each service user. We also updated the penalization model of equation (7) by setting the target value SR_0 to the sum of requests expected from the 5 service users.

The multiple service user problem is significantly more complicated than the single service user one because it requires to consider the allocation of service user-dedicated VMs. Instead of (significantly) complicating the service model to implement support for service user-dedicated VMs, we decided to address this problem by slightly changing the genotype representation. In the new representation, each gene represents the number of components of the corresponding tier (i.e., Tier 1 and Tier 2) allocated for the corresponding service user (i.e., chromosome).

We first empirically verified that a population size of 128 individuals enabled to effectively explore the new and significantly larger search space. So we used that parameter for all the following experiments. With regard to the mutation probability parameter, in this experiment we kept using the adaptive control process that proved to enable a very effective search in the previous experiments (set p_m to 0.3 at start and reevaluate it every 10 generations).

Table V shows the number of VMs allocated to each service user for the best solution we observed in this experiment, that resulted in a cost of 9588.37$/day (1588.37 $/day for running the VMs and 8000.00$/day for SLA_{SU} violation penalty). These results demonstrate the capacity of BDMaaS to effectively optimize complex Cloud-based IT services with multiple service user.

6.5. Adaptiveness to Load and Price Variations

The last set of experimental results has the purpose to assess the capacity of our placement framework to adapt to dynamic variations in the operating conditions. In particular, we simulated three consecutive events: a 20% increase in the request load of US-based service users (SU4 and SU5), consistent with the start of the office shift in USA around UTC+12; a 5% drop in the VM prices of the Amazon Japan data center, consistent with the offering of spot instance VMs at 95% of the price of on demand ones, from Amazon Japan to increase the overnight data center utilization around UTC+13 and on the corresponding detection and exploitation of this opportunity by BDMaaS; and finally a 25% drop in the request load of Japan-based customers (SU1 and SU3), consistent with the end of office shift in Japan around UTC+14.

Let us stress that introducing dynamic variations in the system operating conditions changes the fundamental nature of the optimization problem, turning it into a dynamic optimization problem, that calls for more sophisticated optimization solutions [30]. In particular, researchers have dedicated a significant amount of effort to implement GAs that could effectively deal with dynamic optimization problems, developing a wide range of strategies that keep using the population evolved as a reference starting point and increase its diversity to allow a wider exploration of the search space [31].

Upon the reasonable assumption of knowing when a

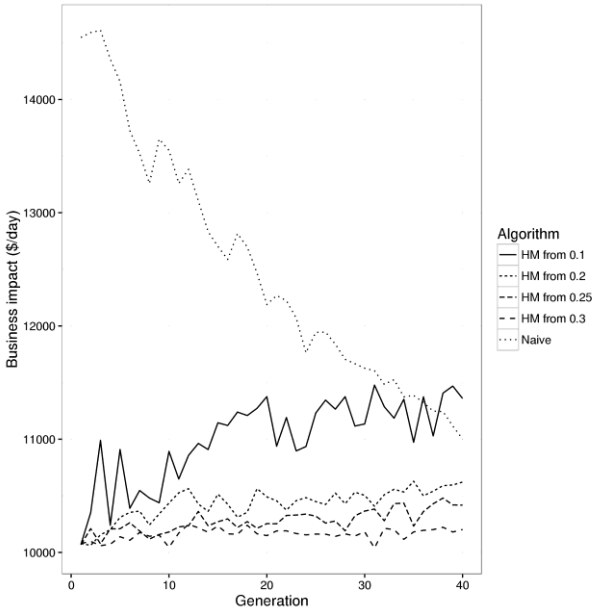

Fig. 5. Convergence speed of the GA in the differentiated service placement experiment after the first variation, using different mutation operators.

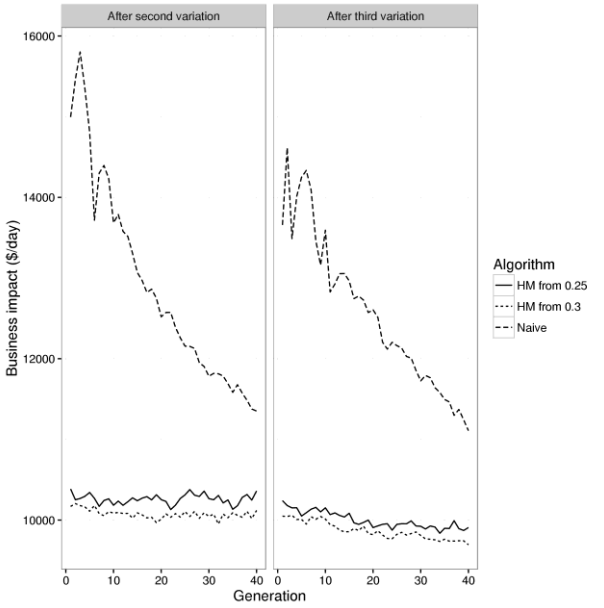

Fig. 6. Convergence speed of the GA in the differentiated service placement experiment after the second and third variation.

change in the system operating condition occurs – something that could be easily achieved through a relatively simple analysis of request load and the monitoring of changes in VM pricing schemes adopted by the data centers – we can introduce dynamic optimization support in our GA by implementing a "reactive" strategy that increases the diversity in the population upon a change detection [32]. In our problem, reactive strategies are indeed preferable to proactive ones because the latter cannot assume to know when a change in the operating conditions occur and need to preserve diversity in the population. Thus, often it requires a significantly higher number of evaluations to converge to a good solution (a behavior that, as we mentioned above, is undesirable when operating with simulation-based target functions).

TABLE VI. OPTIMAL VM PLACEMENT FOR THE DIFFERENTIATED SERVICES EXPERIMENT AFTER 1ST CONDITION VARIATION.

	SU1	SU2	SU3	SU4	SU5
MEDIUM	10	14	38	35	27
LARGE	13	20	40	36	34

TABLE VII. OPTIMAL VM PLACEMENT FOR THE DIFFERENTIATED SERVICES EXPERIMENT AFTER 2ND CONDITION VARIATION.

	SU1	SU2	SU3	SU4	SU5
MEDIUM	13	14	34	32	25
LARGE	12	16	41	41	31

TABLE VIII. OPTIMAL VM PLACEMENT FOR THE DIFFERENTIATED SERVICES EXPERIMENT AFTER 3RD CONDITION VARIATION.

	SU1	SU2	SU3	SU4	SU5
MEDIUM	10	13	23	30	28
LARGE	11	13	32	38	27

More specifically, the effective results we achieved with a mutation phase based on random geometrically distributed variations and on the adaptive control of the p_m parameter that determines the mutation intensity, led us to implement a hypermutation-based approach to enable support for dynamic optimization in our GA. Hypermutation is a dynamic optimization strategy developed in the context of immune artificial systems research and inspired to the behavior of biological cells, that tend to aggressively increase their mutations in response to stressful changes in their environment [28]. The hypermutation strategy we implemented in our GA responds to changes by immediately setting the value of the p_m parameter to a (low) value HM, thus increasing the intensity of mutations and causing the GA population to enter a "hypermutation" state, and then normally resuming the adaptive control of p_m. As we will see shortly, this is a relatively simple but very effective strategy.

We first evaluated our hypermutation strategy implementation by analyzing its response to the 20% increase in US-based service user requests. Reusing the 128-sized GA population produced by the previous experiment as a starting point, we experimented with different values for the hypermutation parameter: 0.1, 0.2, 0.25, and 0.3. Fig. 5 shows the results that we obtained, comparing them with those returned from a naive optimization approach, i.e., running the GA with a random start population. Let us underline that the "HM from 0.1" curve exhibits the same oscillating pattern observed in Fig. 3 for $p_m = 0.1$, corresponding to a strongly explorative behavior of the GA.

Table VI shows the number of VMs allocated to each service user for the best solution we observed in this experiment, that resulted in a cost of 10037.48$/day (1753.82$/day for running the VMs and 8283.66$/day for performance related penalties).

The results clearly demonstrate that the hypermutation based extension to our GA worked much better than the naive approach. As in the first experiment, low values (0.2 and below) for the p_m parameter caused an excessive amount of changes in the population and led to poor results. The 0.25 value for the hypermutation threshold HM

produced the best results. Table VI shows the number of VMs allocated to each service user for the best solution we observed after the first variation.

We then ran also experiments for the other changes in the system condition, i.e., the availability of spot instance VMs in Amazon Japan at 95% of the price for on demande ones and the 25% decrease in requests by Japan-based customers, using the 0.25 and 0.3 values for hypermutation threshold HM. At each step, we primed the GA by reusing the population produced by the previous step with the corresponding HM value. The results are shown in Fig. 6, compared with a run of the GA using a naive approach (restarting it from a random population). As one can see, the 0.25 value for the hypermutation threshold HM led to the best overall results. Tables VII and VIII show the number of VMs allocated to each service user for the best solution we observed after the second and third variation, that respectively resulted in a cost of 9950.12$/day (1667.63$/day for running the VMs and 8282.50$/day for performance related penalties) and of 9694.02$/day (1346.43$/day for running the VMs and 8257.59$/day for performance related penalties).

These results clearly demonstrate that our prototype is very effective at dealing with changes in the operating conditions of the system and that it was consistently capable of finding fitter solutions.

7 Related Work

Existing efforts span several different areas related to business-driven IT management. In the following, without any pretense of being exhaustive, we will first report a very brief overview of works focusing on the usage of GA per-se for the management of complex dynamic systems. Then, we will analyze existing solutions in cloud management and we will address research efforts specifically aimed to apply GA techniques to Cloud management. A comparison and discussion of those management solutions ends the section.

The literature on evolutionary optimization solutions for dynamic problems is very large. For an in depth overview, we refer the reader to the recent and excellent survey [32]. However, the same work reports on growing criticism from several well-known researchers, such as Ursem *et al.* [33] and Branke *et al.* [34], that most of the proposed solutions are tested only with theoretical benchmarks devised by academics, that do not necessarily represent the behavior of a large class of real-life problems. Other authors, such as Kramer [30], report that only very few studies so far performed a rigorous comparison of different parameter control methodologies on the same problem, e.g., to evaluate whether the adoption of adaptive optimization strategies could actually lead to a reduction of convergence speed in dynamic systems.

Moving to Cloud management, the first seminal efforts in this area tackled service/resource placement with different goals, such as minimizing energy consumption [1, 2, 3], improving IT performances [5, 35], and reorganizing service schemes to deliver agreed SLAs [6, 7, 36] and to let service providers express specific placement constraints [6]. Most efforts addressed these goals at the infrastructure level (see also the survey [4]) by considering mainly internal IT objectives such as SLA_{SP}s and technical requirements (e.g., local CPU, memory at each physical host, and communication). We limit our analysis to more recent efforts that recognize communication and service deployment structure restrictions as very relevant constraints that raise the complexity of the VM placement problem [5, 6, 35, 36]. In [35] authors formulate structural constraint-aware VM placement with three types of constraints (i.e., demand, communication, and availability) as an *NP*-hard problem; hence, they propose four approximation algorithms with related heuristics to solve it in feasible time. [36] focuses on the *NP*-hard problem of network-aware VM placement with the goal of reducing the aggregate traffic into the data center (e.g., by co-locating VMs that highly communicate). With a different perspective, authors in [5] define an *NP*-hard VM placement problem (and heuristics to solve it) and solve it in a way that minimizes VM relocations and is resilient to dynamic traffic time-variations. Finally, in [6] authors analyze how the number of structural VM placement constraints and the data center background load constraints affect the solving time of VM placement formulated as an Integer Linear Programming (ILP) problem. Although adopting an internal perspective and not considering additional business constraints, most of these proposals tend to either (similarly to our approach) adopt approximate solutions for dominating problem complexity when using accurate models [5, 35, 36], or to find exact solutions to a simplified approximate model [6].

Focusing on very recent Cloud management efforts, business-driven service placement solutions for large-scale Cloud environments are the closest works with respect to our proposal. A seminal work in this area is [9], that addresses the management of changes to IT infra-

TABLE IX. Summary and comparison of the surveyed solutions

Reference to Solution	Internal/external Perspective (*int/ext*)	Static/Dynamic Optimization (*sta/dyn*)	Exact/Approximate Solution (*exac/appr*)	Business-driven Approach	Simulative Approach
[35]	*int*	*sta*	*appr*	×	×
[36]	*int*	*sta*	*appr*	×	×
[5]	*int*	*sta*	*appr*	×	×
[6]	*int*	*dyn*	*exac*	×	×
[9]	*ext*	*dyn*	*exac*	✓	×
[8]	*int*	*dyn*	*exac*	×	×
[37]	*ext*	*sta*	-	✓	×
[38]	*ext*	*sta*	-	✓	×
[39]	*int*	*dyn*	*appr*	×	×
[40]	*ext*	*dyn*	*appr*	×	×
[41]	*ext*	*dyn*	*appr*	✓	✓
Our approach	*ext*	*dyn*	*appr*	✓	✓

structure and services to satisfy business goals and to minimize costly disruptions on the business, by focusing on an interesting real case study about a 2-tier service. Several other works have recently started offering virtualized resources and services in the form of Virtual Data Centers (VDCs) consisting of VMs connected through virtual switches, virtual routers, and virtual links with guaranteed bandwidth. VDC Planner is a recent proposal to improve the success rate of VDC mapping by minimizing total VM migration costs [8] but, differently from our proposal meant mainly for service providers, this proposal is aimed at increasing Cloud provider revenues. With a more external service provider perspective, [37] presents a comparative analysis of the economic models for Cloud computing and traditional in-house IT service delivery for emergent and established countries. Similarly, [38] studies cost models and possible cost minimization strategies for service placement in federated hybrid Cloud environments. These last two proposals are much related to our work, but they still lack risk-related aspects; in addition, they do not consider large-scale Cloud deployments.

With regards to solutions that propose to use GA for Cloud management, researchers operating in network and system management are increasingly turning towards computational intelligence methods for the optimization of the systems they study. This is in line with a trend that has emerged in research on complex and dynamic systems, where it is becoming more and more convenient to analyze systems through simulation instead of with analytic methods that are often very complex and thus call for simplifying assumptions [23]. [39] presents an interesting comparison between a traditional ILP optimization method with genetic algorithm and particle swarm optimization. The authors conclude that the guarantee of finding the best solution of the problem provided by ILP is well offseted by its scaling issues as the problem dimension grows larger. [40] presents a comparison of several different optimization algorithms for the placement of IT service components in federated Clouds with dynamic prices, using a relatively simple scheme that changes the cost of VM allocations according to the currently available computational capacity at Cloud data centers. However, those works focus on the (static) optimization of systems operating in steady-state and, unlike the work presented in this paper, do not consider dynamic optimization aspects.

In another previous work [41], we attempted to optimize the placement of software components in federated Cloud environments using a hybrid GA plus random search algorithm. Compared to the GA used in this work, the hybrid GA used in [41] allows for a faster exploration of the search space at the price of a simplified VM allocation pattern assumption that might not suit the need of all possible IT services.

Table IX compares above solutions with respect to the main design dimensions introduced and considered in this paper. While several proposal adopt approximate solutions to dominate problem complexity, it is quite evident that the external perspective and the adoption of business-driven IT management approaches are relatively novel, and so there are less systems realizing them. Moreover, although most of considered solutions support dynamic optimization our approach is (apart from our previous work [41], which however has narrower applicability) the only one adopting a simulative approach for business-driven IT management of Cloud, with inherent benefits mitigating possible risks from a business perspective.

8 Conclusions and Future Work

The paper proposed BDMaaS, a business-driven IT service placement solution based on genetic algorithm optimization. We realized a prototype and assessed it by collecting several experimental results about a simulated 2-tier service deployed in a Cloud computing environment with multiple data centers. Collected results show that BDMaaS is able to evaluate the most promising component placement configuration and to dynamically and respond to a wide range of situations from resource pricing changes, to support for differentiated requests and quality levels. BDMaaS engine is available and we believe practitioners will benefit from the possibility to use it for business-driven placement of their services.

Encouraged by these results, we are considering several future research directions: integrating BDMaaS within our IaaS management infrastructure based on the OpenStack Cloud, to automate the deployment of dynamically evaluated service placement reconfigurations; evaluating possible alternative meta-heuristics to GAs; considering additional business elements in the risk management analysis, such as reliability-related aspects in the service component placement choice.

Another relevant future research goal is the optimum selection of a given alternative for a specific component among a pool of candidates. In fact, while BDMaaS is already capable of carrying out this task given a proper description of IT service, modeling in a realistic way the possibility of switching between different SaaS providers would require a comprehensive evaluation of many different aspects in decision making, both tangible (cost, performance, compatibility, etc.) and intangible (risk, security, etc.) nature, that we are exploring as part of our ongoing efforts to further evolve BDMaaS.

References

[1] G. Jung *et al.*, "Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures", in *Proc. of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS'10)*, pp.62-73, 2010.

[2] V. Mann *et al.*, "VMFlow: leveraging VM mobility to reduce network power costs in data centers", in *Proc. of the 10th international IFIP conference on Networking (NETWORKING'11)*, pp. 198-211, 2011.

[3] B. Kantarci, H. T. Mouftah, "Energy-Efficient Demand Provisioning in the Cloud", in *Proc. of Optical Fiber Communication Conference (OFC)*, pp. 1-3, 2012.

[4] M. Mishra *et al.*, "Dynamic resource management using virtual machine migrations", *IEEE Communications Magazine*, vol. 50, no. 9, pp. 34-40, 2012.

[5] O. Biran *et al.*, "A Stable Network-Aware VM Placement for

Cloud Systems", in *Proc. of IEEE/ACM Int.'l Conf. on Cloud, Cluster and Grid Computing (CCGrid)*, pp. 498-506, May 2012.
[6] D. Espling *et al.*, "Modeling and Placement of Cloud Services with Internal Structure", *IEEE Transactions on Cloud Computing*, vol.PP, no.99, pp. 1-11, DOI: 10.1109/TCC.2014.2362120.
[7] J. Kirschnick *et al.*, "Toward ad Architecture for the Automated Provisioning of Cloud Service", *IEEE Communications Magazine*, Vol. 48, No. 12, pp. 121-131, Dec 2010.
[8] Q. Zhang *et al.*, "Dynamic Service Placement in Geographically Distributed Clouds", *IEEE Journal on Selected Areas in Communications (JSAC)*, Vol. 31, No. 12, pp. 762-772, December 2013.
[9] S. Hagen, A. Kemper, "Facing the unpredictable: Automated adaption of IT change plans for unpredictable management domains", in *Proc. of IEEE International Conference on Network and Service Management (CNSM)*, pp. 33-40, 2010.
[10] A. Moura, J. Sauve, C. Bartolini, "Business-driven IT management - upping the ante of IT", *IEEE Communications Magazine*, vol. 46, no. 10, pp.148-153, October 2008.
[11] L. Foschini, M. Tortonesi, "Adaptive and Business-driven Service Placement in Federated Cloud Computing Environments", in *Proc. of the 8th IFIP/IEEE Int. Workshop on Business-driven IT Management (BDIM 2013)*, 2013.
[12] M. Macías, J. Guitart, "SLA Negotiation and Enforcement Policies for Revenue Maximization and Client Classification in Cloud Providers", *Future Generation Computer Systems*, vol. 41, pp. 19-31, Dec. 2014.
[13] D. Hubbard, "How to Measure Anything: Finding the Values of 'Intangibles' in Business", 2nd Edition, Wiley, 2010.
[14] J. Sauvé *et al.*, "Prioritizing Information Technology Service Investments under Uncertainty", *IEEE Transactions on Network and Service Management*, vol. 8, no. 3, pp. 1-15, Sep. 2011.
[15] T. Saaty, L. Vargas, "Models, Methods, Concepts & Applications of the Analytic Hierarchy Process", Springer, 2012.
[16] B. Urgaonkar *et al.*, "Analytic modeling of multitier Internet applications", *ACM Transactions on the Web*, vol. 1, no. 1, May 2007, Article 2.
[17] N. Grozev, R. Buyya, "Performance Modelling and Simulation of Three-Tier Applications in Cloud and Multi-Cloud Environments", *The Computer Journal*, 2013.
[18] W. Lloyd *et al.*, "Don't Settle for Eventual Consistency", *ACM Queue*, vol. 12, no. 3, Mar. 2014.
[19] D. Gross *et al.*, "Fundamentals of Queueing Theory", 4th Edition, Wiley, 2008.
[20] R. Calheiros *et al.*, "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS", *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449-458, Oct.-Dec. 2015.
[21] C. Bartolini, C. Stefanelli, M. Tortonesi, "Synthetic Incident Generation in the Reenactment of IT Support Organization Behavior", in *Proc. of the 13th IFIP/IEEE Integrated Network Management Symp. (IM'13)*, pp. 126-133, 2013.
[22] H. Qian *et al.*, "Service Management Architecture and System Capacity Design for PhoneFactor--A Two-Factor Authentication Service", in *Proc. of 11th IFIP/IEEE Int. Symp. on Integrated Network Management (IM'09)*, pp. 73-80, 2009.
[23] A. Gosavi, "Simulation-based Optimization", 2nd Edition, Springer, 2015.
[24] S. Luke, "Essentials of Metaheuristics", 2nd Edition, Lulu, 2013.
[25] S. Kailasam, P. Dhawalia, S.J. Balaji, G. Iyer, J. Dharanipragada, "Extending MapReduce across Clouds with BStream", *IEEE Transactions on Cloud Computing*, vol. 2, no. 3, pp. 362-376, Jul.-Sep. 2014.
[26] A. Eiben, J. Smith, "Introduction to Evolutionary Computing", Springer, 2003.
[27] D. Goldberg, "Genetic Algorithms", Addison-Wesley, 1989.
[28] H. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments", Technical Report AIC-90-001, NRL, Washington DC, USA, 1990.
[29] X. Yu, M. Gen, "Introduction to Evolutionary Algorithms", Springer, 2010.
[30] O. Kramer, "A Brief Introduction to Continuous Evolutionary Optimization", Springer, 2014.
[31] E. Alba, A. Nakib, P. Siarry, "Metaheuristics for Dynamic Optimization", Springer, 2013.
[32] T. Nguyen, S. Yang, J. Branke, "Evolutionary dynamic optimization: a survey of the state of the art", *Swarm and Evolutionary Computation*, vol. 6, pp. 1-24, Oct. 2012.
[33] R. Ursem *et al.*, "Analysis and modeling of control tasks in dynamic systems", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 378-389, Aug. 2002.
[34] J Branke, E. Salihoğlu, Ş. Uyar, "Towards an analysis of dynamic environments", In *Proc. of the 7th annual conference on Genetic and evolutionary computation (GECCO '05)*, pp. 1433-1440.
[35] D. Jayasinghe *et al.*, "Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement", in *Proc. of the Int. Conf. on. Services Computing (SCC'11)*, pp. 72-79, 2011.
[36] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. of the 29th IEEE Conf. on Information Communications (INFOCOM'10)*, pp. 1154-1162, 2010.
[37] K. Sripanidkulchai and S. Sujichantararat, "A business-driven framework for evaluating cloud computing", in *Proc. of 7th IEEE/IFIP Int. Workshop on Business-Driven IT Management (BDIM 2012)*, pp. 1335-1342, 2012.
[38] Jörn Altmann, Mohammad Mahdi Kashef, "Cost model based service placement in federated hybrid clouds", *Future Generation Computer Systems*, vol. 41, pp. 79-90, Dec. 2014.
[39] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck "Hierarchical Network-Aware Placement of Service Oriented Applications in Clouds", *Proc. of the 14th IEEE/IFIP Network Operations and Management Symp. (NOMS 2014)*, 2014.
[40] W. Li *et al.*, "Cost-Optimal Cloud Service Placement under Dynamic Pricing Schemes", in *Proc. of the 6th IEEE/ACM Int. Conf. on Utility and Cloud Computing (UCC'13)*, 2013.
[41] G. Grabarnik, L. Shwartz, M. Tortonesi, "Business-Driven Optimization of Component Placement for Complex Services in Federated Clouds", *Proceedings of the 14th IEEE/IFIP Network Operations and Management Symp. (NOMS'14)*, 2014.

Mauro Tortonesi [M] graduated from the University of Ferrara, Italy, where he received a Ph.D. degree in computer engineering in 2006. He is an assistant professor at the Engineering Department of the University of Ferrara. His research interests include distributed and mobile computing, opportunistic networking, IT service management, business-driven IT management, and large-scale e-Maintenance.

Luca Foschini [M] graduated from the University of Bologna, Italy, where he received a Ph.D. degree in computer engineering in 2007. He is now an assistant professor of computer engineering at the University of Bologna. His interests include distributed systems and solutions for system and service management, management of cloud computing, context-aware session control and adaptive mobile services, and mobile crowdsensing.