Optimal autonomic management of service-based business processes in the cloud

Leila Hadded · Tarek Hamrouni

Abstract

Cloud computing is an emerging paradigm that provides hardware, platform and software resources as services based on pay-as-you-go model. It is being increasingly used for hosting and executing service-based business processes. However, business processes are subject to dynamic evolution during their life-cycle due to the highly dynamic evolution of the cloud environment. Therefore, to efficiently manage them according to the autonomic computing paradigm, service-based business processes can be associated with autonomic managers. Autonomic managers monitor these processes, analyze monitoring data, plan configuration actions, and execute these actions on these processes. The main objective of cloud computing is to improve the performance level while minimizing operating costs. Thus, due to the diversity of business processes requirements and the heterogeneity of cloud resources, discovering the optimal management cost of a business process in the cloud becomes a highly challenging problem. For this purpose, we propose an approach based on an Integer Linear Program to find out the optimal allocation of cloud resources that meet customers’ requirements and resources’ constraints. Besides, to validate our approach under realistic conditions and inputs, we propose an extension of the CloudSim simulator to analyze the cloud resources consumed by an autonomic business process. Experiments conducted on two real datasets highlight the effectiveness and performance of our approach.

Keywords Cloud computing · Autonomic management · Business process · Deployment · Linear program

1 Introduction

Over the last decade, cloud computing has appeared as a new model enabling on demand network access to shared configurable computing resources (e.g., networks, servers, storage, applications, and services) which can be dynamically provisioned and released with minimal service provider interaction (Mell and Grance 2011). These resources are provided “as a service” over the Internet. The three main models of cloud services are as follows: Infrastructure as a Service (IaaS), which provides computational resources in the form of Virtual Machines (VMs), Platform as a Service (PaaS), and Software as a Service (SaaS), which provides software applications and data. These applications are also known as service-based applications. Assembling services into an application can be ensured by using any appropriate service composition specification that can be either architecture-based (e.g., modeled in UML component diagram Jacobson and Booch 2011 or Service Component Architecture (SCA) Marino and Rowley 2009) or behavior-based (e.g., modeled in Business Process Execution Language (BPEL) (Juric 2006). In the latter case, applications are known as Service-based Business Processes (SBPs).

Autonomic computing has received great attention in the recent years, particularly in cloud computing, to manage cloud resources automatically and dynamically to respond to the requirements of the business defined in Service Level Agreements (SLA) (Jamshidi et al. 2014; Mohamed et al. 2015). The central component in an autonomic computing system is MAPE-K (Monitor, Analyzer, Planner, Executor, and Knowledge) loop, also known as Autonomic Manager (AM) (ibm 2005). An AM consists on periodically collecting monitoring data, analyzing them and generating reconfiguration actions to be executed on the managed system.
SBPs must be properly executed to avoid any time violation that could lead to serious consequences. In fact, on-time delivery of services has a direct impact on customer satisfaction. Thus, SBPs must be associated with AMs to autonomic management these business processes in order to meet the customer specified Quality of Service (QoS) constraints. However, autonomic management of SBPs is a new trend that presents several challenges among which how to find an optimal SBP’s autonomic management cost, which is a very difficult problem. This is due to the heterogeneity of cloud resources that are offered at different prices and the diversity of SBP services requirements. For this reason, our work presented in Hadded et al. (2016) addressed the problem of resource allocation for autonomic SBPs. To this end, we proposed an approach that finds (i) the suitable AM to allocate for each service, and, (ii) the suitable VM to allocate for each AM. However, it considers neither the diversity of customers’ requirements nor the heterogeneity of resources offered by cloud providers, whereas in Hadded et al. (2018), we took into consideration these two aspects to solve the problem. Nevertheless, these works (Hadded et al. 2018; Pirnazar et al. 2018) do not provide an optimal management cost. To the best of our knowledge, the present work offers the first approach that addresses the cost-optimal allocation of cloud resources (AMs and VMs) needed for managing SBPs while satisfying their QoS requirements.

In this paper, we consider a cloud scenario where a SaaS provider sells its autonomic resources to IaaS providers. In their turn, IaaS providers offer, to their customers, services (VMs, AMs) with QoS guarantees to host and run their SBPs subject to a set of QoS requirements. The major contributions of this paper are summarized as follows:

- Through studying the characteristics of the cloud and SBPs, we propose an optimal assignment method to select the suitable cloud resource (AM, VM) that satisfies SBP services requirements and resource constraints.
- We extend a popular and widely used cloud computing simulator (CloudSim) (Calheiros et al. 2011) where we model autonomic SBPs. CloudSim provides the ability to model and simulate the execution and the management of SBPs in a cloud environment.

The remainder of this paper is structured as follows. Section 2 reviews the related works on autonomic computing in large-scale environments. Section 3 introduces the necessary background associated with this work. Section 4 describes the proposed approach. Section 5 presents our experiments to validate and evaluate the performance and scalability of our proposal. Section 6 concludes the paper and highlights future research directions.

2 Related work

In the literature, there have been several research works that aim to add autonomic management behaviors to cloud and distributed environments. In the following, we give an overview of some of these works.

IBM is a pioneer in the field of autonomic computing that proposed an autonomic toolkit, which is a set of tools and technologies designed to allow users adding autonomic behavior to their systems. The authors in Jacob (2004) presented all the needed steps to implement autonomic capabilities for resources. One of the main tools is the autonomic management engine that includes representations of the MAPE-K loop. Moreover, IBM suggested several tools to allow managed resources to create log messages using a standard format understandable by the MAPE-K loop. This is achieved using a touch-point that consists of a sensor and an effector. Moreover, an adapter rule builder is proposed to create specific rules in order to generate adaptation plans.

In Ruz et al. (2011), the authors proposed a framework for autonomic management of component-based applications. The different functionalities of a MAPE-K loop (i.e., Monitoring, Analysis, Planning, and Execution) are implemented as separate components, where each component is responsible for a single task. These components are attached to each component of an application for its self-management.

In Belhaj et al. (2017), the authors presented an approach for improving the decision making process of a MAPE-K loop in order to self-adapt a component-based application. The authors equipped the analyzer component with sophisticated learning blocks, where the decision problem of the analyzer component is modeled as a Markov Decision Process with a finite set of states and actions. During each state transition, a reinforcement signal indicates to the proposed decision maker whether it chooses the suitable action or not. In this work, each component of an application is self-managed by its own MAPE-K loop.

In Jacob (2004), the authors proposed a framework for adding self-adaptation mechanisms to software systems. The proposed framework is based on an abstract model that represents an application as a graph. The graph consists of a set of nodes that represents components of an application and a set of arcs that represents interactions between components. The model is continuously adapted using a model manager. The model manager gathers monitoring data from monitoring probes. The collected data is analyzed using an evaluator that is able to detect violations and trigger adaptations. The appropriate adaptation plan is chosen using an adaptation engine and it is applied using an executor.

Authors in Belhaj et al. (2018); Hadded et al. (2021); Quin et al. (2021) adopted a decentralized approach to the autonomic management of SaaS applications. Each AM is dedicated to manage a part of an application, and in most
studies, they recommend the use of an AM per application service in order to improve management.

In their work (Mola and Bauer 2011), the authors focused on the coordination of multiple AMs in the cloud in order to efficiently manage the overall system. AMs are organized in a hierarchical structure, where the higher-level AMs have more authority over lower-level AMs. These latter are responsible for allocating cloud resources, such as CPU and memory, to the web server in order to avoid SLA violations and improve its response time. The AMs communicate by exchanging predefined messages using a message broker.

De de Oliveira et al. (2013) proposed a framework for the coordination of AMs in the cloud. They presented two kinds of AMs known as “AAM” for Application AM and “IAM” for Infrastructure AM. Each application is managed by means of an AAM which is responsible for determining the best architectural configuration as well as the minimum number of VMs required to provide the best QoS under a certain workload. The IaaS cloud layer is managed by a single IAM which manages resource allocation in the infrastructure layer. An IAM holds a public and shared knowledge while each AAM maintains a private knowledge.

Several research works have been devoted to the issue of interaction and coordination of AMs. Broadly speaking, two methods are used for this. The first one splits the knowledge base of each AM into two parts: a public knowledge that is shared with the other AMs and a private knowledge for the AM (de Oliveira et al. 2013). The second method adds AMs that are in charge of coordination (Gueye et al. 2013).

In the state of the art, there are other research works related to autonomic computing (Dehraj1 and Sharma 2021; Ebadifard and Babamir 2020; Kosinska and Zielinski 2020). All of these approaches have been interested in modeling and implementing autonomic environments in a centralized or decentralized manner. In fact, some researchers have dedicated a centralized AM for the management of cloud and distributed systems, which may cause bottlenecks that could hinder the management efficiency. Other works tried to adopt the decentralization of AMs by (i) assigning an AM to each resource, or, (ii) randomly assigning AMs to resources. These works do not take into account the management cost. However, the work presented in Mohamed and Megahed (2015) addressed the optimal assignment of AMs to cloud resources. Beside this approach, many other studies have been devoted to solve optimization problems (Abo-Bakr et al. 2022; Ghasemishabankareh et al. 2020; Sabir 2021; Sabir et al. 2022a, b, c; Umar et al. 2022). However, none of these works can be applied to deal with the problem discussed in this article. These approaches do not take into account the diversity of QoS requirements and the heterogeneity of cloud resources.

To the best of our knowledge, our proposed approach is the first that considers the problem of finding the optimal autonomic management of SBPs in the cloud while ensuring customers’ QoS requirements and optimizing the management cost.

3 Background

In this section, we first introduce and define the key concepts used in this paper. We define the concept of autonomic management and we highlight the SBP definition. We then briefly describe the problem statement.

3.1 Autonomic management

To achieve autonomic computing, IBM has proposed a reference model for autonomic controller (ibm 2005) called autonomic manager, also known as the MAPE-K (Monitor, Analyzer, Planner, Executor, and Knowledge) loop as depicted in Fig. 1.

![Autonomic manager for a cloud resource](https://via.placeholder.com/150)

Fig. 1 Autonomic manager for a cloud resource
3. The **Planner** is responsible for generating adaptation
    plans of actions to avoid violations of QoS;
4. The **Executor** is responsible for carrying out the adapta-
    tion actions over the managed resource.

### 3.2 SBP

An SBP is a set of related services that aim to accomplish a
specific goal (see Fig. 2). A service is the smallest unit of work
that offers computation or data capabilities. Assembling ser-
vice into an SBP can be ensured using any appropriate
service composition specification such as Event-driven Pro-
cess Chain (EPC) (Scheer et al. 2005) and Business Process
Modeling Notation (BPMN) (Group 2011).

We formally define an SBP as a tuple \( (S, G, \tau, E) \) where:

- \( S \) is the non-empty set of services;
- \( G \) is the set of gateways;
- \( \tau : G \rightarrow \{ AND, OR, XOR \} \) is a function that returns
  the type of each gateway. A gateway acts as either a split
  or a join node. Split gateways have exactly one incoming
  edge and at least two outgoing edges. Join gateways have
  at least two incoming edges and exactly one outgoing
  edge;
- \( E \subseteq (S \cup G) \times (S \cup G) \) is the set of edges representing
  the control-flow of the process.

### 4 Proposed approach

In this section, we present our proposed approach for optimal
autonomic management of SBPs in the cloud. Our objective
is to find the optimal allocation of cloud resources for the
management of SBPs.

The customer requests the execution of its SBP subject to
a set of QoS requirements that include the resource require-
ments (CPU, memory and bandwidth) and the minimum
reliability and availability levels for executing the SBP ser-
dvices. The objective of a cloud provider is to execute the
SBP with a minimum management cost while meeting all
consumer requirements. This challenge is addressed through
the following two components (see Fig. 3):

- **Autonomic SBP Optimizer**: finds the optimal manage-
  ment where the cost of allocated resources (VMs, AMs)
  is minimal while respecting business processes QoS
  requirements;
- **Autonomic SBP Simulator**: lies in extending CloudSim
  simulator (Calheiros et al. 2011) with the modeling of (i)
  SBPs, and, (ii) autonomic managers, in order to simulate
  the behavior of autonomic SBPs in a cloud environment.
  The objective is to estimate the management cost and the
  execution time needed for executing and managing SBPs
  with different cloud resource configurations.

### 4.1 Autonomic SBP optimizer

In order to identify the optimal resource allocation, we pro-
pose an exact optimization model. The proposed Integer
Linear Program (ILP) model is defined in terms of its deci-
sion variables, objective function, and constraints. It takes as
inputs:

- An SBP is the tuple \( (S, G, \tau, E) \).
- A service \( s \in S \) is a tuple \( (rre, rav, dt, rcpu, rram, rbw,
  len) \), where \( rre \) and \( rav \) are, respectively, its minimum
  required reliability (%) and availability (%) level. \( dt \) is the
  size of data transferred from this service (MB), \( rcpu \) is
  the minimum required CPU capacity (cores), \( rram \) is the
  minimum required RAM capacity (GB), \( rbw \) is the mini-
  mum required bandwidth (MB/s), and \( len \) is the service’s
  length/size in millions of instructions (MI).
- A VM is defined as a tuple \( (re, av, cp, dtp, cpu, ram, 
  bw, imax, mips) \), where \( re \) and \( av \) are, respectively, the

\[ Springer \]
Fig. 3 Proposed approach for optimal SBP management

capability of the VM in terms of reliability (%) and availability (%). \( cp \) is the computing price ($/hour), \( dtp \) is the data transfer price ($/Mb), \( cpu \) is the CPU capacity (cores), \( ram \) is the RAM capacity (GB), \( bw \) is the bandwidth capacity (MB/s), \( imax \) is the maximum number of instances of the VM, and \( mips \) is the CPU speed in millions of instructions per second (MI/s). We denote by \( V \) the set of VMs owned by the cloud provider, and \( I_{vk} = \{1, 2, \ldots, imax_k\} \) the indexes of the instances of the VM \( k \).

- \( \alpha \) denotes the maximum VM resource consumption (%) in order to avoid the overload of the VM.
- An AM is formalized as a tuple \((rre, rav, mp, dt, rcpu, rram, rbw, imax, len)\), where \( rre \) and \( rav \) are, respectively, its minimum required reliability (%) and availability (%) level. \( mp \) is the AM price ($/hour), \( dt \) is the size of data transferred from the AM to a service (MB), \( rcpu \) is the minimum required CPU capacity (cores), \( rram \) is the minimum required RAM capacity (GB), \( rbw \) is the minimum required bandwidth (MB/s), \( imax \) is the maximum number of instances of the AM, and \( len \) is its estimated size (MI). We denote by \( M \) the set of AMs owned by the cloud provider, and \( I_{mi} = \{1, 2, \ldots, imax_i\} \) the indexes of the instances of the AM \( i \).

- \( \text{dep}(p,k,h): S \times V \times I \rightarrow \{0, 1\} \) is a location function that for each service \( p \) associates 1 if it is deployed in the instance \( h \) of the VM \( k \), and 0 otherwise.

The execution time of an AM \( i \) on a VM \( k \) is calculated by dividing the size of the AM \( len_i \) by the CPU speed \( mips_k \) multiplied by the number of cores of a CPU \( cpu_k \), which can be formulated as:

\[
et_{ki} = \frac{len_i}{mips_k cpu_k}
\]  

We assume that parallel services will not share any resources.

- **Decision variables**
  We define the following decision variables:

- \( X_{ijk} \) is equal to 1 if the instance \( j \in I_{mi} \) of the AM \( i \in M \) is deployed in the instance \( h \in I_{vk} \) of the VM \( k \in V \), and 0 otherwise;
- \( Y_{ijp} \) is equal to 1 if the instance \( j \in I_{mi} \) of the AM \( i \in M \) and the service \( p \in S \) are deployed in the same VM, and 0 otherwise;
- The communication cost is the sum of data transfer costs and takes the following form:

\[
\text{communication cost} = \sum_{i, j, p, k} d_{tp_i} \cdot d_{pl_i} \cdot X_{ijk}\ 
\]

(i) The computing cost is the sum of resources (AMs, VMs) allocation costs in order to manage the SBP. Here, the allocation cost is the AM execution time \(et\) multiplied by the sum of the AM utilization price and the VM utilization price \((cp + mp)\);

(ii) The communication cost is the sum of data transfer costs between AMs and services. The data transfer cost is equal to the bandwidth utilization price \(d_{tp}\) multiplied by the transferred data size \(dt\). The data transfer cost is deemed negligible if the service and the AM assigned to it are running on the same VM.

Hence, the objective function of our mathematical model takes the following form:

\[
\min \sum_{i=1}^{[M]} \sum_{j=1}^{[V]} \sum_{k=1}^{[S]} \sum_{h=1}^{[V]} e_{ti} \cdot (cp_k + mp_i) \cdot X_{ijk}\ 
+ \sum_{p=1}^{[S]} \sum_{i=1}^{[M]} \sum_{j=1}^{[V]} \sum_{k=1}^{[S]} \sum_{h=1}^{[V]} d_{tp_k} \cdot d_{pi} \cdot X_{ijk} \cdot (1 - Y_{ijp}) \cdot Z_{ijp} \quad (2)
\]

- **Cost objective function**

The proposed objective function of the model minimizes the SBP management cost, including the total computing and communication costs:

- **Constraints**

The above objective function is subject to the following set of constraints:

- **QoS constraints:**

(i) VM constraints impose the minimum reliability (3) and availability (4) levels required by an AM on the VM on which it will be executed.

\[
\forall i \in M, \forall j \in I_{mi}, \forall k \in V, \forall h \in I_{vk}, \forall p \in S \ 
re_{vk} \cdot X_{ijk} \geq re_{mi} \cdot X_{ijk}\ 
+ av_{vk} \cdot X_{ijk} \geq av_{mi} \cdot X_{ijk} \quad (3)
\]

(ii) AM constraints impose the minimum reliability (5) and availability (6) levels required by a service on the AM that will manage it.

\[
\forall i \in M, \forall j \in I_{mi}, \forall p \in S \ 
re_{mi} \cdot Z_{ijp} \geq re_{sp} \cdot Z_{ijp}\ 
+ av_{mi} \cdot Z_{ijp} \geq av_{sp} \cdot Z_{ijp} \quad (5)
\]

- **Resource constraints** guarantee that the VM’s capacities in terms of processing, memory, and bandwidth should satisfy services and AMs requirements. These constraints ensure that the resources utilization is less than a threshold \(\alpha\). The term \(\alpha\) forces the system to work away from the saturation point.

\[
\forall i \in M, \forall j \in I_{mi}, \forall k \in V, \forall h \in I_{vk}, \forall p \in S \ 
X_{ijkh} \cdot rcpu_{mi} + X_{ijh} \cdot Z_{ijp} \cdot Y_{ijp} \cdot rcpu_{sp} \leq acpu_k \quad (7)
\]

\[
X_{ijkh} \cdot rram_{mi} + X_{ijh} \cdot Z_{ijp} \cdot Y_{ijp} \cdot rram_{sp} \leq aram_k \quad (8)
\]

\[
X_{ijkh} \cdot rbw_{mi} + X_{ijh} \cdot Z_{ijp} \cdot Y_{ijp} \cdot rbw_{sp} \leq abw_k \quad (9)
\]

- **Assignment constraint** ensures that each service is managed by one AM.

\[
\sum_{i=1}^{[M]} \sum_{j=1}^{[V]} Z_{ijp} = 1 \quad \forall p \in S \quad (10)
\]

- **Placement constraint** implies that each AM should be assigned to only one VM.

\[
\sum_{k=1}^{[S]} \sum_{h=1}^{[V]} X_{ijkh} \geq Z_{ijp} \quad (11)
\]

- **Linearity constraints:** the product of binary decision variables introduces nonlinearity in Eq. 2 and Eqs. 7–9.

- To linearize the objective function 2, a new decision variable \(V_{ijph}\) allows us to remove the product \(X_{ijkh} \cdot (1 - Y_{ijp}) \cdot Z_{ijp}\). For this new variable, the following constraint is incorporated to ensure that the variable \(V_{ijph}\) is equal to 1 when both variables \(X_{ijkh}\) and \(Z_{ijp}\) take a value of 1 and the variable \(Y_{ijp}\) takes a value of 0; otherwise, \(V_{ijph}\) is equal to 0:

\[
\forall i \in M, \forall j \in I_{mi}, \forall k \in V, \forall h \in I_{vk}, \forall p \in S \ 
V_{ijph} \geq X_{ijkh} + Z_{ijp} + (1 - Y_{ijp}) - 2 \quad (12)
\]

Therefore, the objective function 2 can now be expressed as follows:

\[
\min \sum_{i=1}^{[M]} \sum_{j=1}^{[V]} \sum_{k=1}^{[S]} \sum_{h=1}^{[V]} e_{ti} \cdot (cp_k + mp_i) \cdot X_{ijk}\ 
+ \sum_{p=1}^{[S]} \sum_{i=1}^{[M]} \sum_{j=1}^{[V]} \sum_{k=1}^{[S]} \sum_{h=1}^{[V]} d_{tp_k} \cdot d_{pi} \cdot V_{ijph} \quad (13)
\]

- To reformulate constraints 7–9 as linear constraints, a new decision variable \(W_{ijph}\) allows us to remove the
product $X_{ijkh}Z_{ijp}Y_{ijp}$. The variable $W_{ijpikh}$ is equal to 1 if variables $X_{ijkh}$, $Z_{ijp}$ and $Y_{ijp}$ are simultaneously 1; otherwise, it is equal to 0. For this new variable, the following constraint is incorporated:

$$W_{ijpkh} \geq X_{ijkh} + Z_{ijp} + Y_{ijp} - 2$$  \hspace{1cm} (14)

Therefore, equations (7–9) can now be expressed as follows:

$$X_{ijkh,rcpu_{mi}} + W_{ijpikh,rcpu_{sp}} \leq acpu_k$$  \hspace{1cm} (15)

$$X_{ijkh,rram_{mi}} + W_{ijpikh,rram_{sp}} \leq aram_k$$  \hspace{1cm} (16)

$$X_{ijkh,rbw_{mi}} + W_{ijpikh,rbw_{sp}} \leq abw_k$$  \hspace{1cm} (17)

- **Logical constraints** guarantee the relationships between the variables.

$$Y_{ijp} \geq X_{ijkh} + \text{dep}(p, k, h) - 1$$  \hspace{1cm} (18)

$$Y_{ijp} \leq X_{ijkh}$$  \hspace{1cm} (19)

$$Y_{ijp} \leq \text{dep}(p, k, h)$$  \hspace{1cm} (20)

- **Binary constraints** impose that the decision variables $X_{ijkh}$, $Y_{ijp}$, $Z_{ijp}$, and $W_{ijpkh}$ should be either 0 or 1 (binary variable).

$$X_{ijkh} \in \{0, 1\}$$  \hspace{1cm} (21)

$$Y_{ijp} \in \{0, 1\}$$  \hspace{1cm} (22)

$$Z_{ijp} \in \{0, 1\}$$  \hspace{1cm} (23)

$$W_{ijpikh} \in \{0, 1\}$$  \hspace{1cm} (24)

$$V_{ijpikh} \in \{0, 1\}$$  \hspace{1cm} (25)

Our approach relies on an autonomic SBP optimizer to find out the cost-optimal allocation of cloud resources needed for the autonomic management of an SBP. In addition, it encloses an extension of the CloudSim simulator in order to estimate the execution time of the SBP and the cost of the allocated cloud resources. This will be the object of the remainder of this section.

### 4.2 Autonomic SBP simulator

This section briefly presents the CloudSim extension for the autonomic SBP simulation. This extension takes as input a Unified Description Model (UDM) that contains the relationship between SBP services and cloud resources (the output of the ILP). This UDM is parsed using Eclipse Modeling Framework (EMF 1) to extract information about the characteristics of services and cloud resources.

#### 4.2.1 CloudSim extension

Many simulation techniques to investigate the behavior of cloud computing have been developed (Buyya and Murshed 2002; Calheiros et al. 2011; Casanova et al. 2008; Kliazovich et al. 2012; Núñez et al. 2012). One of the most widely used cloud simulators is CloudSim, which is an open source, java-based simulator (Calheiros et al. 2011). CloudSim enables seamless modeling, simulation, and experimentation of cloud computing environments. It supports more functionalities than other simulation tools. Table 1 details a synthesis of cloud simulators.

| Simulator               | Programming language | Resources |
|-------------------------|----------------------|-----------|
| CloudSim Calheiros et al. (2011) | Java | Yes | Yes | No |
| GridSim Buyya and Murshed (2002) | Java | Yes | No | No |
| SimGrid Casanova et al. (2008) | C | Yes | No | No |
| GreenCloud Kliazovich et al. (2012) | C++,OTcl | Yes | No | No |
| iCanCloud Núñez et al. (2012) | C++ | Yes | HPC | No |

As CloudSim is the best choice to simulate cloud computing resources (Oladimeji et al. 2021), we design and implement an extension of CloudSim as illustrated in Fig. 4 (the colored classes). This allows simulating the autonomic management of SBPs in cloud environments.

Before we present our extension, we first introduce the core components of CloudSim:

- **Datacenter**: behaves like an infrastructure cloud providers (e.g., Amazon, Azure, App Engine). It encapsulates a set of physical machines (hosts, servers) that together provide the basic cloud infrastructure;

- **DatacenterBroker**: represents a software that acts as a mediator between end-users requests and the cloud infrastructure. It selects suitable cloud resources that meet QoS needs of users. This class originally simply

1. [http://www.eclipse.org/modeling/emf/](http://www.eclipse.org/modeling/emf/).
makes decisions about what VM to select to place a given Cloudlet;

- **Host**: represents a physical resource (a computer) characterized by a number of CPU, memory, bandwidth, and storage capabilities;
- **VirtualMachine**: represents a software-based emulation of a computer, which is managed by a host;
- **RamProvisioner**: represents the provisioning policy for allocating memory (RAM) to VMs that are deployed on a host;
- **BwProvisioner**: describes the provisioning policy of bandwidth to VMs that are deployed on a host;
- **VmScheduler**: models the policy required for allocating processor cores to VMs running in a host;
- **Cloudlet**: models an application component/service that runs on a VM.

To model autonomic SBPs, the following classes have been designed:

- **SBP**: models the service-based business process to be executed on the cloud. An SBP consists of a set of services that are executed in a specific order according to gateways to achieve a specific business objective;
- **Service**: represents an extended class of **Cloudlet**. This class specifies the QoS requirements that include the resource requirements (CPU, memory and bandwidth) and the minimum reliability and availability levels for executing a service;
- **Gateway**: defines how an SBP behaves;
- **AutonomicManager**: represents an AM;
- **Monitor, Analyzer, Planner, Executor**: each one is an extended class of **Cloudlet**.

As shown in Fig. 4, the white classes are the main components of the CloudSim simulator. The blue classes depict an AM, while the orange ones describe an SBP.

### 4.2.2 Unified description model

In this section, we present the description of our Unified Description Model (UDM) which describes SBP control-flow, QoS requirements and allocated cloud resources. It is defined as an eXtensible Markup Language (XML) document and it is divided into two parts: (i) SBP, and, (ii) cloud resources. Listing 1 shows an extract of the UDM model of the SBP presented in Fig. 2.

The **Process** element is composed of a set of nodes and edges. The nodes correspond to the services and the gateways of the SBP. The edges describe the relationship between nodes. In the UDM, each service has a set of attributes such as unique identifier, size, and the required cloud resources. The resource element identifies the cloud resource (VM, AM) needed by the service. For instance, lines 5–6 in Listing 1 depict that the service ReceiveOrder requires VM1 and AM4 as resources.

The **Resources** element describe the set of resources offered by the extended CloudSim tool. In this paper, we consider the following resources: (i) VMs characterized by type, number of CPU cores, memory (RAM), and bandwidth, (ii) AMs characterized by its reliability and availability capability.
5 Validation and evaluation

In this section, we present experiments designed to evaluate the quality and the performance of our approach, measured in terms of management cost and response time as well as its flexibility measured in terms of the ability of the ILP to cope with new constraints and new resource capacities.

Our approach is evaluated on two public real datasets of business process models from IBM Fahland et al. (2011) and the SAP reference model (Keller and Teufel 1998). The first experiment compares the performance of our method against our former work (Hadded et al. 2016). The second experiment investigates the impact, on the objective function and response time of our ILP, of adding a deadline constraint as well as scaling up and down the cloud resources capacities. However, to the best of our knowledge, there are no other existing research studies that focus on the problem addressed in this paper. Therefore, we are not able to cover other comparisons.

All experiments were carried out on a laptop equipped with an Intel® Core™ i7-4750HQ with 2.00 GHz processor and 12 GB of memory. The commercial solver CPLEX 12.6.3 (IBM 2022) is used to solve the optimization problem.

5.1 Experimental parameters

In our experiments, we use real business process models from two large public datasets:

1. The first one is shared by an IBM research group. This dataset consists of 560 business processes (Juric 2006) designed for telecommunications, financial services, and other domains. It is presented in XML format following the BPMN standard. The number of services in these processes varies between 2 and 69.
2. The second one is from the SAP reference model (Keller and Teufel 1998) which contains 205 models in EPC notation (Scheer et al. 2005). The number of services in these processes is between 1 and 43.

The VM configurations are based on the current Amazon EC2 and are given in Table 2. In these experiments, the maximum number of VM instances is randomly generated between 1 and $\frac{n}{2}$, where $n$ is the number of the SBP services.

Another important parameter of the experiments is the maximum percentage of VM resource consumption ($\alpha$). It is set to 90%. Table 3 shows the randomly generated AMs data input.

5.2 Comparison with our former proposal

Our former approach presented in Hadded et al. (2016) consists of first determining the best mapping of AMs to services that minimizes the number of AMs (computing cost) while avoiding bottlenecks, and then, finding the best mapping of AMs to VMs that minimizes the overall communication cost.

As shown in Fig. 5, the proposed solution is cheaper than our former one. The average management cost decreases from 8.761$ to 4.160$. It reduces the cost by 52.517%. This decrease can be explained by the fact that the aim of the proposed ILP is to simultaneously minimize two objectives: the computing and communication costs, while in our former work these objectives are optimized separately by using two different ILP, resulting in a higher communication cost.

5.3 ILP flexibility

In order to demonstrate the flexibility of our approach, we first extend the proposed ILP by adding a new constraint considering the total execution time of the SBP. Then, we increase and decrease the maximum number of instances for each VM type.

http://aws.amazon.com/fr/ec2/.
Table 2 VM configurations

| VM type | CPU (cores) | CPU capacity (MIPS) | RAM (GB) | Bandwidth (MB/s) | Reliability (%) | Availability (%) | Price ($/hour) | Data transfer ($/MB) |
|---------|-------------|---------------------|---------|-----------------|----------------|-----------------|----------------|---------------------|
| Micro   | 1           | 200                 | 2       | 100             | 75             | 65              | 0.133          | 0.004               |
| Small   | 2           | 400                 | 4       | 1000            | 80             | 70              | 0.266          | 0.007               |
| Medium  | 4           | 600                 | 8       | 2000            | 85             | 75              | 0.532          | 0.015               |
| Large   | 8           | 800                 | 16      | 3000            | 90             | 80              | 1.064          | 0.031               |
| × Large | 16          | 1000                | 32      | 4000            | 95             | 85              | 2.128          | 0.062               |
| 2 × Large | 32         | 2000                | 64      | 8000            | 99             | 95              | 4.256          | 0.124               |

Table 3 The inputs ranges used to create AMs with different attribute values

| Information          | Range           |
|----------------------|-----------------|
| AM number            | [1..10]         |
| Maximum instance number of AM | [1..n] |
| AM's utilization price | [0.01..5] |
| Requirement in reliability | [0..100] |
| Requirement in availability | [0..100] |
| Requirement in CPU   | [1..32]         |
| Requirement in RAM   | [0..64]         |
| Requirement in bandwidth | [0..8000] |
| Length of AM         | [1..100000]     |

Fig. 5 The management cost using the proposed ILP and our former proposal

5.3.1 Deadline constraint

The evaluation relies on adding a new constraint which imposes that all services in the SBP must be executed before a deadline required by the customer.

\[ \text{makespan} \leq \text{deadline} \quad (26) \]

The makespan (i.e., the completion time) of an SBP consists of two parts (see Eq. 27), which are the computing time \( ct \) of all services in the SBP and the transmission time \( tt \) among these services (see Eq. 29).

\[ \text{makespan} = ct + tt \quad (27) \]

\[ ct = \sum_{i=1}^{[M]} \sum_{j=1}^{[I]} \sum_{k=1}^{[V]} \sum_{h=1}^{[I]} e_{ki} X_{ijkh} \quad (28) \]

\[ tt = \sum_{p=1}^{[S]} \sum_{i=1}^{[M]} \sum_{j=1}^{[I]} \sum_{k=1}^{[V]} \sum_{h=1}^{[I]} \sum_{t=1}^{[I]} dt_{pi} b_{w_{k} I_{ijh}} X_{ijkh} (1 - Y_{ijp}) Z_{ijp} \quad (29) \]

Table 4 shows that, in average, the proposed ILP reaches the optimal solution in a few seconds (3.002s in average). However, solving the problem of minimizing the monetary cost of managing SBPs under a deadline constraint, the ILP reaches the optimal solution but it requires higher computational time. The average computational time is increased from 3.002s to 8.356s and the average management cost increases from 4.162$ to 6.053$. To sum up, we conclude that the proposed ILP is flexible and tries to find solutions in a reasonable time. Although by adding a new constraint, it is more difficult to obtain the optimal solutions.

5.3.2 Resource scaling

The evaluation associated with this experiment is based on scaling up and down the maximum number of VM instances. First, we scaled up the maximum number of VM instances from \( n/2 \) to \( n \). As depicted in Table 5, our proposal reaches the optimal solution in a matter of seconds (3.484s). It reaches the optimal solution even better. Thus, the average management cost decreases from 4.162$ to 2.665$ thanks to the fact that, by doubling the number of VM instances, AMs are more likely to deploy in VMs with cheaper prices.

Second, we scaled down the number of VM instances to 1. As shown in Table 5, the ILP finds the optimal solution in a reasonable time (2.122s) and the average management cost increases from 4.162$ to 7.617$.
### Table 4  Management costs and response times for the ILP while adding the deadline constraint

| Services number | Proposed ILP | ILP with deadline constraint |
|-----------------|--------------|------------------------------|
|                 | Obj. fct ($) | Response time (s) | Obj. fct ($) | Response time (s) |
| [1..10]         | 0.356        | 0.428               | 1.218        | 0.313             |
| [11..20]        | 1.247        | 1.357               | 2.409        | 3.145             |
| [21..30]        | 2.612        | 2.191               | 3.621        | 5.602             |
| [31..40]        | 3.803        | 2.879               | 5.390        | 8.269             |
| [41..50]        | 5.397        | 3.955               | 7.556        | 10.778            |
| [51..60]        | 7.503        | 4.591               | 10.312       | 12.539            |
| [61..69]        | 8.216        | 5.613               | 11.871       | 17.847            |
| **Average**     | 4.162        | 3.002               | 6.053        | 8.356             |

### Table 5  Management costs and response times for the ILP while scaling up and down the number of VM instances

| Services number | Proposed ILP | ILP with resource scaling down | ILP with resource scaling down |
|-----------------|--------------|-------------------------------|-------------------------------|
|                 | Obj. fct ($) | Time (s) | Obj. fct ($) | Time (s) | Obj. fct ($) | Time (s) |
| [1..10]         | 0.356        | 0.428   | 0.356        | 0.183    | 0.356        | 0.428   |
| [11..20]        | 1.247        | 1.357   | 3.126        | 0.826    | 1.583        | 2.702   |
| [21..30]        | 2.612        | 2.191   | 4.862        | 1.569    | 1.583        | 2.702   |
| [31..40]        | 3.803        | 2.879   | 6.693        | 2.061    | 2.327        | 3.457   |
| [41..50]        | 5.397        | 3.955   | 9.418        | 2.562    | 3.232        | 4.680   |
| [51..60]        | 7.503        | 4.591   | 13.285       | 3.412    | 4.912        | 5.247   |
| [61..69]        | 8.216        | 5.613   | 15.583       | 4.247    | 5.409        | 6.321   |
| **Average**     | 4.162        | 3.002   | 7.617        | 2.122    | 2.665        | 3.484   |

### 5.4 Conclusion

The obtained experimental results demonstrate that the proposed ILP is effective in terms of both quality and response time. Whenever the number of services is considerable, our proposal reaches the optimal solution in a reasonable time. We can conclude that it is much better than our former approach. In fact, a gain of 52.517% was measured on the management cost. Moreover, the ILP can be easily adapted to cope with new constraints and different resource capacities.

### 6 Conclusion

Managing service-based business processes in the cloud requires using AMs in order to handle the dynamic nature of the cloud environment. This new trend still faces several challenges such as how to find a cost-optimal allocation of cloud resources needed for the autonomic management of an SBP. However, this problem becomes more complex due to the exponential search space closely related to the number of possible VMs-AMs combinations as well as the QoS requirements of the SBP that further complicates the search.

In this paper, we proposed an approach for optimal autonomic management of SBPs in the cloud. To do so, we solved our optimization problem using a mathematical formulation. More precisely, we proposed a linear programming model that is composed of an objective function and a set of constraints. The objective function consists in minimizing the management cost under various constraints such as services requirements and resource constraints. Moreover, we extended the CloudSim simulator in order to validate our approach under realistic working conditions. We then performed extensive experiments and computed the performance in terms of the cost of the allocated cloud resources and response time. We evaluated our approach on two real datasets of business processes. The experimentation results show the effectiveness, performance, and flexibility of our approach.

Besides, in our experimentation, we considered that the number of SBP services is about 70. Thus, in the future, we plan to deal with SBPs composed of a large number of services such as scientific workflows. Over the last decades, researchers are significantly applying soft computing techniques in cloud computing (Gupta et al. 2018). These techniques are able to give effective solutions for many challenges and issues in cloud computing. One of the basic techniques is genetic algorithm that has been frequently applied to solve combinatorial optimization problems (Chandran et al. 2020; Iranmanesh and Naji 2021; Soulegan et al. 2022).
Therefore, we plan to implement a genetic algorithm-based approach for optimal autonomic management of SBPs in order to search for near-optimal solution when dealing with a large number of services. Furthermore, we aim to test the proposed approach on a real cloud.

Author Contributions LH and TH were responsible for conception and design of study and analysis and/or interpretation of data. LH was responsible for acquisition of data and drafted the manuscript. TH performed critical revision.

Declarations

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

Ethical approval This article does not contain any studies with human participants or animals performed by the authors.

References

Abo-Bakr RM, Mohamed NA, Mohamed SA (2022) Meta-heuristic algorithms for solving nonlinear differential equations based on multivariate Bernstein polynomials. Soft Comput 26:605–619

An architectural blueprint for autonomic computing. Technical report, IBM (2005)

Belhaj N, Belaid D, Mukhtar H (2017) Self-adaptive decision making for the management of component-based applications. In: CoopIS. Springer, pp 570–588

Belhaj N, Belaid D, Mukhtar H (2018) Framework for building self-adaptive component applications based on reinforcement learning. In: IEEE International conference on services computing, SCC. IEEE

Buyya R, Mursheed M (2002) Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurr Comput Pract Exp 14

Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41:23–50

Casanova H, Legrand A, Quinson M (2008) SimGrid: a generic framework for large-scale distributed experiments. In: International conference on computer modeling and simulation. IEEE, pp 126–131

Chandran R, Kumar SR, Gayathri N (2020) Genetic algorithm-based Tabu search for optimal energy-aware allocation of data center resources. Soft Comput 24:16705–16718

de Oliveira F, Ledoux T, Sharrock R (2013) A framework for the coordination of multiple autonomic managers in cloud environments. In: International conference on self-adaptive and self-organizing systems. IEEE, pp 179–188

Dehraj P, Sharma A (2021) A review on architecture and models for autonomic software systems. J Supercomput 77:388–417

Ebadifard F, Babamir SM (2020) Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. Clust Comput 24:1573–7543

Fahlund D, Favre C, Koehler J, Lohmann N, Völzer H, Wolf K (2011) Analysis on demand: instantaneous soundness checking of industrial business process models. Data Knowl Eng 70:448–466

Ghasemibashbankareh B, Ozlen M, Li X, Deb K (2020) A genetic algorithm with local search for solving single-source single-sink nonlinear non-convex minimum cost flow problems. Soft Comput 24:1153–1169

Group OM (2011) Business process model and notation (BPMN). Technical report

Gueye SMK, De Palma N, Rutten É (2013) Component-based autonomic managers for coordination control. In: Coordination

Gupta BB, Agrawal DP, Yamaguchi S, Sheng M (2018) Advances in applying soft computing techniques for big data and cloud computing. Soft Comput 22:7679–7683

Hadded L, Charrada FB, Tata S (2021) Optimizing autonomic resources for the management of large service-based business processes. IEEE Trans Serv Comput 14:779–790

Hadded L, Charrada FB, Tata S (2016) Optimization and approximate placement of autonomic resources for the management of service-based applications in the cloud. In: CoopIS. Springer, pp 175–192

Hadded L, Charrada FB, Tata S (2018) Efficient resource allocation for autonomic service-based applications in the cloud. In: IEEE international conference on autonomic computing (ICAC). IEEE, pp 193–198

IBM (2022) IBM CPLEX optimizer. http://www.ilog.com/products/cplex/

Iranmanesh A, Naji HR (2021) DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. Clust Comput 24:667–681

Jacob B (2004) A practical guide to the IBM autonomic computing toolkit. IBM Corporation, International Technical Support Organization

Jacobson L, Booch JRG (2021) The unified modeling language reference manual

Jamshidi P, Ahmad A, Pahl C (2014) Autonomic resource provisioning for cloud-based software. In: International symposium on software engineering for adaptive and self-managing systems (SEAMS). IEEE, pp 95–104

Juric MB (2006) Business process execution language for web services BPEL and BPEL4WS. Packt Publishing, Birmingham

Keller G, Teufel T (1998) Sap R/3 process oriented implementation. Addison-Wesley Longman, Boston

Kliazovich D, Bouvry P, Khan SU (2012) Greencloud: a packet-level simulator of energy-aware cloud computing data centers. J Supercomput 62:1263–1283

Kosinski J, Zielinski K (2020) Autonomic management framework for cloud-native applications. J Grid Comput 18:779–796

Marino J, Rowley M (2009) Understanding SCA (service component architecture). Addison-Wesley Professional

Mell PM, Grance T (2011) The NIST definition of cloud computing. Technical report

Mohamed M, Amziani M, Belaid D, Tata S, Mellit T (2015) An autonomic approach to manage elasticity of business processes in the cloud. Future Gener Comput Syst 50:49–61

Mohamed M, Megahed A (2015) Optimal assignment of autonomic managers to cloud resources. In: IEEE international conference on service operations and logistics, and informatics (SOLI). IEEE, pp 88–93

Mola O, Bauer MA (2011) Collaborative policy-based autonomic management: in a hierarchical model. In: International conference on network and service management. IEEE

Núñez A, Vázquez-Pololeti J, Caminero A, Castañé G, Carretero J, Llorente I (2012) ICanCloud: a flexible and scalable cloud infrastructure simulator. J Grid Comput 10:185–209

Oladimeji O, Oyeyiola D, Oladimeji O, Oyeyiola P (2021) A comprehensive survey on cloud computing simulators. Sci J Inf 8:51–59

Pirnazar M, Hasheminasab H, Qasemy Z, Mohri-Esfahani E, Eslamian S, Ostad-Ali-Akaki K (2018) The evaluation of the usage of the fuzzy algorithms in increasing the accuracy of the extracted land use maps Elham Mohri-Esfahani. Int J Glob Environ Issues 17:307–321
Quin F, Weyns D, Gheibi O (2021) Decentralized self-adaptive systems: a mapping study. In: International symposium on software engineering for adaptive and self-managing systems (SEAMS). IEEE

Ruz C, Baude F, Sauvan B (2011) Flexible adaptation loop for component-based SOA applications. In: ICAS

Sabir Z (2021) Stochastic numerical investigations for nonlinear three-species food chain system. Int J Biomath

Sabir Z, Baleanu D, Ali M, Sadat R (2022) A novel computing stochastic algorithm to solve the nonlinear singular periodic boundary value problems. Int J Comput Math 5:1–14

Soulegan NS, Barekatain B, Neysiani BS (2021) MTC: minimizing time and cost of cloud task scheduling based on customers and providers needs using genetic algorithm. Int J Intell Syst Appl 13:38–51

Umar M, Sabir Z, Raja MAZ, Albasyouni K, Mahmoud S, Guerrero Sánchez Y (2022) An advance computing numerical heuristic of nonlinear SIR dengue fever system using the Morlet wavelet kernel. J Healthc Eng 2022:1–14

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.