A Survey of Evolutionary Computation for Resource Management of Processing in Cloud Computing

Mateusz Guzek[1], Pascal Bouvry[2], and El-Ghazali Talbi[3]

[1]SnT, University of Luxembourg, Luxembourg
[2]FTSC/CSC, University of Luxembourg, Luxembourg
[3]LIFL, University of Lille, France

Abstract

Cloud computing is significantly reshaping the computing industry. Individuals and small organizations can benefit from using state-of-the-art services and infrastructure, while large companies are attracted by the flexibility and the speed with which they can obtain the services. Service providers compete to offer the most attractive conditions at the lowest prices. However, the environmental impact and legal aspects of cloud solutions pose additional challenges. Indeed, the new cloud-related techniques for resource virtualization and sharing and the corresponding service level agreements call for new optimization models and solutions. It is important for computational intelligence researchers to understand the novelties introduced by cloud computing. The current survey highlights and classifies key research questions, the current state of the art, and open problems.

I. Introduction

Cloud Computing (CC) resource management is a novel, important, and fascinating domain of research. The resource management of CC systems presents a diverse and interesting mixture of challenges that Computational Intelligence (CI) must address. These include large-scale optimization problems with dynamics and uncertainty, resulting from multiple users with a variety of requirements and objectives. Most CC resource management problems are NP-hard, which means that there are no fast, optimal algorithms to solve them. While classical problems such as scheduling, mapping, and load balancing have already been addressed by the CI community using techniques such as evolutionary optimization, and fuzzy and neural controllers, the specific properties of CC require that we revisit them. Our survey starts with a clarification of the concepts of CC and explains its fundamentals.

A. Cloud Computing and Scope of the Survey

The ISO/IEC 17788:2014 standard defines CC as a "[p]aradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand. (...) Examples of resources include servers, operating systems, networks, software, applications, and storage equipment." In order to define and constrain purchased cloud services, the cloud service provider and the user enter into a contract. This contract is usually known as a Service Level Agreement (SLA).

B. Service and Deployment Models

Cloud service models are commonly described as X-as-a-Service and abbreviated to XaaS. X represents the entity consumed as a service. Typically, the three main levels of cloud are Software-, Platform-, and Infrastructure-as-a-Service [1] (SaaS, PaaS, and IaaS, respectively). As the cloud market develops, other offerings are appearing to widen the cloud spectrum, including high-level Business Process (BPaaS) or low-level Hardware (HaaS). All these service models can be presented as a multi-layered structure, where higher (more complex) service models may depend on lower service models, as presented in Table I.

Cloud deployment models can be classified into four groups [1]: *private clouds*, used by a single organization, *community clouds* provisioned for a community of consumers, *public clouds* open to the general public, and *hybrid clouds* combining different deployment models.

C. Resource Sharing and Virtualization

CC intrinsically relies on resource sharing. Sharing is typically transparent to the users but requires management by providers. However, in specific cases, explicit content sharing is still required [2]. Virtualization is commonly used to create pools of virtual resources on top of physical resources. Virtualization techniques enable partition of physical resources and ensure isolation of different user spaces. Isolation is indeed crucial to ensure security and performance warranties [3]. The multi-tenancy of physical resources results in challenges for each of the above-mentioned aspects of isolation [4].

(c) 2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works. DOI: 10.1109/MCI.2015.2405351

TABLE I. DESCRIPTIONS AND ORDER OF SERVICE MODELS AND THE RELATED RESOURCE MANAGEMENT PROBLEMS.

Service Model	Description	Resource Management Challenges
Business Process-as-a-Service (BPaaS)	Business process outsourcing to cloud	Quality of Service (QoS) and the cost of processes
Software-as-a-Service (SaaS)	Complete applications, accessible via thin clients	Performance and cost of an application
Platform-as-a-Service (PaaS)	Platforms to develop and run applications	Performance of a platform given available resources
Infrastructure-as-a-Service (IaaS)	Virtualized infrastructures	Allocation of virtual resources on the hardware
Hardware-as-a-Service (HaaS)	On-demand access to fully configurable hardware	Utilization of the hardware

D. Taxonomy of Cloud Computing Resource Management

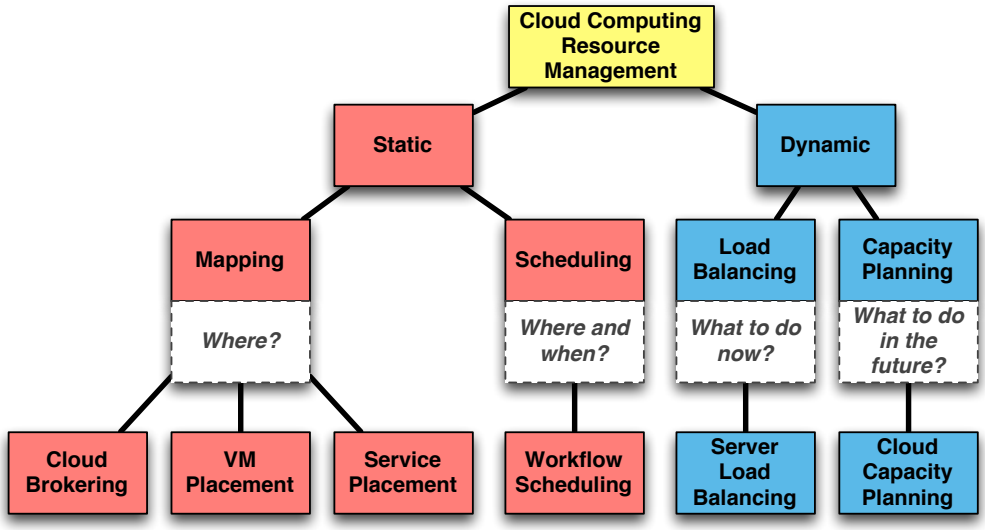

Fig. 1. A simple taxonomy of CC resource management problems.

The field of CC resource management has its roots in distributed and parallel systems resource management. In this paper, we propose a simple taxonomy of CC resource management problems (Fig. 1), based on the more elaborated taxonomy of Casavant and Kuhl [5].

The main criterion for the classification of CC resource management problems is based on the problem's dynamics. In the case of static (or offline) problems, the full list of requests is known a priori, while for dynamic (or online) problems, resource requests become apparent (or may change) over time. CC typically offers on-demand services. Therefore CC resource management has a tendency to be intrinsically dynamic. Indeed, although future users' behaviors and interactions are unknown, the system must remain operational. In some less delay-sensitive use cases, such as batch scheduling or load balancer training, the optimization can be performed offline. A detailed discussion on specific CC resource management problems is presented in Section III. Most resource management problems are by nature NP-hard [6], [7]. CI tools seem to be a natural answer to the resulting complexity, uncertainty, scale, and dynamics.

E. Review of Applicable Computational Intelligence Tools

Evolutionary computation is commonly used to solve combinatorial optimization problems. However, given its computational cost, it is more suited to static problems. The need for fast, real-time response to dynamic problems highlights the efficiency of Artificial Neural Networks (ANNs) and Fuzzy Systems (FSs), which, after a training phase, can react quickly. The dynamic CC environment presents challenges concerning the incremental and robust training of these tools. The resulting solutions are often hybrid systems, in which advanced behaviors and algorithms are combined with faster and simpler systems for real-time decision making.

II. Cloud Computing Resource Management Models

A. Classic Resource Management Concepts

Processing systems models are commonly divided into descriptions of the workload and the system. Scheduling systems are often described using the three-field notation [7], [8], while queueing systems can be expressed in terms of the Kendall's notation [9]. The scheduling problem consists in allocating tasks to machines in given time slots, while queuing theory studies the behavior of the service systems that execute arriving tasks.

In contrast to these time-dependent concepts, in some CC problems the time dimension may be absent, e.g. in the case of long-term allocations of Virtual Machines (VMs) or services [10], or in the case of applications that are tolerant to delays. Another approach to scheduling consists in first clustering the tasks, and then assigning the clusters to the processors [11]. Such simplification leads to the mapping problem, which can be reduced to the bin-packing problem, which is known to be NP-hard [12]. Bin-packing problems have online and offline variations. The latter solves the problem task by task, while the former solves the problem in batches. VM colocation can be used to reduce total resource consumption [13].

This survey section underlines the most important characteristics of CC workloads and systems. A set of relevant CC resource management problems will then be proposed and classified using these characteristics.

B. Workload Models

CC workloads are submitted by cloud users, i.e. individuals or organizations that use cloud services. The tasks may require multiple resource types and additional objects, such as input or output data. The granularity of a task may vary from a simple HTTP request, to the execution of a larger process up to the whole life cycle of a VM.

In the most simple case, the tasks are independent, i.e. the success and finish time of each task are accounted separately. More complex models involve related tasks. This typically implies that the system response time is defined as the finish time of the last task in a group. Two such models are: the Bag of Tasks (BoT) model, in which a set of tasks has no additional constraints; and the Directed Acyclic Graph (DAG) model featuring precedence constrained tasks [14], [15]. A special subset of workflow applications are multi-tier applications, which involve simple and repetitive patterns of dependencies between tasks, representing different layers of Web services [16].

The behavior of users, or groups of users, can vary over time. This aspect is noticeable at the higher CC layers as changes in users' requests [17] and at the lower CC layers as variable resource utilization [18], [19]. To summarize, cloud usage patterns are heterogeneous due to the varying user needs.

C. System Models

In the context of CC, the notion of a machine (or a processor) covers a wide range of concepts and differs between studies. It may stand not only for a physical server, but also for a VM or even a whole data center. VMs are treated as processors or as pieces of a workload depending on the layers of the cloud resource management problem involved. Some models and studies propose a multi-layer approach, where tasks are allocated to VMs, which are further allocated to servers [20], [21]. Processors in a system may be homogenous, i.e. have identical characteristics, or heterogeneous. In the latter case, the execution time of a task depends of the allocated machine, as machines vary in capabilities and speed.

New or standardized data centers may be composed of homogeneous machines. As time passes, and new hardware is shipped to data centers, they are either totally refurbished or, more often, become heterogeneous. Machines can be described by the resource types they provision and their location in the data center network.

Data center networks are crucial for ensuring CC performance. Important considerations include network topology, link latency and bandwidth. Because of their properties, links can be modeled as a set of machines [22]. Communication activities may then be directly modeled, e.g. as a separate class of tasks in DAGs [23]. Some studies neglect the impact of networking, ignoring transmission time, and the delays and congestion that may occur. While peer-to-peer concepts are seldom explicitly mentioned in the context of CC, a peer-to-peer network topology is often used to express the connectivity between data centers. However, some more realistic models involve consideration of network topology, including the modeling of switches and routers as networking nodes.

Spatially-distributed cloud systems are composed of multiple data centers, which increases problem complexity. Geographic location is linked to the availability of auxiliary resources (electrical energy and its source, cool air and water) and the existence of laws or regulations that can result in restrictions on the ways data are processed. If multiple data centers are run by different providers, the system can be called a multi-cloud. If the providers cooperate to provide a common platform, the system is called a cloud federation.

D. Objectives

The standard objectives of resource management in IT systems (summarized in Table II) are the response time for a user, and the throughput for a service provider. These objectives can be further defined using the notions of *makespan*, which denotes the

maximum completion time for a batch of tasks, and the *mean flowtime*, which is defined as the mean time spent by tasks in the system. SLAs often promote service uptime as a critical performance metric, i.e. the percentage of time that the service is available to its users. Uptime can be related to the success rate, which is the ratio of the number of successfully executed tasks to the total number of tasks.

TABLE II. CLASSIFICATION OF CLOUD RESOURCE MANAGEMENT OBJECTIVES.

Objective group	Objectives
Performance	Response Time, Uptime, Throughput, Makespan
Financial	Price, Income, Cost
Environment	Energy, Peak power, Thermal, CO_2 Emissions
Others	Reliability, Security, Data Center Location

The service-related aspects of CC offer a second group of objectives, which are cost-related. As in other economic settings, cloud customers are interested in obtaining the best performance for the lowest price. On the other hand, the providers' goal is to maximize their income, by maximizing revenue and minimizing costs.

One of the emerging objectives is the minimization of the environmental impact, driven by the policies of governments and companies, the technical challenges in reliable provisioning of high power, and the cost of electricity for large-scale IT systems. Environmental objectives include the minimization of electrical energy use, the peak power draw of the systems, cooling costs, and CO_2 emissions.

Finally, there is a more diverse group of additional objectives or constraints. These include auxiliary factors resulting from highly-distributed and shared cloud environments, including reliability of scheduling on heterogeneous infrastructures, security in the case of distribution and multi-tenancy, and, last but not least, the appropriate location of computation and storage of data for legal compliance.

Multiple objectives are solved either by aggregating them into a single objective, by applying hierarchical optimization of consecutive objectives, or by using the Pareto optimality approach [24]. In the latter case, the result of optimization is a set of non-dominated solutions, i.e. each solution on the Pareto front is better than the others in at least one objective.

E. Real Implementations of Cloud Computing Systems

Models are in essence simplifications of reality. The two following examples of real cloud implementation platforms illustrate the challenges of modeling. Amazon and Google run industrial-scale clouds that share many common traits. We focus on the IaaS offers of these two providers. The Amazon Web Services (AWS) cloud is currently composed of 10 regions[1], each composed of multiple data centers, while the Google Computing Platform global infrastructure consists of 3 regions, each located on a different continent, which include 8 zones in 12 geographic locations[2]. Additionally, the AWS cloud includes 52 edge locations all over the world, which support localized content delivery networks and Domain Name System (DNS) services. Amazon regions have heterogeneous capabilities, i.e. each of them supports a different subset of functionalities. Amazon relies on a modified version of the Xen [25] hypervisor, while Google uses the KVM [26] hypervisor[3].

III. CLOUD COMPUTING RESOURCE MANAGEMENT OPTIMIZATION

This section describes a selection of studies which apply CI tools to CC resource management optimization problems. Following the taxonomy presented in Section I-D, the problems are divided into static and dynamic classes. Table III presents a coarse-grained classification of all considered studies, organizing them by problem and the CI method used to solve them. Finally, Fig. 2 maps the state of the art in CC resource management to the layers of CC.

A. Static Problems

1) Cloud Brokering: In some cases, the workload can be submitted to the cloud by an intermediary, known as a cloud broker (Fig. 3). There may be multiple reasons for using the services of such an intermediary, including the access to a wider or better set of service offerings, or the expertise available to optimize resource selection. In the most general form, cloud brokering is the process of matching requests (typically VMs or jobs that aggregate multiple VMs) from multiple users to the offers provided by multiple clouds.

The fundamental objective of cloud brokering is the minimization of the allocation price, often coupled with some Quality of Service (QoS) objectives (response time, user satisfaction). In all the cases described, no dependencies between VMs or jobs were considered. Multiple clouds may vary their offers, increasing problem heterogeneity. The literature concerning the cloud brokering problem is summarized in Table IV.

[1]http://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

[2]http://www.google.com/about/datacenters/inside/locations/index.html

[3]https://cloud.google.com/compute/docs/faq

TABLE III. CLASSIFICATION OF APPLICATION OF COMPUTATIONAL INTELLIGENCE TOOLS TO CLOUD COMPUTING RESOURCE MANAGEMENT PROBLEMS.

	Problem	EA	ANN	FS	ACO & Bee	PSO	SA	Others	Hybrid
Static	Cloud Brokering	[27], [28], [29], [30], [10]			[30]	[30]	[27]		[31]
	VM Placement	[32]		[32]	[33], [34], [35]	[36]	[37]	[21]	
	Service Placement	[38], [39], [40], [16], [14], [41], [42], [43]				[42]	[44]		
	Workflow Scheduling	[45], [46], [47], [48], [15], [49], [50]			[15]	[47], [51], [48], [15]	[37]	[21]	[52], [53]
Dynamic	Capacity Planning	[54], [18]	[17], [18]	[55]			[19]		
	Server Farm Load Balancing		[56]	[55]	[57]			[58], [59]	[56]

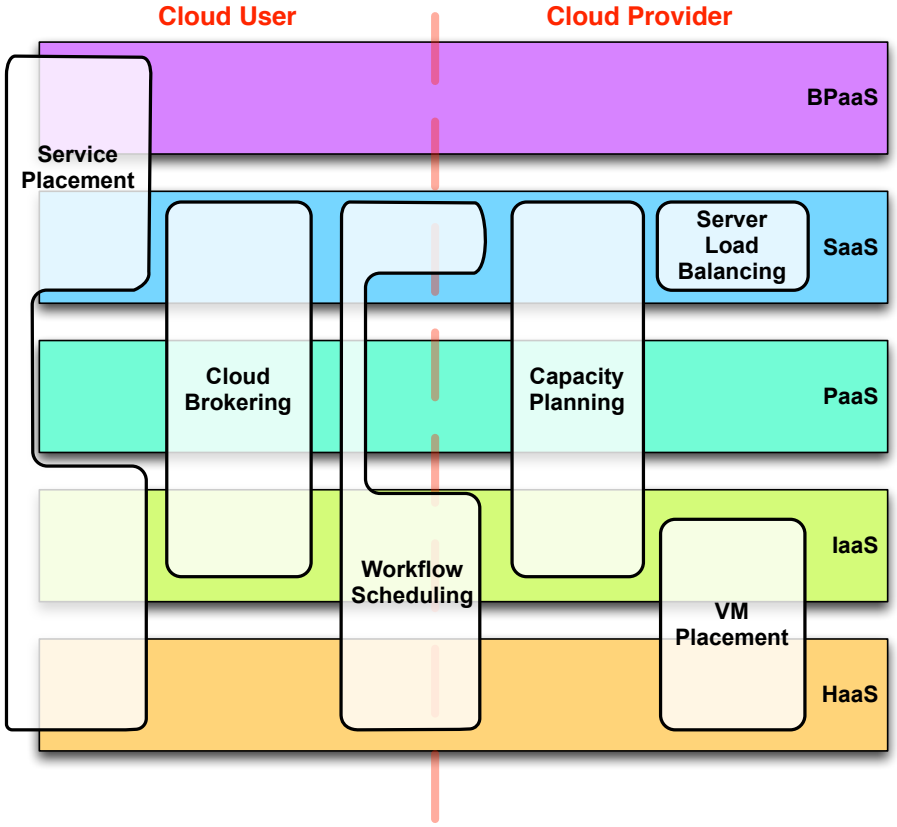

Fig. 2. Mapping of state-of-the-art CI solutions for resource management problems to CC layers.

The first approach to cloud brokering exploits the hypothesis that cloud providers offer a homogeneous infrastructure and that the pool of resources is unlimited. It is therefore only necessary to select a VM type for each task. This approach most commonly uses a vector-based representation of a solution, where each position is a task, and the value of the positions determines the selected VM type.

Kessaci et al. [28] propose MOGA-CB, a multi-objective Genetic Algorithm (GA) for the minimization of response time and the cost of VM instances. The algorithm schedules instances each containing 10,000 VM requests, divided into 30s rounds and

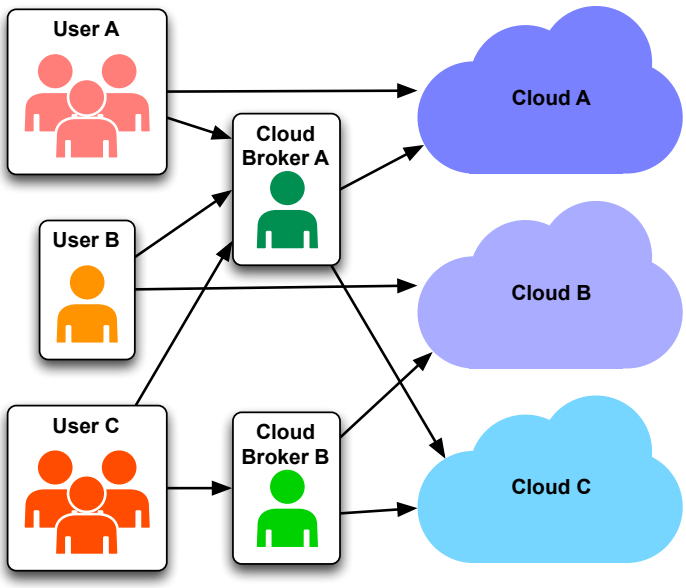

Fig. 3. Cloud brokers in a workload submission process.

TABLE IV. COMPARISON OF CLOUD BROKERING OPTIMIZATIONS.

Method	Technique	Workload Type	System	Resources	Objectives	Multi-Objective
MOGA-CB [28]	GA	VM Requests	Cloud	VM Type	Response Time, Cost	Pareto
Kessaci et al. [29]	GA	VM Requests	Cloud Federation	Clouds (Data Centers)	Energy, CO_2, Cost	Pareto
CDOXplorer [27]	GA	Services	Multi-cloud	VM Type	Response Times, Costs, and SLA Violations	Pareto
Legillon et al. [10]	GA	Services	Multi-Cloud	VM Type	Cost	No
CLOUDRB [30]	PSO (ref. ACO, GA)	Jobs	Multi-cloud	Cloud Resources	Makespan, Cost, Number of Rejected Requests, Successful Jobs, User Satisfaction	Aggregation
Iturriaga et al. [31]	GA + SA	VM Requests	Multi-cloud	Reserved VM Instances	Broker Revenue	No

generated by a Poisson process with $\lambda = 15$. Its running time is lower than 30s, ensuring that the solution is ready before the round time elapses. A similar study [29] by the same authors is devoted to a multi-objective problem of minimization of energy consumption, CO_2 emission, and deployment cost. The optimization is guided by a preference vector predefined by system users, and relies on a uniform mutation operator and a variation of the two-point crossover operator. The investigated cloud federation has 8 clouds and each cloud has between 200 and 2,600 hosts. A GA reduces energy consumption by up to 29.4%, CO_2 emissions by up to 26.3% and improves profit by up to 3.6% compared to the results of a greedy heuristic.

Legillon et al. [10] propose a GA to minimize the cost of the placement of services composed of processes having no intra-communication. The GA includes problem-specific operators like scaling up and down, splitting, merging, and removing VMs. The GA is compared against the CPLEX mathematical programming solver on a set of instances with up to 10 processes, 4 resources, 4 services, and 26 VM types. The GA provides near-optimal solutions for smaller instances. For larger instances the

performance of the GA is worse than CPLEX. However, the GA is more stable than CPLEX, which crashed on some instances.

The problem can be tackled in a broader perspective by using Cloud Deployment Options (CDO). Here, the representation includes not only the VM type of a service, but also an initial VM count and a precise description of the scaling (growing/shrinking) policies. CDOXplorer [27] is based on a multi-objective NSGA-II algorithm [60]. CDOXplorer minimizes of the response times, costs, and SLA violations. The crossover uses only a set of predefined crossover points, which ensures that the resulting offspring has a valid structure. Similarly, multiple mutation types introduce variations to each part of the solution. Both IGD and hypervolume performance metrics analyses indicate that CDOXplorer outperforms simple and systematic random sampling techniques, as well as Simulated Annealing (SA), in experiments that involved up to 22 VM types from 3 different providers.

Another approach to cloud brokering is based on a fixed set of cloud resources or reserved instances leased by the broker. The number and types of available machines are fixed, which makes solving the brokering problem similar to solving that of VM placement in private clouds (discussed in the next section). The first example of a system with a fixed resources set is the study by Iturriaga et al. [31], which proposes an Evolutionary Algorithm (EA) with distributed sub-populations hybridized with an SA to maximize cloud broker revenue. Each sub-population is subject to two-point crossover and uniform mutation operators, followed by an SA-based local search. The maximum tested system size is 400 VM requests and 50 reserved instances (VMs), with a stopping criterion of 90s on a single HPC server. The EA is superior to the reference list scheduling algorithm by 18 to 43%.

CLOUDRB [30] is a framework for scheduling scientific applications in cloud. The problem is optimized using Particle Swarm Optimization (PSO) with the particle velocity modified by the best personal and global solutions found, while the reference solvers consist of an Ant Colony Optimization (ACO) and a GA optimizers. Like the study by Iturriaga et al. [31], the particles map tasks to resources in a multi-dimensional space composed of the matching resources. The system is tested on larger instances with 100-500 cloud resources and 200-1,000 user requests, and considers multiple objectives, which are aggregated. The comparisons of the makespan, cost, number of rejected requests, successful jobs, and user satisfaction are consistent, identifying the PSO as the best algorithm, and the ACO as the second best.

2) VM Placement in Private Clouds: VM placement in private clouds entails allocating VMs to a pre-defined physical infrastructure. Because of the coarse grain, large scale, and relatively long duration of deployment, the problems are solved offline, dividing the workload into batches. The most commonly used representation of solutions is similar to that used in cloud brokering; i.e. the solution vector is a mapping of VMs to machines. In contrast to the cloud brokering problem, it is commonly assumed that the resources (servers) belong to a uniform, high-throughput infrastructure (e.g. a data center). Table V summarizes the literature described below, addressing the VM placement optimization in private clouds.

TABLE V. COMPARISON OF VM PLACEMENT OPTIMIZATIONS IN PRIVATE CLOUDS.

Method	Technique	Workload Type	DAG	Resources	Objectives	Multi-Objective	Cloud Layer
Nefeli [37]	SA	VMs	yes	Servers	Makespan (Multiple Constraints)	Aggregation	IaaS
Laili et al. [21]	RCO (ref. GA)	Tasks / VMs	yes	Service / Server	Cost, Trust, Execution Speed and Efficiency	Aggregation	SaaS/IaaS
APA-VMP [36]	PSO	VMs	no	Servers	Energy, Number of Allocated Workloads	Hierarchical	IaaS
MGGA [32]	GA+FS	VMs	no	Servers	Resource Utilization, Power, Temperature	Fuzzy aggregation	SaaS/IaaS
VMPACS [33]	ACO (ref. GA)	VMs	no	Servers	Resource Utilization, Energy	Pareto	IaaS
Mateos et al. [34]	ACO	VMs	no	Servers	Makespan, Weighted Flowtime	No	SaaS/IaaS

Nefeli [37], a virtual infrastructure gateway running on top of OpenNebula [61], is a functional platform that uses an SA algorithm to solve the problem of optimal placement of VMs running workflows, with additional user guidelines. The guidelines from the users and the objectives of the provider are expressed in the form of constraints such as high availability, migration

minimization, network load balancing, and power minimization. The neighborhood function is based on the probability of changing a VM allocation. Nefeli obtains a 17% shorter execution time for media file transformations than the standard OpenNebula scheduler.

Workflow scheduling, encompassing service placement in private clouds, is also studied by Laili et al. [21], who propose the Ranking Chaos Optimization (RCO) approach. They address a twofold problem: each task is allocated on a machine and uses a service. Each position in the chosen representation has two fields: first for the service, and the second for the machine, in order to solve the twofold problem linked to task allocation. RCO is a population-based algorithm that features an elitist selection operator, which applies a chaotic change phase; i.e. the variation size is based on a logistic probability distribution. This approach is coupled with a dynamic heuristic based on local search that exploits a priori knowledge of the problem. As in Nefeli, the multiple objectives are tackled via aggregation, but the simulated results include larger instances than the Nefeli experiments on a real systems: the RCO algorithm optimizes workflows with up to 30 tasks, 50 cloud services and 100 servers, outperforming a standard GA by around 30%.

The optimization of independent tasks can be illustrated by an interesting variety of CI tools, which deal with multiple objectives in different ways. The adaptive power-aware virtual machine provisioner (APA-VMP) [36] uses Self-adaptive PSO (SAPSO) to solve the VM placement problem. The representation is based on the Smallest Position Value rule [62] to adapt the continuous PSO environment to the combinatorial problem. SAPSO particle velocity is modified by the same factors as in CLOUDRB, i.e. personal and global best. However, these are adapted using an evaporation factor, which gradually discards the past results, in order to adapt the algorithm to system dynamics. This hierarchical optimization process first searches for solutions that satisfy the performance constraints of VMs, and then looks for those with the lowest energy consumption. SAPSO is compared with standard PSO and Multi-Ensemble PSO (MEPSO) [63]. In experiments involving 100 servers and sequential batches of 20 tasks, SAPSO returns solutions with the lowest number of invalid VM allocations and the highest energy efficiency.

The last three studies presented in this section are validated against very similar instance sizes. This can give some insights on the scalability and performance of the CI tools. MGGA [32] is a grouping GA with a multi-objective fuzzy evaluation which considers the aggregated objectives. As a grouping representation MGGA uses a chromosome divided into genes corresponding to machines. Each gene value corresponds to a group of tasks allocated to the machine. A ranking crossover selects the groups with the highest fitness for its operation, while the mutation operator deletes random groups and reassigns their tasks to other groups. The objectives of the optimization are the minimization of power, wasted resources, and the temperature of a cloud system. The experiments were performed on a model profiled using a real system, and show the superior performance of MGGA over 4 bin-packing heuristics and 2 single-objective GAs. The largest instances with 1,000 servers and 2,000 VMs were initialized in less than 180s, and consecutively created a new generation in less than 20s on a standard desktop machine.

The multi-objective ant colony system algorithm (VMPACS) [33] minimizes wasted resources and energy consumption using a Pareto approach. The pheromone trails are associated with the allocation of tasks to machines; the heuristic is guided by the aggregation of the two objectives. Only the Pareto optimal solutions produced by the ants are selected to improve the learning of the ants. VMPACS is compared with MGGA [32], a single-objective ACO (SACO) algorithm [35], and a single-objective FFD algorithm [64]. The comparison was performed on an instance with 200 VMs and 200 physical hosts. VMPACS returns better quality Pareto fronts than MGGA, and its solutions dominate those returned by the single-objective algorithms, among which SACO was the best. The scalability of VMPACS was tested for instances with up to 2,000 VMs, which required less than 3 minutes to execute on a desktop machine.

Mateos et al. [34] propose an ACO for the minimization of weighted flowtime and makespan in multi-user cloud environments running parameter sweep experiments, where tasks are allocated on VMs. Despite the two objectives of the study, the optimization is guided by a single proxy variable, which corresponds to the machine load, meaning that it can be considered as a single objective optimization approach. Each task is mapped to an ant that travels between machines. The pheromone trail is built using the load table of visited hosts and the trails of the other ants encountered during the journey. At each step each ant can choose a machine randomly, or using the load table to select the least-loaded machine. The ratio of random movement decreases over time. The ACO is compared against random and best-effort policies on instances with up to 2,000 tasks and 1,050 hosts, generated on the basis of real-life experiments. It returned results 6 to 30% better than the best-effort policy.

3) Service Composition and Placement: Service placement concerns cloud users who would like to optimize the costs and performance of their services. The service placement problem answers the needs of SaaS and BPaaS users: how to optimally allocate a complex service onto an available set of (virtualized) resources. The resources could vary in nature, from physical infrastructure, through VMs, up to running software components. Services and business processes are commonly defined as workflows of tasks or activities [38], [16], [42]. The standard objectives are cost and QoS, the latter expressed in terms of service throughput [16] or latency [14], [43]. This system design optimization problem has no strict time limits and can therefore be solved offline. Table VI summarizes the literature described below, which addresses the service placement and composition.

Yusoh and Tang present a series of studies [38], [39], [40] concerning SaaS component placement. The problem includes the mapping of service components to VMs and storage systems. An original coarse-grained regrouping of related tasks (application components), called SaaS, is proposed to enhance the classical mapping of tasks and machines to a GA chromosome. The studies use a single point inter-group crossover operator, which works on SaaS instead of tasks. The mutation operator works inside groups, changing allocation of single tasks. The most advanced solutions include a Cooperative Coevolutionary GA (CCGA) [40]

TABLE VI. COMPARISON OF SERVICE PLACEMENT AND COMPOSITION OPTIMIZATIONS.

Method	Technique	Workload type	DAG	System	Resources	Objectives	Multi-Objective
Yusoh and Tang [38], [39], [40]	GA	SaaS Components	yes	Cloud	VMs, Storage Systems	Delay, Migration Cost	Aggregation
E^3[16]	GA	Abstract Services	yes	Service-Oriented Computing	Service Instances	Throughput, Latency, Cost	Pareto
SanGA [14]	GA	Services	yes	Service-Oriented Computing	Service Instances	Latency, Cost	Aggregation
Li et al. [42]	PSO (ref. GA)	Abstract Services	yes	Cloud Logistics Platform	Concrete Services	Delay, Cost, Availability, Reliability	Aggregation
GreenMonster [43]	EA + Local Search	Services	no	Multi-Cloud	Internet Data Centers	Energy, Latency	Pareto
Rahimi et al. [44]	SA	Services	yes	Tiered-Cloud	User Devices and Local/Public Clouds	Cost, Power, Delay	Aggregation

that splits populations into groups in order to optimize compute and storage allocations. CCGA outperforms Classical GA (CGA) by 25 to 35%, although its completion time is up to 8 minutes for instances with 1,400 servers, which, in the worst case, is more than 3 times longer than CGA. The results of CCGA are further refined by RGGA, which proves to be 20 to 30% better than First Fit Descent (FFD) algorithms, and was tested for the placement of up to 1,500 VMs. RGGA execution time is much longer (1-2 minutes) than the FFD (less than 1s).

E^3 [16] uses E^3 Multi-Objective GA (E^3-MOGA) and Extreme-E^3 (X-E^3) for simultaneous optimization of throughput, latency, and cost of services, which are represented as workflows. The solution representation of E^3 has a complex internal structure. A gene value represents a set of concrete services. A group of genes forms an abstract service. Abstract services are grouped at a higher level by user groups. The operators applied are one-point crossover and uniform mutation. The purpose of E^3-MOGA is to find equally distributed solutions, while X-E^3 is designed to find extreme solutions. The experiments are performed on 4 abstract services, each composed of 3 concrete services. The combination of the two E^3 algorithms is compared against NSGA-II: the combination covers 67% of the NSGA-II hyperspace, while NSGA-II covers 23% results of the combination.

A similar group of objectives is targeted by SanGA [14], which is a self-adaptive GA that solves the service composition problem. The services are expressed as workflows and the objectives of the optimization are latency and price, aggregated as a weighted sum. SanGA uses vector representation, roulette-wheel selection, top-k elitism, uniform and network-aware local search mutations, and single-point and network-aware crossovers. SanGA automatically tunes the probability of the evolutionary operators. Extensive experimentation involves comparison of the SanGA algorithm with others: random, Dijkstra, standard GA, and NetGA [41]. The experiments involved a network topology with 100,000 locations and workflows featuring between 10 and 100 tasks, where each task could have between 500 and 5,000 available service instances. SanGA has the best performance among the tested algorithms and its self-adaptivity speeds up the search, e.g. the reported optimization time is lower than 4s on a standard desktop machine.

A higher CC layer BPaaS problem is tackled by Li et al. [42], who solve a service location problem in the domain of cloud logistics with PSO. The particle encoding consists of a vector mapping tasks (abstract services) to machines (concrete services). Velocity vector updates are influenced by personal and global bests. The optimization objective is a weighted sum of time, price, availability, and reliability. The evaluation was performed on a workflow with 5 tasks, each coupled with 20 concrete services. The proposed solver finds the minimum value solution faster than a GA solver.

Energy consumption minimization is a key concept in environmental considerations, as well as for maximizing battery lifetime of mobile devices. This concept is included in the two following studies. GreenMonster [43] is a multi-objective EA using a local search that optimizes renewable energy consumption (RE), cooling energy consumption (CE) and user-to-service distance (USD). GreenMonster uses a vector-based encoding, single-point crossover and uniform mutation operators and an elitist selection strategy. The local search process tries to move each task to another machine, favoring RE over other objectives. The simulations included 9 data centers, each hosting between 8 to 200 servers (878 in total) and 16 to 400 services (1,756 in total). GreenMonster results outperform static and random allocations, with 34% higher RE, 3.2% lower CE, and 23% shorter USD compared to a random placement.

Rahimi et al. [44] propose an SA optimization for mobile applications. An application is modeled as a workflow of services that could be either executed on user devices or in a cloud (local or public). The starting solutions are initialized by a greedy heuristic, based on locality, which relies on a vector-based representation. An SA-inspired search process reassigns tasks to other machines within a threshold distance, which is associated with the process temperature. The experimental validation indicates that the algorithm returns solutions whose quality is between 54 and 84% of the optimal solutions, while the speedup obtained is between 4 to 20. The experiments were performed for instances up to 105 users, 16 local and 20 public clouds.

4) Workflow Scheduling: The workflow scheduling problem is the CC resource management problem which has been most studied in the related literature. The workflow scheduling problem in the cloud consists of allocating a set of precedence-constrained tasks, organized as a DAG, on a set of (potentially virtualized) resources. The problem is oriented towards SaaS and IaaS users. The standard objectives are to minimize makespan and cost. However, additional objectives may also include the optimization of energy consumption [52], [50], [53] or communications [47]. Compared to Service Placement, workflow scheduling is less focused on the latency issue and the cost of networking. Table VII summarizes the literature described below, concerning the service placement and composition.

TABLE VII. COMPARISON OF CLOUD WORKFLOW SCHEDULING OPTIMIZATIONS.

Method	Technique	Workload type	Resources	Objectives	Multi-Objective
Tsai et al. [46]	EA + Taguchi method	Tasks and Subtasks	Cloud Resources	Time, Cost	Pareto
Szabo and Kroeger [47]	GA (ref. PSO)	Tasks	VMs	Makespan, Cost, Communications	Pareto
CLPS-GA [53]	GA + knowledge base	Tasks	Processors	Makespan, Energy	Pareto
Mezmaz et al. [52]	EA + heuristic	Tasks	Processors	Energy, Makespan	Pareto
Pandey et al. [51]	PSO	Tasks	Compute Nodes, Storage	Cost	No
Liu et al. [48]	GA, PSO	Tasks	Machines, Data hosts	Makespan, Flowtime	Aggregation
Wu et al. [15]	GA, ACO, PSO	Tasks	VMs	Makespan, Cost	Aggregation
BOSS [45]	Auction vs. GA	Tasks	Machines	Makespan, Cost	Pareto
Guzek et al. [49], [50]	GA	Tasks	Processors	Makespan, Energy	Pareto

Tsai et al. [46] use Improved Differential EA (IDEA) combined with the Taguchi method [65] to solve the multi-objective problem of scheduling series of subtasks in a cloud environment composed of multiple resources. The solution is encoded as a permutation of subtasks. Each position of a solution consists of a subtask identifier and the allocated resource. The crossover operator exchanges only the allocation information, while the mask-based mutation operator changes only the task order. The mask is refined using the Taguchi method. The target of the method is to optimize the signal-to-noise ratio performance metric. The objectives of the optimization are execution cost and time. IDEA outperforms other EAs: DEA, NSGA-II, SPEA2, IBEA in the two tested scenarios. The largest problem size tackled included 10 tasks, each composed of 10 subtasks, and 10 resources.

A similar approach, but featuring a different representation, is used by Szabo and Kroeger [47] for the execution of scientific workflows on public clouds. They studied a problem of minimization of runtime, cost, and communication overhead. The study compared a multi-objective EA based on NSGA-II, a single-objective steady-state GA, a heuristic, and a PSO. The EA's solutions are represented by two chromosomes: a vector of task assignments and a permutation of task priorities. The operators consist of single-point crossover and uniform mutation operators for the vector part and single-point order crossover and swap mutation operators, both preserving the ordering constraints, for the permutation part. The workflows used for experiments had varying degrees of computational and IO intensity and a size of between 25 and 1,000 tasks. Only a single VM type was considered. For small workflows, the single-objective GA provided one order of magnitude better solutions, while MOEA reached an improvement of two orders of magnitude. For large instances, MOEA is 80% better than the single-objective GA.

Case Library and Pareto Solution based hybrid GA (CLPS-GA) [53] is a hybrid solver for workflows with two objectives: optimizing makespan and energy consumption. The algorithm includes multi-parent crossover operator (MPCO) and Case Library, which is used to search for a similar, previously solved solution to initialize the population. The experiments were performed on instances with up to 30 tasks and 4 processors. The experiments show that MPCO preserves the diversity of the population,

especially for large problem instances. The comparison of CLPS-GA with the adaptive, chaos, and local GAs shows that CLPS-GA has the fastest convergence and the highest stability, while it ranks second in terms of solution diversity.

The remainder of the studies use various versions of the vector representation. Mezmaz et al. [52] propose a GA to optimize the energy consumption and makespan of workflow execution. The algorithm is based on a parallel multi-start island model and is hybridized with the ECS [66] heuristic used as a local search method. The representation is a permutation of tasks, with ECS selecting the machines and their speed. The operators are problem-specific: the mutation operator is based on the b-level [67] concept, while a two-point crossover interchanges sequences of tasks after sorting them. Validation is performed on a set of instances with up to 120 tasks and 64 processors, with varying degrees of heterogeneity and differing communication costs. 83% of the solutions of the hybrid algorithm improve on the solutions of ECS; the average decrease of the energy consumption is 48%, while the average makespan is reduced by 12%.

Pandey et al. [51] solve the problem of cost minimization (including computation and communication costs) for workflow scheduling in cloud environments by means of a PSO algorithm. The particles are mapped into a number of dimensions equal to the number of tasks, each representing an allocation of tasks to machines. Particle velocity is updated using personal and global bests. The problem instances evaluated included 5 tasks and 3 distributed data centers. The PSO outperforms the greedy Best Resource Selection heuristic at least by an order of magnitude in most cases.

Liu et al. [48] solve a security-constrained workflow scheduling problem, where tasks' security demands (sd) and machines' security ranks (sr) are described on a five-level fuzzy scale. The probability of failure increases with the growth of the difference between sd and sr. A particle has the number of dimensions equal to the number of tasks, and the value in each dimension can be mapped to the machine. The feasible schedules cannot include any task allocations that exceed a predefined threshold of risk. The objective of the optimization is the aggregation of makespan and flowtime, with equal weights. Multi-start GA (MSGA), multi-start PSO (MSPSO), and variable neighborhood PSO (VNPSO) are tested on instances with up to 16 tasks, 6 machines, and 5 data hosts. VNPSO has the best performance among the tested algorithms, while its execution time is no more than 116% of the fastest MSPSO algorithm.

Wu et al. [15] propose a framework that places VMs that fulfill service constraints via a randomized package algorithm and subsequently schedules workflows on the set of VMs. A package is a sequence of tasks belonging to the same workflow instance and ordered according to the precedence constraints. The solutions are encoded as a vector of task assignments. The minimization of schedule length and cost is done by three algorithms: a GA, an ACO, and a PSO. The simulations with up to 300 tasks and 20 VMs show varying performance among the algorithms: the GA is the most stable, the ACO scales the best, while the PSO does not scale well when makespan is considered. The PSO is also the slowest (up to 58s for large instances).

An auction-based Biobjective Scheduling Strategy (BOSS) [45] algorithm solves a problem in bi-objective (cost and makespan) DAG scheduling in a multi-cloud environment. The algorithm is evaluated against NSGA-II* and SPEA2*. The encoding uses a vector of task assignments and a permutation of task priorities as a solution representation. The experimental section involves an extensive study performed using the GridSim [68] simulator with systems of up to 3,000 tasks and 1,200 machines. The solutions of the BOSS algorithm are dominated by the results of EAs in fewer than 2% of the cases.

Guzek et al. [49] use a Cellular GA (CGA) to optimize the workflow schedule in the presence of explicit communications costs on homogeneous infrastructures. A solution is represented as a vector of task assignments, which is further processed by a list heuristic based on their b-level to produce the final schedule. The proposed grouping crossover intrinsically minimizes the communications volume, and thus indirectly the makespan and the energy consumption. Diversification of the search is driven by a uniform mutation operator. The CGA was tested against a standard GA in simulations with real application structures and varying communication costs. The CGA consistently outperformed the GA, especially for the communication-intensive workflows. A further study of a heterogeneous version of the problem includes combinations of the grouping operator with three multi-objective EAs: MOCell, NSGA-II, and IBEA [50]. The representation is extended by an additional vector of processing speeds, which impact the energy consumption. The experiments show that the number of processors and the communication to computation ratio have the largest impact on the quality of the solutions.

A solution to the the hybrid resource allocation problem by Laili et al [21] was discussed in Section III-A2.

B. Dynamic Problems

1) Capacity Planning: Capacity planning problems for CC target the anticipation of the future load of CC systems, in order to match provisioned resources with user demand. The main objective is to optimize the user satisfaction (QoS) while minimizing the cost. User satisfaction is sensitive to under-provisioning (or under-subscription) of resources, i.e. a situation when the provisioned resources do not ensure the QoS expected by users. Table VIII summarizes the literature described below, which deals with service placement and composition.

The first two works presented consider IaaS layer. Ghosh et al. [19] propose an SA-based solver for cloud capacity planning. The cloud system is stochastically modeled and composed of three pools of machines: hot, warm, and cool. Two optimization objectives are considered in this study: the minimization of the total cost and of the total infrastructure cost. The SA uses a problem-specific neighborhood complemented by an initialization process that places all machines in the hot pool. The selection of a solution from the neighborhood is based on a uniform probability mass function. The acceptance of a new state is based

TABLE VIII. COMPARISON OF CLOUD CAPACITY PLANNING OPTIMIZATIONS.

Method	Technique	Workload type	Dep.	System	Resources	Objectives	Multi-Objective	Online	Cloud Layer
Ghosh et al. [19]	SA	Jobs	No	Private Cloud	VMs, PMs	Costs	Aggregation	Offline	IaaS
Rao et al. [55]	FS	Requests	No	Private Cloud	PMs	User-defined Metrics	Aggregation	Online	IaaS
Gonsalves and Itoh [54]	GA	Tasks	Yes	Petri-Net Services	Servers	Waiting and Service Costs	Aggregation	Offline	BPaaS
Nae et al. [17]	ANN	Requests	No	Multiplayer Game	Servers	Load Prediction Accuracy	No	Online	SaaS
Kousiouris et al. [18]	GA, ANN	Requests	No	Cloud	Servers	Makespan Estimation Accuracy	No	Offline	SaaS

on the Metropolis criterion [69]. The solver finds optimal solutions for instances with 20 machines. The algorithm runtime is typically below 4% of the time needed to perform an exhaustive search to minimize total cost, and less than 0.2% in minimizing total infrastructure cost.

Rao et al. [55] propose a more general and self-tuning fuzzy control system that optimizes user-defined metrics (related e.g. to performance, cost, or energy consumption) of cloud applications with multiple classes of services allocated to VMs. The controller implements a static fuzzy control logic. The scaling factor controller, together with the output amplifier, are responsible for adjusting the system to the dynamic capacity of servers. The control reference is provided by QoS profile, while an error value is an output of the QoS monitor. The approach outperforms other control techniques such as Kalman filter, adaptive proportional integral, and auto-regressive-moving-average. The scalability of the system was tested in a real testbed consisting of 16 servers and 128 VMs.

BPaaS dynamic optimization is tackled by Gonsalves and Itoh [54]. They propose a Client Server Petri net, which is further optimized by a GA to minimize the total operational cost, calculated as a sum of waiting and service costs. Consequently, the problem belongs therefore to the BPaaS optimization class. The GA uses a real-number, two-dimensional problem-specific representation which forms a table: each task is associated to two variables (service time and priority) for each phase of the process. The crossover operator randomly chooses two cells of the table, and the rectangular shape delimited by the cells forms the interchanged region. The mutation operator adds a random noise component while preserving the constraints. The decision variables of the problem consist of the number of servers allocated to a service, the priority of the services, and the service time. A sample automobile purchase system is being optimized for a timespan of 30 days in systems with 2 to 7 servers.

The two final studies presented consider the SaaS layer issues. Nae et al. [17] propose a system that predicts and manages resources capacity for massively multiplayer online games. The application requirements are described as distinct resources (CPU, memory, network input and output) and are provisioned by distributed data centers. Each data center corresponds to an aggregate of machines coupled with a set of SLA-inspired provisioning rules. The proposed ANN is a multi-layer perceptron with (6,3,1) internal structure. The goal of the system is to predict the load and to request the appropriate quantity of resources for the next time step, minimizing over- or under-subscription. The proposed ANN consistently outperforms other resource management mechanisms, such as sliding window median, moving average, or exponential smoothing. The validation via simulations of a system with up to 2,000 simultaneous users was based on a set of traces including 10,000 samples, each representing a period of 2 minutes.

Kousiouris et al. [18] propose an ANN-based framework for forecasting the load of a GNU Octave [70] system. The structure of ANNs is created by a GA. An ANN is encoded using a bit-string representation, which contains the genes representing the number of layers, transfer function, and the number of neurons per layer. The string length is constant and limited by the maximum allowed structure size. The best among the trained networks was able to forecast the load with mean and maximum absolute errors smaller than 8% and 23%, respectively. The system was validated using a set of simulations with up to 10,000 service calls, and its scalability was proven to be linear for up to 8 concurrent users.

2) Server Farm Load Balancing: The server farm load balancing problem is an on-line problem encountered by SaaS providers. The objective is to optimally dispatch incoming requests from users to the pool of available machines [56], [57]. The main objective is to fulfill the expected QoS, while additional objectives may be related to minimizing the utilization (cost, energy consumption) of the infrastructure. The solutions set out in Table IX present a wide variety of methods.

Dhinesh and Venkata [57] present honey bee behavior inspired load balancing (HBB-LB). The tasks that need to be balanced (the ones that are assigned to overloaded machines) are represented as bees, while machines (VMs) with low load levels are represented as bees' destinations. Simulated bee behavior is used in a scouting mechanism, and is exploited by the bees that follow, acting as foragers. Tasks have three possible priorities which affect the allocation decisions. HBB-LB decreases makespan and response time, while improving the load balance and reducing the number of migrated tasks. The solution was tested for a

TABLE IX. COMPARISON OF CLOUD LOAD BALANCING OPTIMIZATIONS.

Method	Technique	Workload type	Resources	Objectives	Multi-Objective
Dhinesh and Venkata [57]	Honey Bee	Tasks	VMs	Makespan	No
Sharifian et al. [56]	ANN + FS	Requests	PMs	Response Time, Throughput	No
Laredo et al. [58]	Sandpile	BoT Jobs	PMs	Makespan	No
Pinel et al. [59]	Savant	Tasks	PMs	Makespan	No

system composed of up to 7 VMs and 40 tasks.

A fuzzy control resource management solution that balances the load among servers by Rao et al. [55] was discussed in Section III-B1.

Another system proposed by Sharifian et al. [56] uses a radial basis function ANN (RBF) and an adaptive neuro-fuzzy inference system (ANFIS) to adjust the probability of selecting a server. RBF uses utilization information and the stretch factor. It has a three-layer structure with 4 hidden neurons. ANFIS uses a first order Sugeno model [71] and is composed of five layers: fuzzy, product, normalized, defuzzy, and summation. The parameters of ANFIS are derived using a combination of the least square and gradient descent methods. The implementation includes a cluster with 1 load balancer, 16 web servers, and 16 database servers. The cluster is used by up to 60,000 simulated clients. ANFIS has the best running characteristics and is the most scalable, while RBF is second, when compared to utility function, WRR [72], and CAP [73].

Laredo et al. [58] propose a sandpile-inspired algorithm for distributed load balancing that optimizes BoT workloads distribution on heterogeneous infrastructures. This nature-inspired load balancer organizes the system as a cellular automaton which triggers avalanches in case of local unbalance. The avalanches follow a power law in respect of event size, meaning that small events occur very often, while large events are very rare. The system was experimentally tested for very large scale problems with up to 2,048 machines and 2,048 BoTs, each composed of 1,000 tasks. The simulations show that the approach produces near-optimal solutions, providing results at least 5 times better in terms of makespan and throughput than a round robin, and several percent better in terms of flowtime.

Pinel et al. [59] propose another novel method using an algorithm based on the savant syndrome [74] to optimize load balancing on heterogeneous system. The algorithm structure imitates the ways of finding solutions to computationally hard problems employed by autistic people, using a distributed pattern recognition scheme. As a result, the savant algorithm is highly scalable for any problem size. However, the problem is solved in batches, contrary to the other above-mentioned load balancing algorithms. An advanced version of the savant algorithm hybridized with a local search is proposed. This solution outperformed the MinMin [75] algorithm in a simulated environment that included 16 machines and 512 tasks.

IV. DISCUSSION

A. Lessons learnt

The CI tools discussed in this survey are well suited to the diverse range of the previously presented CC problems. Their applicability to solving static or dynamic problems creates the main division between the applications of CI techniques. The metaheuristics used for combinatorial optimization, such as EAs, ACO, and PSO are more applicable to offline problems, while FS and ANN, which are paradigms commonly used to solve control problems, are applicable to online problems. As predicted by the No-Free Lunch Theorem [76], none of the metaheuristics is consistently superior to any other. This shows that careful algorithmic design must be applied in order to match the specific needs of a particular problem.

This survey has also paid attention to the scale of problems that are solved by CI. In many papers, the systems are limited to a small number of machines and tasks, raising questions about their scalability and applicability in the cloud context. Evolutionary computation scalability is limited by the need to evaluate the generated solutions. For large-scale problems, solving large instances is time consuming, due to the need for more evaluations (or generations) or more individuals by the optimization algorithm. By comparing evolutionary approaches to various static problems, we can conclude that the instance size solvable by a standard desktop machine has no more than a few thousand tasks or machines.

The scalability of dynamic resource management approaches is limited mainly by the training phase, which is most commonly performed offline using a training set. The dynamic optimizers are then validated on large sets of tasks, but the scale of tested systems is generally insufficient to consider them as solutions ready to deploy in CC. The cloud capacity on offer, as well as user needs, consistently keep growing, implying that the proposed solutions should be able to scale and address problems with at least thousands of machines and tasks.

Hybridization helps in finding solutions for large problem instances [32], [43], [52]. Another approach to addressing large-scale problems is based on parallel versions of the algorithms [52]. However the speedup should be balanced against the increased cost of finding solutions.

The dynamics and the issue of predictability are emerging challenges for the CC optimization problems and can be solved by CI tools, as presented in the sections dealing with the load balancing and workload prediction. Combining robust optimization with machine learning tools can therefore enhance the expected quality of results in CC systems.

B. Alternative methods

As most of the problems are NP-hard, there are no efficient, optimal algorithms. Exact methods such as branch-and-bound are applicable only to small instances. Alternative methods include heuristics (without specific quality guarantees), approximate algorithms, or reductions of problems to easier versions, provided that loss of detail in the model is acceptable. Quantum computing may be an applicable alternative in the future. However, the large scale of CC poses a great challenge.

The most common competitors of CI tools, are heuristics and algorithms based on mathematical models, and include linear programming [77], integer linear programming [78], non-linear programming [79], Lyapunov optimization techniques [80], stochastic integer programming [81], and sample-average approximation [82]. The mathematical models do not model the whole complexity of the problems, simplifying them to fit the assumptions of the optimization framework. Depending on the applied method, the optimal solution obtained for a simplified problem may be far from the global optimum of the real problem. These approaches create not only competition but also new opportunities. Indeed, CI may introduce an additional layer of adaptation to such solutions, providing tools for learning and the refinement of the control of cloud systems using fast-response heuristics or control systems.

C. Unexplored territories

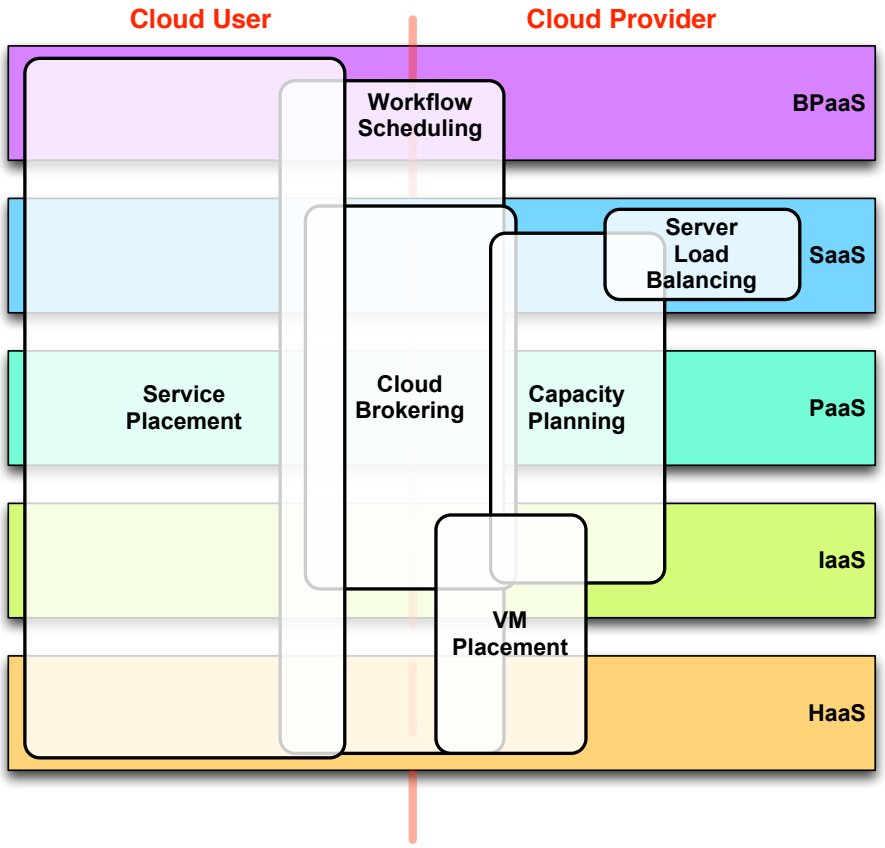

Fig. 4. Relationship of proposed, unexplored goals for CI solutions of resource management problems to CC layers.

CC is not yet a mature field, and presents many opportunities for researchers who want to tackle real-life, hard optimization problems. The first unexplored domain is many-objective optimization. As presented in Section II-D, multiple objectives exist for optimization of CC systems. These will gain in importance as CC becomes a well-established and regulated field, which will increase the focus on fulfilling auxiliary objectives such as privacy or compliance with various norms and standards.

A great majority of the presented CI tools feature problem-specific adaptations, such as specialized operators or local searches. A natural advance in the field would be to combine the various methods in order to improve algorithmic performance. This approach will be made easier by the fact that a large variety of solutions share very similar problem encodings.

Additionally, the optimization of very large scale systems (with more than 10,000 resources) has still not been tackled by the CI community, although this is a real problem faced by large cloud providers such as Amazon or Google. Additionally, the large scale of such systems blurs the distinction between online and offline solutions. In open cloud systems, with several thousands of users, even offline problems must be solved quickly and efficiently.

To underline the problems not yet addressed by CI, Fig. 4 is a proposal for the future shape of the field. The differences between the proposal and the current state of the art (Fig. 2) highlight open research problems, which are discussed in more detail below.

The research community rarely tackles PaaS problems. That could be because PaaS technology and infrastructures are most commonly owned by single companies (e.g. Microsoft, Google), reducing the transparency and generality of any potential research, while their in-house developments are trade secrets. Assuming that standardization of the field advances, this is a pressing problem, especially interesting from the cloud brokering perspective.

Despite the many multi-objective studies, there are no coherent cloud optimization frameworks that could simplify leveraging existing solutions to provide cross-layer optimization that tackles multiple objectives of various natures. Potential candidates for such frameworks are cloud management systems (e.g. OpenNebula [61], OpenStack[4]), cloud simulators (e.g. CloudSim [83], GreenCloud [84]), or open cloud research platforms (e.g. Bonfire [85]). Such a vertical consolidation could be complemented by a horizontal one, i.e. by the inclusion of multiple resource types in the system models [86] and optimization processes [10], [17]. Uniform frameworks would also help in selecting the most appropriate CI tools for a specific sub-problem of CC resource management.

The strengths of various CI tools should be combined to tackle cross-layer optimization of CC systems. The cloud resource management problems presented in this paper seldom use information from more than two layers; achieving optimization of multiple objectives requires better knowledge and understanding of the existing interdependencies. A very tempting perspective is to join the strengths of all CI paradigms for such applications. An example could be a system using ANN for workload prediction, FS for dealing with uncertainties, and multi-objective EA to search for the best solutions.

A promising direction is to use another paradigm of artificial intelligence, namely multi-agent systems, to enable cooperation between autonomous and intelligent systems [87]. Modeling each of these systems as an agent could hide the internal complexity and highlight the relationships between entities in the systems. The concurrent decision making by multiple agents could be modeled using a game-theoretical approach, which could be very effective in solving problems in complex settings with multiple interactions between various autonomous entities in CC systems.

V. Conclusions

This survey has presented a wide range of CC problems addressed by all major CI paradigms. CI proves to be applicable to multiple resource management problems that exist at all layers of CC. The advantages of CI paradigms in the field of CC are complementary: ANNs can be used to predict workloads, FSs may be used to deal with uncertainties, and EAs may be applied for search and optimization. CI and CC are in fact cross-fertilizing: CI contributes multiple tools that can deal with the dynamics, the uncertainty, the growing scale and multiple objectives of cloud systems. CC proposes challenging new problems that lead to the creation and the improvement of CI theory and practice. As CC proves to be a new major paradigm, with growing market share and research interest, and as CI becomes more widely applied and accepted in real-life (often critical) systems, everything suggests that the connections between CI and CC will expand and strengthen over time.

There is still plenty of room for improvement, exploration, and hybridization of novel approaches to the optimization in the domain of CC. Indeed the current state of the art is merely an introduction to the solution of the wide range of novel challenging CC problems. The most promising are likely to be connected with holistic and cross-layer optimization. Due to the multiplicity of involved factors, a multi-agent or game theoretic approach may be necessary to coordinate all engaged CI tools. There are other opportunities for hybridization, such as joining CI with machine learning or mathematical programming: the former for increased adaptability, the latter for accuracy and predictability of results. On the technical side, the CC landscape is continuously evolving, impacting the relevance of the existing problems and the emergence of new ones. Hybrid clouds are prominent examples, using multiple CC deployment models in a single system. Despite sketching the landscape of CC resource management problems in this survey, there is still no uniform formal framework that could serve as a precise, simple, and detailed classification of the field. In conclusion, considering all the new challenging problems, research and development in this field are still in their infancy, and the opportunities for the CI community are manifold.

Acknowledgments

This work has been partially funded by FNR and CNRS INTER/CNRS/11/03/Green@Cloud project, FNR and NCBiR INTER/IS/6466384/ISHOP project, FNR and Tri-ICT with the AFR contract no. 1315254. We would also like to thank Prof. Joanna Kołodziej, Dr. Johnatan Pecero, and Dominic Dunlop for their valuable comments.

[4]http://www.openstack.org/

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, 2011.

[2] X. Liu, Y. Zhang, B. Wang, and J. Yan, "Mona: Secure multi-owner data sharing for dynamic groups in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1182–1191, 2013.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, pp. 34–41, April 2005.

[4] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *2010 Second International Conference on Computer and Network Technology (ICCNT)*, pp. 222–226, April 2010.

[5] T. Casavant and J. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on Software Engineering*, vol. 14, no. 2, pp. 141–154, 1988.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman & Company, 1979.

[7] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Handbook on Scheduling*. Berlin: Springer-Verlag, 2007.

[8] E. L. Lawler, J. K. Lenstra, A. H. Rinnooy Kan, and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity," *Handbooks in operations research and management science*, vol. 4, pp. 445–522, 1993.

[9] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *The Annals of Mathematical Statistics*, pp. 338–354, 1953.

[10] F. Legillon, N. Melab, D. Renard, and E.-G. Talbi, "Cost minimization of service deployment in a multi-cloud environment," in *2013 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2580–2587, June 2013.

[11] T. Yang and A. Gerasoulis, "DSC: scheduling parallel tasks on an unbounded number of processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 951–967, 1994.

[12] J. Coffman, E.G., M. Garey, and D. Johnson, "Approximation algorithms for bin-packing an updated survey," in *Algorithm Design for Computer System Design* (G. Ausiello, M. Lucertini, and P. Serafini, eds.), vol. 284 of *International Centre for Mechanical Sciences*, pp. 49–106, Springer Vienna, 1984.

[13] M. Sindelar, R. Sitaraman, and P. Shenoy, "Sharing-aware algorithms for virtual machine colocation," in *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, (New York, NY, USA), pp. 367–378, 2011.

[14] A. Klein, F. Ishikawa, and S. Honiden, "SanGA: A self-adaptive network-aware approach to service composition," *IEEE Transactions on Services Computing*, vol. 7, no. 3, pp. 452–464, 2014.

[15] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems.," *Journal of Supercomputing*, vol. 63, no. 1, pp. 256–293, 2013.

[16] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E^3: A multiobjective optimization framework for SLA-aware service composition," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 358–372, 2012.

[17] V. Nae, A. Iosup, and R. Prodan, "Dynamic resource provisioning in massively multiplayer online games," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 3, pp. 380–395, 2011.

[18] G. Kousiouris, A. Menychtas, D. Kyriazis, K. Konstanteli, S. Gogouvitis, G. Katsaros, and T. Varvarigou, "Parametric design and performance analysis of a decoupled service-oriented prediction framework based on embedded numerical software," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 511–524, 2013.

[19] R. Ghosh, F. Longo, R. Xia, V. Naik, and K. Trivedi, "Stochastic model driven capacity planning for an infrastructure-as-a-service cloud," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 667–680, 2014.

[20] M. Guzek, S. Varrette, V. Plugaru, J. E. Pecero, and P. Bouvry, "A holistic model of the performance and the energy efficiency of hypervisors in a high-performance computing environment," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 15, pp. 2569–2590, 2014.

[21] Y. Laili, F. Tao, L. Zhang, Y. Cheng, Y. Luo, and B. R. Sarker, "A ranking chaos algorithm for dual scheduling of cloud service and computing resource in private cloud," *Computers in Industry*, vol. 64, no. 4, pp. 448–463, 2013.

[22] A. Tanenbaum and D. Wetherall, *Computer Networks*. Prentice Hall, 5th ed., 2011.

[23] D. Kliazovich, J. Pecero, A. Tchernykh, P. Bouvry, S. Khan, and A. Zomaya, "CA-DAG: Communication-aware directed acyclic graphs for modeling cloud computing applications," in *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*, pp. 277–284, June 2013.

[24] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[25] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, (New York, NY, USA), pp. 164–177, ACM, 2003.

[26] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the Linux Virtual Machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, pp. 225–230, jul 2007.

[27] S. Frey, F. Fittkau, and W. Hasselbring, "Search-based genetic optimization for deployment and reconfiguration of software in the cloud," in *2013 35th International Conference on Software Engineering (ICSE)*, pp. 512–521, May 2013.

[28] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based genetic algorithm for optimized assignment of VM requests on a cloud brokering environment," in *2013 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2496–2503, June 2013.

[29] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation," *Cluster Computing*, vol. 16, no. 3, pp. 451–468, 2013.

[30] T. S. Somasundaram and K. Govindarajan, "CLOUDRB: A framework for scheduling and managing high-performance computing (HPC) applications in science cloud," *Future Generation Computer Systems*, vol. 34, pp. 47–65, 2014.

[31] S. Iturriaga, S. Nesmachnow, B. Dorronsoro, E.-G. Talbi, and P. Bouvry, "A parallel hybrid evolutionary algorithm for the optimization of broker virtual machines subletting in cloud systems," in *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pp. 594–599, Oct 2013.

[32] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications (GreenCom)*, pp. 179–188, Dec 2010.

[33] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[34] C. Mateos, E. Pacini, and C. G. Garino, "An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments," *Advances in Engineering Software*, vol. 56, pp. 38–50, 2013.

[35] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *2011 12th IEEE/ACM International Conference on Grid Computing (GRID)*, pp. 26–33, Sept 2011.

[36] R. Jeyarani, N. Nagaveni, and R. V. Ram, "Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 811–821, 2012.

[37] K. Tsakalozos, M. Roussopoulos, and A. Delis, "Hint-based execution of workloads in clouds with Nefeli," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1331–1340, 2013.

[38] Z. Yusoh and M. Tang, "A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, July 2010.

[39] Z. Yusoh and M. Tang, "A penalty-based grouping genetic algorithm for multiple composite saas components clustering in cloud," in *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1396–1401, Oct 2012.

[40] Z. Yusoh and M. Tang, "Composite SaaS placement and resource optimization in cloud computing using evolutionary algorithms," in *2012 IEEE 5th International Conference on Cloud Computing (CLOUD)*, pp. 590–597, June 2012.

[41] A. Klein, F. Ishikawa, and S. Honiden, "Towards network-aware service composition in the cloud," in *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, (New York, NY, USA), pp. 959–968, ACM, 2012.

[42] W. Li, Y. Zhong, X. Wang, and Y. Cao, "Resource virtualization and service selection in cloud logistics," *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1696–1704, 2013.

[43] D. H. Phan, J. Suzuki, R. Carroll, S. Balasubramaniam, W. Donnelly, and D. Botvich, "Evolutionary multiobjective optimization for green clouds," in *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, (New York, NY, USA), pp. 19–26, ACM, 2012.

[44] M. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. Vasilakos, "MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," in *2012 IEEE Fifth International Conference on Utility and Cloud Computing (UCC)*, pp. 83–90, Nov 2012.

[45] H. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1203–1212, 2013.

[46] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Computers & Operations Research*, vol. 40, no. 12, pp. 3045–3055, 2013.

[47] C. Szabo and T. Kroeger, "Evolving multi-objective strategies for task allocation of scientific workflows on public clouds," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, June 2012.

[48] H. Liu, A. Abraham, V. Snel, and S. McLoone, "Swarm scheduling approaches for work-flow applications with security constraints in distributed data-intensive computing environments," *Information Sciences*, vol. 192, pp. 228–243, 2012.

[49] M. Guzek, J. Pecero, B. Dorronsoro, P. Bouvry, and S. Khan, "A cellular genetic algorithm for scheduling applications and energy-aware communication optimization," in *2010 International Conference on High Performance Computing and Simulation (HPCS)*, pp. 241–248, June 2010.

[50] M. Guzek, J. E. Pecero, B. Dorronsoro, and P. Bouvry, "Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems," *Applied Soft Computing*, vol. 24, pp. 432–446, 2014.

[51] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pp. 400–407, April 2010.

[52] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E.-G. Talbi, A. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1497–1508, 2011.

[53] F. Tao, Y. Feng, L. Zhang, and T. Liao, "CLPS-GA: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling," *Applied Soft Computing*, vol. 19, pp. 264–279, 2014.

[54] T. Gonsalves and K. Itoh, "GA optimization of petri net-modeled concurrent service systems," *Applied Soft Computing*, vol. 11, no. 5, pp. 3929–3937, 2011.

[55] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "QoS guarantees and service differentiation for dynamic cloud applications," *IEEE Transactions on Network and Service Management*, vol. 10, no. 1, pp. 43–55, 2013.

[56] S. Sharifian, S. A. Motamedi, and M. K. Akbari, "A predictive and probabilistic load-balancing algorithm for cluster-based web servers," *Applied Soft Computing*, vol. 11, no. 1, pp. 970–981, 2011.

[57] L. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.

[58] J. Laredo, P. Bouvry, F. Guinand, B. Dorronsoro, and C. Fernandes, "The sandpile scheduler," *Cluster Computing*, vol. 17, no. 2, pp. 191–204, 2014.

[59] F. Pinel, P. Bouvry, B. Dorronsoro, and S. Khan, "Savant: Automatic parallelization of a scheduling heuristic with machine learning," in *2013 World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 52–57, Aug 2013.

[60] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[61] R. Moreno-Vozmediano, R. Montero, and I. Llorente, "IaaS cloud architecture: From virtualized datacenters to federated cloud infrastructures," *Computer*, vol. 45, no. 12, pp. 65–72, 2012.

[62] M. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," in *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, vol. 2, pp. 1412–1419, 2004.

[63] W. Du and B. Li, "Multi-strategy ensemble particle swarm optimization for dynamic optimization," *Information Sciences*, vol. 178, no. 15, pp. 3096–3109, 2008. Nature Inspired Problem-Solving.

[64] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Int. CMG Conference*, pp. 399–406, 2007.

[65] G. Taguchi, S. Chowdhury, and S. Taguchi, *Robust engineering*. McGraw-Hill, 2000.

[66] Y. C. Lee and A. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009. CCGRID '09*, pp. 92–99, May 2009.

[67] I. Ahmad, Y.-K. Kwok, and M.-Y. Wu, "Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors," in *Second Int. Symposium on Parallel Architectures, Algorithms, and Networks*, (Beijing), pp. 207–213, 1996.

[68] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency And Computation: Practice And Experience*, vol. 14, no. 13, pp. 1175–1220, 2002.

[69] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[70] J. W. Eaton, D. Bateman, and S. Hauberg, *GNU Octave version 3.0.1 manual: a high-level interactive language for numerical computations*. CreateSpace Independent Publishing Platform, 2009.

[71] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[72] T. Schroeder, S. Goddard, and B. Ramamurthy, "Scalable web server clustering technologies," *IEEE Network*, vol. 14, no. 3, pp. 38–45, 2000.

[73] E. Casalicchio, V. Cardellini, and M. Colajanni, "Content-aware dispatching algorithms for cluster-based web servers," *Cluster Computing*, vol. 5, no. 1, pp. 65–74, 2002.

[74] L. Mottron, K. Lemmens, L. Gagnon, and X. Seron, "Non-algorithmic access to calendar information in a calendar calculator with autism," *Journal of autism and developmental disorders*, vol. 36, no. 2, pp. 239–247, 2006.

[75] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Journal of the ACM (JACM)*, vol. 24, no. 2, pp. 280–289, 1977.

[76] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[77] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, 2010.

[78] J. L. Berral, R. Gavalda, and J. Torres, "Adaptive scheduling on power-aware managed data-centers using machine learning," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, GRID '11, (Washington, DC, USA), pp. 66–73, IEEE Computer Society, 2011.

[79] D. Ardagna, S. Casolari, M. Colajanni, and B. Panicucci, "Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems," *Journal of Parallel and Distributed Computing*, vol. 72, no. 6, pp. 796–808, 2012.

[80] Y. Guo, Y. Gong, Y. Fang, P. Khargonekar, and X. Geng, "Energy and network aware workload management for sustainable data centers with thermal storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2030–2042, 2014.

[81] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *IEEE Asia-PacificServices Computing Conference, 2009, APSCC 2009*, pp. 103–110, Dec 2009.

[82] S. Chaisiri, R. Kaewpuang, B.-S. Lee, and D. Niyato, "Cost minimization for provisioning virtual servers in amazon elastic compute cloud," in *2011 IEEE 19th International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 85–95, July 2011.

[83] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[84] D. Kliazovich, P. Bouvry, and U. Khan, Samee, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[85] J. Jofre, C. Velayos, G. Landi, M. Giertych, A. C. Hume, G. Francis, and A. Vico Oton, "Federation of the bonfire multi-cloud infrastructure with networking facilities," *Computer Networks*, vol. 61, pp. 184–196, Mar. 2014.

[86] M. Guzek, D. Kliazovich, and P. Bouvry, "A holistic model for resource representation in virtualized cloud computing data centers," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, pp. 590–598, Dec 2013.

[87] M. Guzek, G. Danoy, and P. Bouvry, "ParaMoise: Increasing capabilities of parallel execution and reorganization in an organizational model," in *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS'13*, pp. 1029–1036, May 2013.