Chapter 1

Introduction

1.1 Problem Statement

Data centers are large-scale computing infrastructures which consume huge amount of energy each year: a typical data center consumes as much energy as 25,000 households [19]. Thus, reducing the energy consumption becomes the major concern of Cloud providers. In addition, data centers and computation powers support the modern Cloud computing industry, software industry and etc. Therefore, reducing the cost of data centers will lead to a reduction of cost of softwares which consequently be beneficial to most people who access the Internet on a daily basis. Among several components that consume energy such as cooling system, physical machines (PMs) (e.g servers), and network devices, PMs accounts for 40% and have a huge improvement space, since they are always in low utilization (e.g on average, from 10% to 50% of required resources) [5, 62]. This low utilization of resource problem can be solved by fine granularity management of Cloud resources (e.g CPUs and RAMs) using a new virtuzliation technology: containers [25,32,65] and a new service model: Container as a Service (CaaS) [56]. CaaS is a mixture of traditional IaaS (Infrastructure as a Service) [48] and PaaS (Platform as a Service); it utilizes both containers and virtual machines (VMs) as the fundamental resource management units. In CaaS, applications that were used to deployed in VMs (e.g in IaaS) are now deployed in containers. Container is an operating system (OS) level of virtualization; multiple containers can run on a VM and share OS. Therefore, server consolidation [76] can be applied in a joint of containers and VMs environment to achieve better energy reduction.

Server consolidation is an important stategy in improving the utilization throughout the Cloud resource management processes as shown in Figure 1.1 including new application allocation [35], periodic optimization [49], overloading and under-loading adjustments [49].

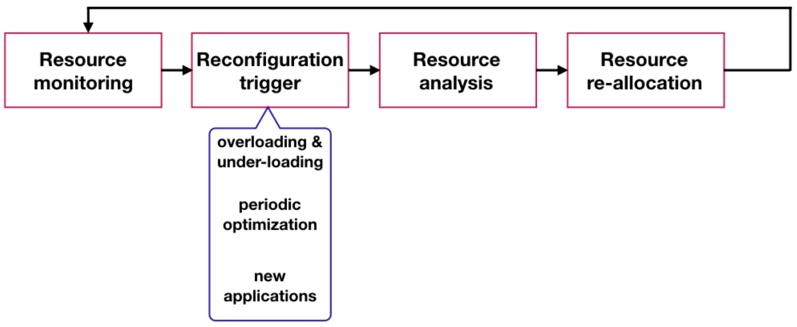


Figure 1.1: A workflow of resource management [49]

According to the characteristic of each process, server consolidation can be roughly classified into two categories: static problems [49] and dynamic problems [8]. Accordingly, server consolidation approaches also have corresponding categories. Static approaches use historical average resource utilization data as input to map applications to PMs. Static consolidation normally involves large amount of applications and PMs, therefore, the optimization is quite time-consuming and often conducted in a off-line fashion. Periodic optimization belongs to this category. It takes a number of existing applications and re-allocate them into a number of PMs. Dynamic approaches take one application each time, allocates it into one of the PMs. The operation is conducted in an online fashion, therefore, it requires fast reaction. Overloading and under-loading can be categories to dynamic consolidation problem [9]. New application allocation can be seen as either static: allocate a batch of new applications, or dynamic: allocate a new application each time. In this proposal, we consider it as a static problem.

Examples of static server consolidation are shown in the following in VM-based and container-based Cloud. In traditional VM-based Cloud, Server consolidation can be described as, given a number of Physical Machines (PMs) which can be represented as the resources(e.g CPU cores and RAM); a number of requests for fixed configurations of VMs (assume applications have been deployed in VMs), each configuration can also be represented as aforementioned resources; The objective is to allocate these requested VMs into a minimum number of PMs. The decision variable is the location of each requested VM. In container-based Cloud, instead of allocating requested VMs in PMs, a set of containers (assume applications have been deployed into containers) represented as resources is first allocated to a number of fixed type VMs, then, these VMs are allocated to PMs. The decision variables are the allocation of containers (upper level) and VMs (lower level). For the upper level of allocation, the objective is to maximize the utilization of resources (e.g a balanced utilization among several resrouces), while the lower level's objective is to minimize the number of PMs.

Traditional VM-based server consolidation are modeled as bin-packing problems [46]. This is because VMs and PMs are naturally modeled as items and bins and server consolidation and bin-packing have the same optimization objective: minimize the number of bins/PMs. The complexity of bin-packing problem is NP-hard which means it is extreme time-consuming to find its optimal solution when the number of decision variables are large. Container-based server consolidation can be categorized as a bilevel optimization problem [18]. Bilevel problems are typically non-convex and strongly NP-hard [78]. In this case, two levels are lower-level: Containers to VM and upper-level: VMs to PMs. These two levels optimization are connected through decision variables. In this case, two levels of optimization are both bin packing problems and they are cooperating [43].

Currently, most research focus on VM-based server consolidation and these methods can not be directly applied on container-based consolidation because of the different structure. Only few research focus on container-based server consolidation problem. One of the state-of-the-art research is from Piraghaj and et al [56]. They first propose a VM-resizing technique that defines the types of VM based on analyzing the historical data from Google. Then they propose a two-step allocation: first allocate containers to VMs and then allocate VMs to PMs. Their major contribution is the method of defining types of VM. The allocation of containers does not optimize the energy consumption and the allocation of VMs are traditional First Fit algorithm. In addition, they propose a dynamic consolidation [54] using a series simple heuristics such as Random Host Selection Algorithm or First Fit Host Selection. Their resource allocation system completely relies on dynamic consolidation without using static methods. Although their system can execute allocation fast, the energy efficiency cannot be guaranteed. The reasons are mainly from two aspects, firstly, they mainly rely on simple bin-packing algorithms to allocate containers to VMs. As Mann's research [46] showed, server consolidation is a lot more harder than bin-packing problem because of the multi-dimensional of resources, many constraints. Therefore, general bin-packing algorithms do not perform well. Secondly, they use a two-step allocation. Because of the interaction of two allocations, separated optimization approach will lead to local optima [47]. Therefore, these two allocations should be considered simultaneously.

The overall goal of this thesis is to develop new container-based server consolidation approaches to solve three problems: joint allocation of containers and VMs, periodic global optimization and dynamic consolidation.

1.2 Motivation

In this thesis, we aim at providing a series of approaches to continuously optimize the joint allocation of VMs and containers. A continuous optimization procedure mainly involves with three types of server consolidation: initialization, global consolidation, and dynamic consolidation. Different stages have distinctive goals, therefore, they are considered as separated research questions. In addition, a scalability problem of static optimization is considered as an optional objective.

1. Joint allocation of containers and VMs (new applications initialization),
   In this research, we take Joint allocation as a static problem which is fundamental for server consolidation problem. At this stage, a set of containers is allocated to a set of VMs and these VMs are allocated to a set of PMs. This task is challenging because the problem is a bilevel optimization where each level is a bin packing problem. Exhaustive search of entire solution space is practically impossible, for the number of possible permutation of solution is huge. Current approaches [33, 54] use simple heuristics such as First Fit to solve the problem. These greedy-based heuristics do not consider the complex structure of the problem, therefore, often reach a local optimal solution.

2. Global consolidation,
   A Global consolidation is conducted to improve the global energy efficiency in a periodical fashion. Data center constantly receives new allocations, releasing of old resources. These changing degrades the compact structure of a data center. Therefore, the data center needs a global optimization to improve the overall energy efficiency.

   The challenges are three folds, firstly, similar with initialization problem, the problem has two level of allocations and they interact with each other. Secondly, like VM-based consolidation, Container-based consolidation is considered as a multi-objective problem with minimization of migration cost as well as keeping a good energy efficiency. In bilevel optimization, multi-objective can be defined in either or both level, therefore, it further increases the complexity. Thirdly, consolidation is a time-dependent process which means the previous solution affects the current decision. Previous VM-based research only consider each consolidation as an independent process. As a consequence, although in one consolidation, the migration is minimized, It may lead to more migrations in the future consolidation. We will consider the robustness of consolidation and propose a novel time-aware server consolidation which takes the previous immediate consolidation and the future consolidation into consideration.

3. Dynamic consolidation,
   It takes one container and allocates it to VMs. Since the size of container can be dynamically adjusted, when the an application is under-provision or over-provision, the

original container is halted, resized and re-allocated. Hence, there is a need to allocate this new container in real time.

To solve a dynamic consolidation, heuristics and dispatching rules are often used [7, 27, 60, 63]. In this scenario, a dispatching rule is considered as a function that determines the priorities of VMs that a container can be placed. However, dynamic placement is much complex than bin-packing problem [46]. Because of its dynamic nature, human designed heuristics are ill-equipped in approximating solutions when the environment has changed [67].

Hyper-heuristic methods, sepcifically, Genetic Programming (GP) technique [3] can learn from the best previous allocation and automatically evolves dispatching rules to solve this problem. GP has been applied in generating dispatching rules for bin-packing problem [13,67] and other scheduling problems [51]. The results have shown promising results.

There are mainly two challenges, first, it is difficult to identify the related factors that construct the heuristic. Factors or features are the building blocks of heuristics. It is a difficult task because the relationship between a good heuristic and features are not obvious. Second, representations provide different patterns to construct dispatching rules. It is also unclear what representation is the most suitable for the consolidation problem.

4. Large-scale of static server consolidation problem,
   In this case, initialization and global consolidation are belonged to this category. Since Cloud data center typically has hundreds of thousands PMs and more, static server consolidation is always very challenging. Many approaches have been proposed in the literature to resolve the problem. There are mainly two ways, both relied on distributed methods, hierarchical-based [36, 50] and agent-based management systems [85]. The major problem in agent-based systems is that agents rely on heavy communication to maintain a high-level utilization. Therefore, it causes heavy load in the networking. Hierarchical-based approaches are the predominate methods. In essence, these approaches are centralized methods where all the states of PMs within its region are collected and analyzed. The major disadvantage of hierarchical-based approaches is that it only provides local solutions. In fact, it is infeasible and unnecessary to check all the states of PMs since the search space is too large and most PMs do not need a change. This idea motivates a way to improving the effectiveness is to reduce the number of variables so that the search space is narrowed. In this thesis, we are going to investigate the way to eliminate the redundant information.

1.3 Research Goals

1.3.1 Objective One: Develop EC-based approaches for the single objective joint allocation of containers and VMs

Currently, most research focus on VM-based server consolidation technique. They often modeled this problem as a vector bin-packing problem [88]. Container adds an extra layer of abstraction on top of VM. The placement problem has become a two-step procedure, in the first step, containers are packed into VMs and then VMs are consolidated into physical machines. These two steps are inter-related to each other. Previous research [56] solve this problem in separated steps where the first step allocate containers to VMs and the second step allocate VMs to PMs with simple bin-packing heuristics which leads to local optima. We will consider these two allocation simultaneously to reach a near-optima solution.

1. First, our first sub objective is to propose a descriptive single objective model for the bilevel optimization problem of joint allocation of container and VM. The reason to establish this model is because current server consolidation models are mostly VM-based, they cannot be directly applied on bilevel problems. Therefore, variables, constraints and objective functions need to be clarified before applying any optimization algorithm. Each level of the problem will be formulated to a multi-dimensional vector bin packing problem. It is still unclear that which objective function is the best to capture the relationship between container and VM so that the overall energy is low. We will investigate several resource wastage models [26, 28, 83] and select a suitable one. In addition, several models have to be considered, including energy model [19], price model [1], and workload model [45]. In addition, we will start from the simplest case - single dimension of resource - to more general multi-dimensional resources model.

2. Second, we will first develop a baseline approach that solve the problem using nested Evolutionary algorithms [64]. We will start from the simplest form: one dimensional bin-packing in each level to more complex multi-dimensional bin-packing.

   Nested methods have been used in solving bilevel problem for years, they are reported as effective approaches. We will investigate several approaches such as Nested Particle Swarm Optimization [44], Differential evolution (DE) based approach [2, 91] and Co-evolutioanry approach [43]. In order to adapt our problem to these existing approaches, we will develop suitable representations and genetic operators.

3. Third, although nested approaches have been reported effective, they are often very time consuming. Therefore, our third sub-objective will focus on developing more efficient algorithms. There are several possible directions to be explored such as metamodeling-based methods [80] and single-level reduction.

1.3.2 Objective Two: Develop EC-based approaches for the multi-objective joint allocation problem

As previous section (see 1.2) mentioned, the task is multi-objective since the number of VM migration has to be minimized while keep the overall energy low. In addition, periodic optimization is a time-dependent problem which means the optimal consolidation in previous operation might lead to more migrations in the current consolidation. The robustness of a data center is particularly important. The robustness measures the stableness of result of consolidation. Furthermore, we will investigate proactive approaches - considering future allocation.

1. First, we will develop EC-based approaches to solve the multi-objective joint allocation problem. In this problem, multiple objectives may involve at both of the levels. We will start from a simple case considering multi-objective in lower level: Minimizing VM migration and energy consumption. Currently, there are few studies using EC methods [21, 86] for multiobjective bilevel optimization. We will investigate which one is more suitable for this binary problem. Furthermore, like the case in single objective problem, we need to develop new representations, genetic operators to apply the algorithms.

2. Second, we will design a robustness measure. Previous studies only use simple measurement which counts the migration number between two static consolidation. This measurement aims at minimizing the number of migration between two static placement processes. It may cause more migration in the next consolidation. Therefore,

it needs a time-aware measure of the robustness of system. Therefore, in this objective, the first sub-problem we are going to solve is to propose a robustness measure. Currently, only a few research propose robustness aware server consolidation techniques [30, 70] have been proposed. They are either static threshold or probability-based threshold to measure the robustness of PMs. We will investigate an adaptive measure based on the historical data and current status.

3. Third, we will design a proactive server consolidation approach. Based on a prediction of future server consolidation and the robustness measure,we will first design an approach which maximize the robustness and also minimize the current energy consumption. Proactive consolidation [24, 72] has been studied extensively. Their experience in analyzing the workload patterns can be useful in designing new algorithms.

1.3.3 Objective Three: Develop a hyper-heuristic Genetic Programming (GP) approach for automatically generating dispatching rules for dynamic consolidation

Previously, dynamic consolidation methods,including both VM-based and container-based, are mostly based on bin-packing algorithm such as First Fit Descending and human designed heuristics. As Mann's research [46] shown, server consolidation is more harder than bin-packing problem because of multi-dimensional of resources and many constraints. Therefore, general bin-packing algorithms do not perform well with many constraints and specific designed heuristics only perform well in very narrow scope. Genetic programming has been used in automatically generating dispatching rules in many areas such as job shop scheduling [51]. GP also has been successfully applied in bin-packing problems [13]. Therefore, we will investigate GP approaches for solving the dynamic consolidation problem. We will start from considering one-level of problem: migrate one VM each time to a PM.

1. First, we will investigate which features and attributes are important when dealing with energy efficiency problem. As the basic component of a dispatching rule, primitive set contains the states of environment including: status of VMs (e.g. utilization, wastage), features of workloads (e.g. resource consumption). Although there is no research has investigate how to use them to construct dispatching rules, there are extensive statistical analysis on workload [77]. The effectiveness of functional set and primitive set will be tested by applying the constructed dispatching rules on dynamic consolidation problem.

2. Develop GP-based methods for evolving dispatching rules. This sub-objective explores suitable representations for GP to construct useful dispatching rules. It also proposes new genetic operators as well as search mechanisms.

1.3.4 Objective Four (Optional) Large-scale Static Consolidation Problem

Propose a preprocessing method to eliminate redundant variables Current static consolidation takes all servers into consider which will lead to a scalability problem. In this objective, we will investigate two branches of methods, first one categorizes a number of containers into fewer groups so that the granularity decreases [56]. Second method categorizes PMs so that only a small number of PMs are considered. This approach will dramatically reduce the search space. The potential approaches that can be applied in this task are various clustering methods.

1.4 Published Papers

During the initial stage of this research, some investigation was carried out on the model of container-based server consolidation [71].

1. Tan, B., Ma, H., Mei, Y. and Zhang, M., "A NSGA-II-based Approach for Web Service Resource Allocation On Cloud". *Proceedings of 2017 IEEE Congress on Evolutioanry Computation (CEC2017).* Donostia, Spain. 5-8 June, 2017.pp.2574-2581

1.5 Organisation of Proposal

The remainder of the proposal is organised as follows: Chapter **??** provides a fundamental definition of the Container-based server consolidation problem and performs a literature review covering a range of works in this field; Chapter **??** discusses the preliminary work carried out to explore the techniques and EC-based techniques for the initialization problem; Chapter **??** presents a plan detailing this projects intended contributions, a project timeline, and a thesis outline.

Chapter 2

Literature Review

This chapter begins by providing a fundamental background to the field of Cloud computing in Section 2.1. Section **??** discusses the resource allocation in PaaS as well as the bi-level optimization including both biologically and non-biologically inspired approaches; Section **??** delves into approaches that optimize the static placement. In addition, time-series aware problems and their approaches will be discussed; Section **??** discusses the issue of dynamic server consolidation and genetic programming; Section **??** covers approaches of scalability; Finally, Section **??** presents a summary of the important points identified in this review, alongside a discussion of the limitations of existing approaches.

2.1 Background

2.1.1 Cloud computing

Cloud computing is a computing model offers a network of servers to their clients in a on-demand fashion. From NIST's definition [48], *"cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

To illustrate how it works, considering the case: a Cloud provider builds a data center which contains thousands of servers connected with network. These servers are virtualized which means they are partitioned into smaller units of resources called *Virtual Machines (VMs)* or *Containers* [25]. A web-based application provider can access and deploy their applications (e.g Endnote, Google Drive and etc.) in these resource units from anywhere in the world. Once the applications start serving, application users can use them without installing on their local computers.

Cloud computing involves three stakeholders (see Figure 2.1): Cloud providers, Cloud users (applications providers), and End (application) users [35]. Cloud providers build data centers, provide maintenance and resource management on hardware infrastructures such as servers. Cloud users develop and deploy applications on Cloud infrastructures. End users consumes applications developed by Cloud users and hosted by Cloud providers.

The detailed goal and objectives of stakeholders are described below.

- *Cloud providers'* goal is to increase the profit by boosting the income and reducing the expense. Their income comes from Cloud users' rental of servers or *Physical Machines (PMs)* in terms of resource quality (e.g 3.5GHz dual-core CPU), quantity (e.g 3 PMs), and time (e.g 1 year). Therefore, Cloud providers objective is to maximize utilization of computing resources. A high utilization brings two benefits, firstly, it increases

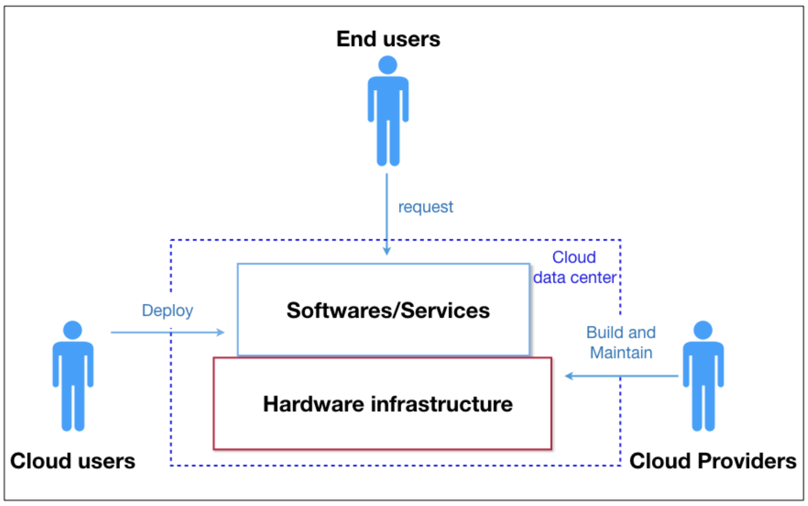


Figure 2.1: Stakeholders of Cloud computing

income by accommodating more applications in limited resources. Secondly, it cuts the expense of energy consumption by packing applications in a minimum number of PMs so that idle PMs can be turned off.

- *Cloud users*' goal is also to increase the profit mainly through two objectives, attracting more End users and reduce the expense of resources. The first objective can be achieved by improving the quality of service as well as lower the fee for End users. Either way depends not only on the software entities but also the quality of service (QoS) offered by Cloud provider. The second objective can be achieved by a good estimation of the reserved resources, so that they do not rent insufficient or too much resources which cause performance degradation or wastage.

- *End Users*' goal is to obtain a satisfactory service. It is achieved by signing a Service Level Agreement (SLA) with Cloud users which constrains the performance of the services.

Cloud computing has three traditional service models [48]: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The relationship among three service models is showed in Figure 2.2. Service models of Cloud computing are critical in solving energy consumption problem because their distinct ways of managing resources have sever effect on the problem. These distinct ways of resource management mainly result from the responsibilities among stakeholders.

- IaaS, a Cloud provider hosts hardwares such as PMs and cooling infrastructure on behalf of Cloud users. Computational resources are often encapsulated in virtualized computing units called virtual machines (VMs). Cloud providers establish a number of types of VM for simplifying the management. The 'type' means a VM contains a certain quantity of resources such as 2-cores and 1 GB RAM. *Traditional* IaaS and PaaS use VM as the fundamental unit of resources.

  A typical procedure of an application deployment includes several steps. Initially, the Cloud user estimates the resources that their applications might consume and selects a type of VM which can satisfy the requirement. After the Cloud user has made the decision, he/she sends requests to Cloud providers for a number of VMs. Finally, Cloud providers received the request, provisioned and allocated these VMs to PMs.

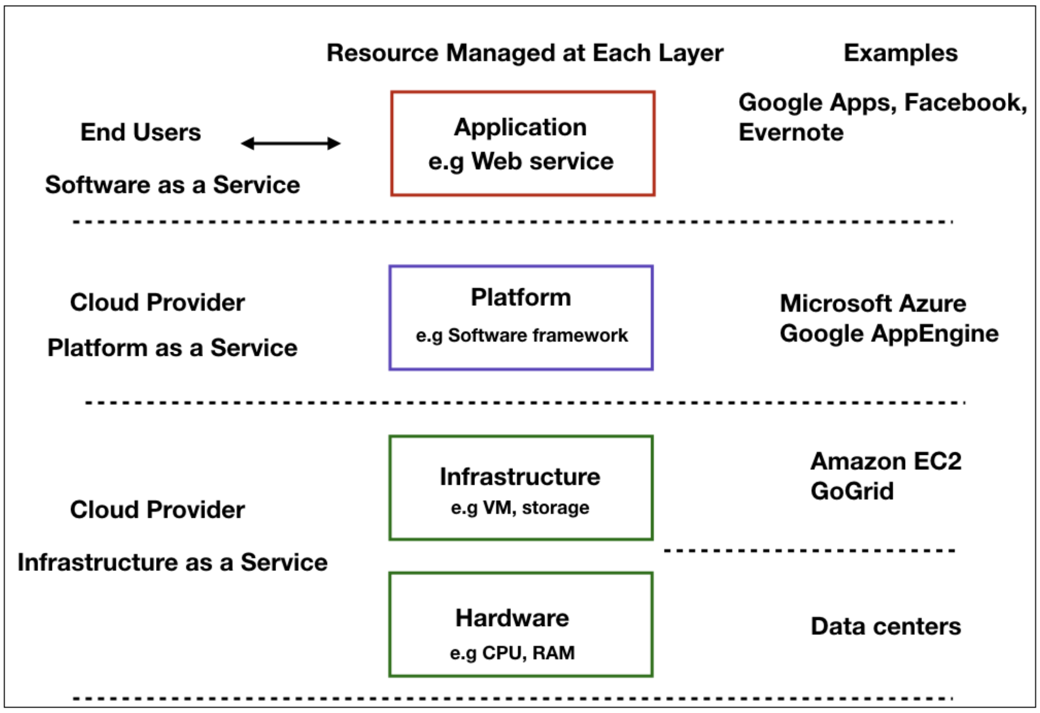


Figure 2.2: Cloud computing architecture [89]. IaaS provides the fundamental resources such as CPU cores, RAM. The resources are usually wrapped with various types of virtual machine. PaaS provides a software platform sitting on top of IaaS for Cloud users to develop applications. SaaS describes the relationship between applications and End Users.

- PaaS, a Cloud provider offers a platform which allows Cloud users to develop, test and deploy their applications on it.

  From resource management perspective, PaaS is sitting above the IaaS which means the underlying resource is still based on IaaS VM types. Different from IaaS, PaaS takes the responsibility of selecting VMs and allows Cloud users to focus on software development.

- SaaS, Cloud users develop applications and deploy them on Cloud so that End users can access them via the Internet. Although this service model does not directly related to the resource management, it provides the fundamental reasons for resource management and optimization: applications receive fluctuated requests from End users. Because of the dynamic nature of workloads, the underlying resources must also be dynamic adjusted to meet the requirement.

Overall, Cloud computing has five chacteristics:

1. On-demand self-service: A Cloud user can require computing resources (e.g CPU time, storage, software use) without the interaction with Cloud provider.

2. Broad network access: Computing resources are connected and delivered over the network.

3. Resource pool: a Cloud provider has a "pool" of resources which are normally virtualized servers. In IaaS, it provides predefined sizes of VMs. In PaaS, the resources are 'invisible' to Cloud users who have no knowledge or ability to control.

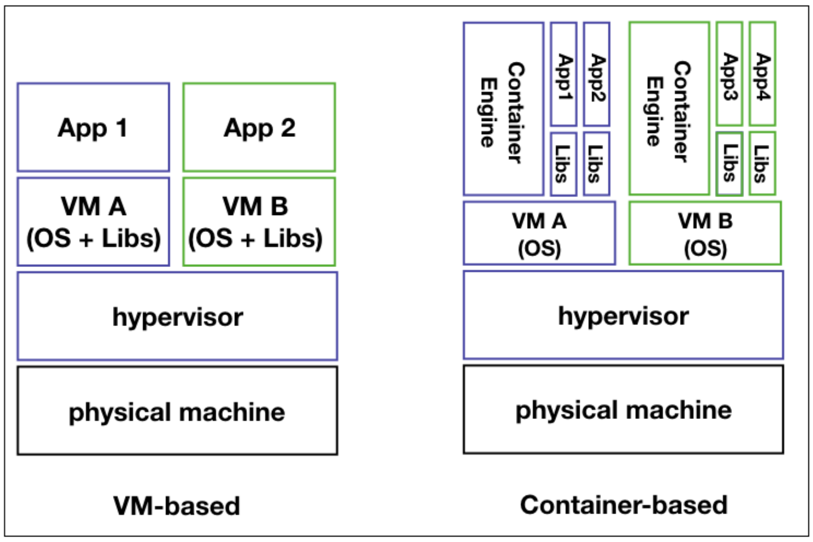


Figure 2.3: A comparison between VM-based and Container-based virtualization [54]

4. Rapid elasticity: From the perspective of Cloud users, computing resources are assigned and released in real time. In addition, the resources assign to their software is “infinite”. Therefore, Cloud users do not need to worried about the scalability of their applications.

5. Measured Service: Cloud provides an accurate measure of the usage of computing resources. It is fundamental to the pay-as-you-go policy.

2.1.2 Virtualization

Virtualization [74] is the fundamental technology that enables Cloud computing. It partitions a physical machine’s resources (e.g. CPU, memory and disk) into several independent units called virtual machines (VMs) or containers. This technology rooted back in the 1960s’ and was originally invented to enable isolated software testing, because each virtualized unit can provide good isolation which means multiple applications can run in separated VMs within the same PM without interfering each other [66]. Soon, people realized that it can be a way to improve the utilization of hardware resources: With each application deployed in a VM, a PM can run multiple applications.

There are two classes of virtualization (see Figure 2.3): Hypervisor-based or VM-based and container-based virtualization.

Virtual machine

A virtualized system includes a new layer of software - the hypervisor or the vrirtual machine monitor (VMM). The VMM arbitrates accesses to the PM’s resources so that guests’ OS can share them [75]. In the previous decades VM-based hypervisors such as Xen [4], KVM [40], and VMware ESX [79] dominate this field.

Container

Container-based virtualization is also often addressed as operating-system-level virtualization. It includes two types of container: OS container and application container [55]. OS container (as shown in Figure 2.4) can run in both PM and VM. Each container provides

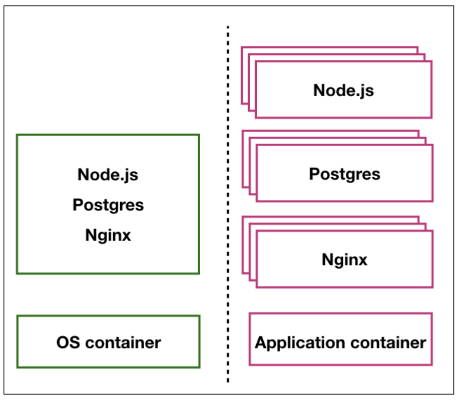


Figure 2.4: A comparison between OS container and Application container [55]

an isolated environment. There are mainly three implementations of OS-level of containers: OpenVZ, Google's control groups, and namespace [59]. Google and Facebook have been using OS container for years and being beneficial for its lightweight and fast communication among applications.

In contrast of OS containers, an application container, such as Docker and Rocket, runs a single process. It allows to separate an applications into many components. With application container, it is easy to achieve auto-scaling on a single process. In comparison between OS and application container, the former can be seen as a VM runs multiple applications where each application can has its separated environment (e.g. libraries), while an application container act like a single unit in OS container and it can be allocated in both OS container and VM.

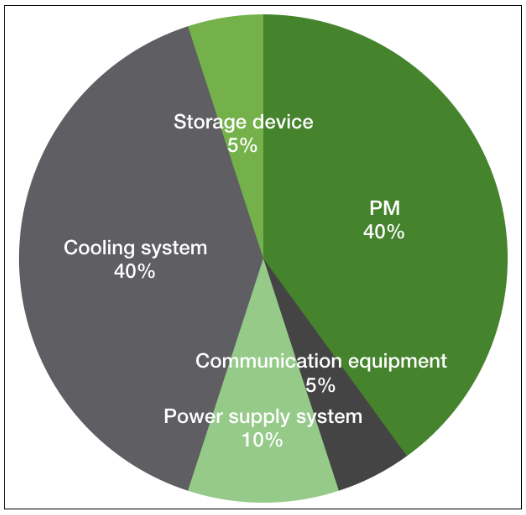
The recent development of application container such as Docker and Kubernetes [10] have attract the attention from both academia and industry. It provides a finer granularity of resource management by enabling an application level of operations including deployment, scaling, and migration.

Comparison between Container and VM

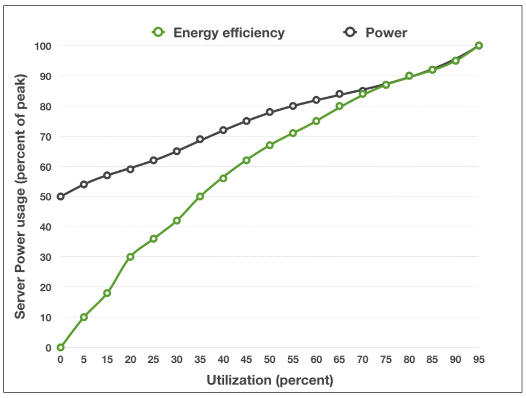
Several research [23,25,32,81] have compared VM-based and container-based virtualization. The advantages of container-based virtualization are mainly from three aspects:

- Low overhead, containers have a lightweight management. which generate much less overhead than a hypervisor. On the other hand, shared OS system also reduces the overhead on running multiple OSs.
- Containers have a near-native performance of CPU, memory, disk and network. While VM has a poor I/O performance [61]: up to 50% reduction of bandwidth (e.g hard disk and network). This defect also has a negative effect on the migration performance, since a VM-image is ranging from hundreds of MB to several GBs.
- Containers naturally support vertical scaling while VMs do not. Vertical scaling means a container can dynamically adjust its resources under the host's resource constraint.This feature offers a fine granularity management of resources.

On the other hand, the disadvantages of containers are categorized in two aspects. Currently, containers have rather poor performance in isolation for memory, disk, and network. Security is immature in containers [10], therefore, high security required applications are not recommended to be deployed in such systems.



(a) Energy consumption distribution of data centers [58]



(b) Disproportionate between utilization and energy consumption [5]

2.1.3 Energy Efficiency in Cloud Data centers

Apart from upfront investment, Energy consumption [38] is the major expense of data centers. Therefore, it is also the top concern of Cloud providers. Energy consumption is derived from several parts as illustrated in Figure 2.5a. Cooling system and servers or PMs account for a majority of the consumption. A recent survey [15] shows that the recent development of cooling techniques have reduced its energy consumption and now server consumption has become the dominate energy consumption component.

According to Hameed et al [31], servers are far from energy-efficient. The main reason for the wastage is that the energy consumption of servers remains high even when the utilization are low. Therefore, a concept of *energy proportional computing* [5] raised to address the disproportionate between utilization and energy consumption. This leads to using virtualization technology to achieve server consolidation.

2.1.4 Drawbacks of Traditional Service Models

There are three characteristics in IaaS which naturally lead to a low resource utilization.

- Resource over-provisioning
  As previous mentioned, VMs are either selected by Cloud users or automatically selected by software platforms. In either case, it requires an estimation of required resources. However, The accurate estimation is almost impossible because of unpredictable workloads; A simple way is to reserve more resources for ensuring the QoS at peak hours [14], rather than completely rely on auto-scaling, simply because auto-scaling is more expensive than reservation. However, the peak hours only account for a short period, therefore, in most of time, resources are wasted. In IaaS, the types of VM are a part of the contract, Cloud providers cannot simply change the type of VMs after provisioning.

- Unbalanced usage of resources
  Specific applications consume unbalanced resources which leads to vast amount of resource wastage [73]. For example, computation intensive tasks consume much more CPU than RAM; a fixed type of VM provides much more RAM than it needs. Because

the tasks use too much CPU, they prevent other tasks from co-allocating. This also causes wastage.

- Heavy overhead of VM hypervisors and redundant operating systems (OSs)
  In VM-based resource allocation, heavy overhead is caused by the hypervisor of VMs and the separated operating systems running in the PM. A hypervisor manages and monitors the VMs running on a PM. The overhead of a hypervisor is heavier with the increasing numbers of VM. Redundant operating system is another reason for overhead, as normal applications do not need specific operating systems; Commonly used OSs - such as Linux-based: RedHat, or Windows server versions - are well enough for their needs. Therefore, running applications in separated operating system simultaneously is unnecessary.

For traditional PaaS, Cloud providers can adjust the VMs' location and the type of VMs. Therefore, it overcomes the first drawback of IaaS. Because of PaaS is built upon IaaS, the one-on-one relationship between application and VM still exist. PaaS can only select the most suitable type instead of changing their sizes. Therefore, the unbalanced resource problem cannot be solved. In addition, PaaS brings a restriction for the applications deployed on it. PaaS build a software middle-ware to allow Cloud users' development. The middle-ware requires the deployed applications to be compatible with the environment, for example, Google App engine only allows certain programming languages and libraries. Therefore, the generality of PaaS is limited. It is urgent to provide a environment which supports automatic resource management as well as an editable programming environment.

2.1.5 Container as a Service

Container as a Service (CaaS) [55] is a new service model which is usually considered as a combination of IaaS and PaaS. CaaS uses containers and VM as its fundamental resource allocation unit as shown in Figure 2.3 on the right hand side.

CaaS has advantages of both IaaS and PaaS but without their disadvantages. On one hand - similar to PaaS - CaaS allows Cloud providers to manage resource in a fine granularity with containers, therefore, it may lead to high utilization of resources. On the other hand - similar to IaaS - CaaS allows customers to customize their software environment without being constrained by platforms. Therefore, it has more flexibility than PaaS.

2.2 Server consolidation

Server consolidation packs a number of VMs on fewer number of PMs to improve the resource utilization and decrease the energy consumed by PMs. It is often applied to solve the problem of physical server sprawl [39]: a situation that more PMs are used in a low-utilized way.

From a broad technologies perspective, there are generally two technologies can be used to achieve server consolidation: clustering and virtualization. Clustering is used in a situation that the applications running in PMs are I/O intensive. This is because current virtualization technologies such as KVM [40] or Xen [4] have a 20% to 55% of reduction of I/O bandwidth (e.g disk reads and writes, network bandwidth) in comparison with non-virtualized PM [61]. Virtualization is more suitable for applications which require little CPU utilization (e.g 15%) and low I/O needs. Web services are mostly categorized into this group. Three major benefits of virtualization make it be the first choice for web-based application consolidation: No reliance on hardware, easy to provision and live migration.

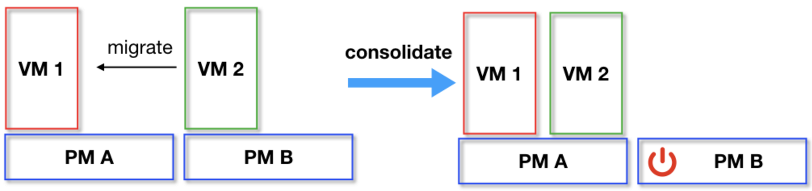


Figure 2.6: A Server Consolidation example: Initially, each PM runs an application wrapped with a VM in a low resource utilization state. After the consolidation, both VMs are running on PM A, so that PM B is turned off to save energy [5].

Virtualization-based Server consolidation [89] utilized a dynamic migration technique (e.g pre-copy [16] and post-copy [34]) to resolve the low utilization problem by gathering applications into a fewer number of PMs (see Figure 2.6), so that the resource utilization of PMs are maintained at a high level. In the meanwhile, idle PMs can be turned off to save energy. Consolidation dramatically improves hardware utilization and lowers PM and cooling energy consumption.

Server consolidation can be done in two ways: Static and Dynamic [77, 82] which are applied in different resource management scenarios (discuss in next section). In some scenarios, for example: new application initialization and global consolidation, involve large number of variables, therefore, it is very time-consuming job and often conducted in an off-line fashion. In other scenarios, when PMs are overloading or underloading, it requires fast a decision-making to migrate one or more VMs to reduce the burden on overloaded PM or improve the utilization. It migrates one VM at a time with a dynamic method.

2.2.1 General Server Consolidation Scenarios

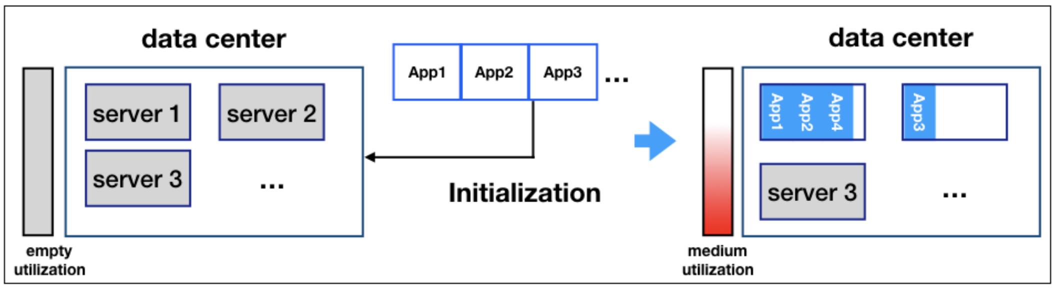
The server consolidation in data center can be applied to three [49,69] scenarios: Application initialization, Prediction and Global consolidation, and Dynamic resource management (see Figure 2.7).

1. *Application initialization* is applied when new applications or new VMs arrive and the problem is to allocate them into a minimum number of PMs.

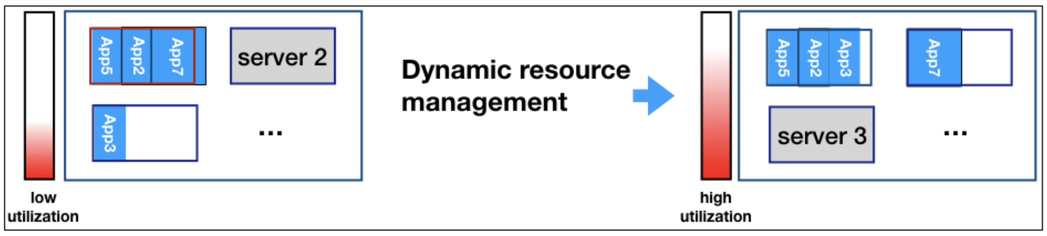
   In this problem, a set of applications or VMs are waiting in a queue. The resource capacity of the PM and usage by applications are characterized by a vector of resource utilizations including CPU, memory and etc. Then, the allocation system must select a minimum number of PMs to accommodate them so that after the allocation, the resource utilizations remain high. The problem is to consider the different combinations of applications so that the overall resource utilization is high. This problem is naturally modeled as a static bin-packing problem [17] which is a NP-hard problem meaning it is unlikely to find an optimal solution of a large problem.

2. *Prediction and Global consolidation* is conducted periodically to adjust the current allocation of applications so that the overall utilization is improved.

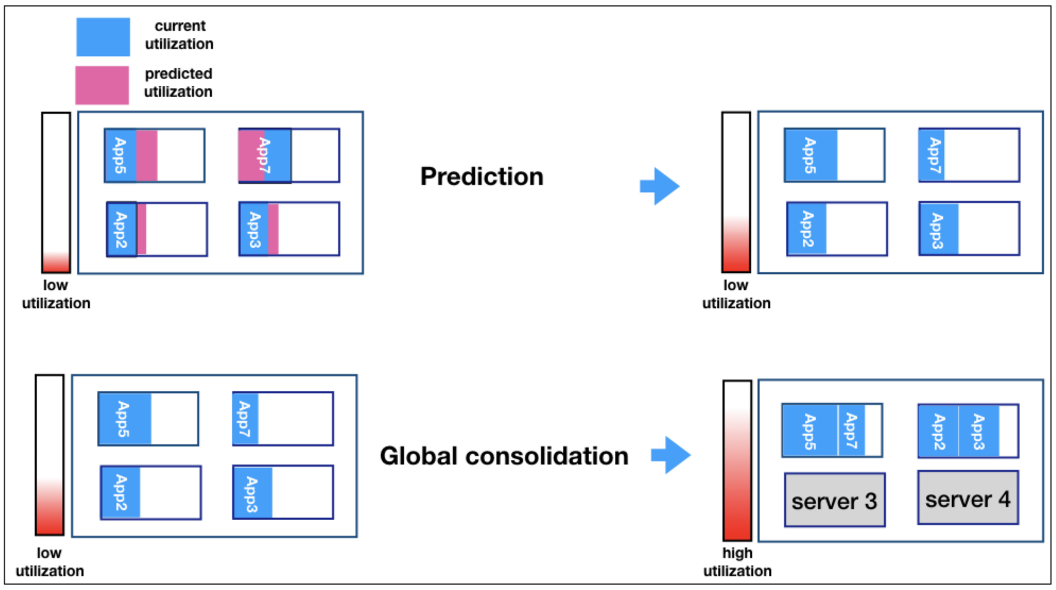
   In this problem, time is discrete and it can be split into basic time frames, for example: ten seconds. A periodical operation is conducted in every N time frames. A cloud data center has a highly dynamic environment with continuous arriving and releasing of applications. Releasing applications cause hollow in PMs; new arrivals cannot change the structure of current allocation. Therefore, after the initial allocation, the overall energy efficiency is likely to drop along with time elapsing.



(a) Initialization



(b) Dynamic resource management



(c) Prediction and Consolidation

Figure 2.7: Three scenarios of resource management

In prediction, an optimization system takes the current applications' utilization records as the input. Make a prediction of their utilization in the next period of time. In Global consolidation, based on the predict utilization and the current allocation - including a list of applications/VMs and a list of PMs, the system adjusts the allocation so that the global resource utilization is improved.

In comparison with initialization, instead of new arrivals, the global consolidation considers the previous allocation. Another major difference is that global consolidation needs to minimize the differences of allocation before and after the optimization. This is because the adjustment of allocation relies on a technique called live migration [**?**], and it is a very expensive operation because it occupies the resources in both the host and the target. Therefore, global optimization must be considered as a time-dependent activity which makes the optimization even difficult.

In comparison with dynamic consolidation, global consolidation takes a set of VMs as input instead of one. Therefore, it is time consuming and often treated as a static problem.

3. *Dynamic resource management* is applied in three scenarios. **First**, it is applied when a PM is overloading. In order to prevent the QoS from dropping, an application is migrated to another PM. This is called hot-spot mitigation [49]. **Second**, it is applied when a PM is under-loading. Under-loading is when a PM is in a low utilization state normally defined by a threshold. At this moment, all the applications in the under-loading PM are migrated to other active PMs, so the PM becomes empty and can be turned off. This is called dynamic consolidation. **Third**, it is applied when a PM having very high level of utilization while others having low. An adjustment is to migrate one or more application from high utilized PMs to low ones. This is called load balancing.

   No matter which scenario it is, a dynamic resource management always involves three steps .

   - *When to migrate?* refers to determine the time point that a PM is overloaded or underloaded. It is often decide by a threshold of utilization.
   - *Which application to migrate?* refers to determine which application need to be migrated so that it optimize the global energy consumption.
   - *Where to migrate?* refers to determine which host that an application is migrated to. This step is called dynamic placement which is directly related to the consolidation, therefore, it is decisive in improving energy-efficiency.

   Among three operations, dynamic placement is a dynamic and on-line problem. The term "dynamic" means the request comes at an arbitrary time point. An on-line problem is a problem which has on-line input and requires on-line output [**?**]. It is applied when a control system does not have the complete knowledge of future events.

   There are two difficulties in this operation, firstly, dynamic placement requires a fast decision while the search space is very large (e.g hundreds of thousands of PMs). Secondly, migrate one application at a time is hard to reach a global optimized state.

Finally, a consolidation plan includes four major items:

1. A list of existing PMs after consolidation

2. A list of new virtual machines created after consolidation

3. A list of old PMs to be turned off after consolidation

4. The exact placement of applications and services

2.2.2 VM-based Static Consolidation Techniques

Static initialization, is also frequently referred to initial placement problem [35]. Whenever a request for provisioning of applications by one or more Cloud users. The resource management system schedules the applications into a set of PMs. Currently, most state-of-the-art research focus on VM-based placement, in this case, applications are installed in VMs. Therefore, "application placement" and "VM placement" are used interchangeable in the literature.

In energy-aware resource management, the initialization has the objective of minimizing the used PMs. In literature, the static initialization problem is often modeled as the vector bin packing problem. Each application represents an item and PMs represents bins.

A d-dimensional Vector Bin Packing Problem (VBP_d), give a set of items $I^1, I^2, \ldots, I^n$ where each item has d dimension of resources represented in real number $I^i \in R^d$. A valid solution is packing I into bins $B^1, B^2, \ldots, B^k$. For each bin and each dimension, the sum of resources can not exceed the capacity of bin. The goal of Vector Bin Packing problem is to find a valid solution with minimum number of bins. VBP_d is an NP-hard problem.

Because of its NP-hard nature, several researchers propose well-known bin-packing heuristics First Fit (FF), First Fit Decreasing (FFD), Best Fit (BF) and etc. These algorithms have constant-factor approximation to bin-packing problems. However, VM placement is more complex than bin-packing problem [46],

Panigraph et al [52] study variants of First Fit Decreasing (FFD) algorithms and inspired by bad isntances for FFD-type algorithms, they propose a geometric heuristics which outperform FFD-based heuristics in most of cases.

2.2.3 VM-based Dynamic Consolidation Techniques

2.3 Container-based Consolidation Techniques

Initialization is one of the major step in resource management. It can be considered as a static problem [35] or dynamic problem [8]. As we discussed in the previous section, a dynamic allocation normally cannot gives a global optimized solution of a batch of tasks. Therefore, in the context of maximizing the energy efficiency of a data center, we category initialization into a static optimization approach.

Mesos [33] is a platform for sharing commodity clusters between cluster computing frameworks such as Hadoop and MPI. It has a two-level of resource allocation architecture where a master node and several slave node. A master node only decides how many resources to offer to each framework (slave) based on fair sharing policy [29]. Each slave node belongs to a cluster framework and it makes the decision of which resources to use. Each framework has to define its allocation policy. Mesos focus on sharing resources across multiple frameworks and has a better scalability since it deligates the application placement to decentralised slave nodes. The main problem for Mesos is that it does not consider energy consumption for Cloud providers with user-defined allocation policy.

Piraghaj et al [54] propose an architecture for container-based resource management. Their allocation approach is policy-based, where it allocates each VM with containers until its estimated utilization above 90%. This heuristic is similar to First Fit, it is fast but does not guarantee global optimal.

2.4 Bilevel Optimization

2.5

Chapter 3

Preliminary Work

3.1 Background

3.1.1 Traditional approaches

Ref. [87] proposes a single-objective genetic algorithm to solve placement of service (SaaS) on physical machines. Their major contributions are three-fold. Firstly, they consider web services as a workflow and optimize the makespan of a workflow. Secondly, they design a representation to the problem. Thirdly, they do not only consider computing nodes, but storage nodes as well.

Ref. [84] develops a *Resource-Allocation-Throughput (RAT)* model for web service allocation. The *RAT model* mainly defines several important variables for an atomic service which represents a software component. Based on this model, firstly, an atomic service's throughput equals its coming rate if the resources of the allocated VM are not exhausted. Secondly, increasing the coming rate will also increase an atomic service's throughput until the allocated resource is exhausted. Thirdly, when the resource is exhausted, the throughput will not increase as request increasing. At this time, the virtual machine reaches its capacity.

Anton Beloglazov et al. [6] propose two algorithms for VM allocation. The first one is a bin-packing algorithm, called Modified Best Fit decreasing (MBFD) which is used when a new VM allocation request arrives. The second algorithm, named Minimization of Migration, is used to adjust the current VMs allocation according to the CPU utilization of a physical machine. Their experiments have shown that these methods lead to a substantial reduction of energy consumption in Cloud data centers.

3.1.2 Power Model

Shekhar's research [68] is one of the earliest in energy aware consolidation for cloud computing. They conduct experiments of independent applications running in physical machines. They explain that CPU utilization and disk utilization are the key factors affecting the energy consumption. They also find that only consolidating services into the minimum number of physical machines does not necessarily achieve energy saving, because the service performance degradation leads to a longer execution time, which increases the energy consumption.

Bohra [11] develops an energy model to profile the power of a VM. They monitor the sub-components of a VM which includes: CPU, cache, disk, and DRAM and propose a linear model (Eq 3.1). Total power consumption is a linear combination of the power consumption of CPU, cache, DRAM and disk. The parameters α and β are determined based on the observations of machine running CPU and IO intensive jobs.

$$P_{(total)} = \alpha P_{\{CPU,cache\}} + \beta P_{\{DRAM,disk\}} \tag{3.1}$$

Although this model can achieve an average of 93% of accuracy, it is hard to be employed in solving SRAC problem, for the lack of data.

Anton Beloglazov et al. [6] propose a comprehensive energy model for energy-aware resource allocation problem (Eq 3.2). P_{max} is the maximum power consumption when a virtual machine is fully utilized; k is the fraction of power consumed by the idle server (i.e. 70%); and u is the CPU utilization. This linear relationship between power consumption and CPU utilization is also observed by [41, 57].

$$P(u) = k \cdot P_{max} + (1-k) \cdot P_{max} \cdot u \tag{3.2}$$

3.1.3 Multi-objective Evolutionary Optimization

A multi-objective optimization problem consists of multiple objective functions to be optimized. A multi-objective optimization problem can be stated as follows:

$$\min \ \vec{f}(\vec{x}) = (f_1(\vec{x}), \ldots, f_m(\vec{x})), \tag{3.3}$$
$$s.t. \ \vec{x} \in \Omega. \tag{3.4}$$

where Ω stands for the feasible region of $\vec{x}$.

Multi-objective Evolutionary Optimization Algorithm (MOEA) are ideal for solving multi-objective optimization problems [22], because MOEAs work with a population of solutions. With an emphasis on moving towards the true Pareto-optimal region, a MOEA algorithm can be used to find multiple Pareto-optimal solutions in one single simulation run [37]. Therefore, this project would employ MOEA approaches. This is also the first time to employ MOEAs technique for SRAC problem.

3.2 Problem Description

We consider the problem as a multi-objective problem with two potentially conflicting objectives, minimizing the overall cost of web services and minimizing the overall energy consumption of the used physical machines.

To solve the SRAC problem, we model an atomic service as its request and requests' coming rate, also known as frequency.

The request of an atomic service is modeled as two critical resources: CPU time $A = \{A_1, A_i, \ldots, A_t\}$ and memory consumption $M = \{M_1, M_i, \ldots, M_t\}$, for each request consumes a A_i amount of CPU time and M_i amount of memory. The coming rate is denoted as $R = \{R_1, R_i, \ldots, R_t\}$. In real world scenario, the size and the number of a request are both variant which are unpredictable, therefore, this is one of the major challenges in Cloud resource allocation. In this paper, we use fixed coming rate extracted from a real world dataset to represent real world service requests.

The cloud data center has a number of available physical machines which are modeled as CPU time $PA = \{PA_1, PA_j, \ldots, PA_p\}$ and memory $PM = \{PM_1, PM_j, \ldots, PM_p\}$. PA_j denotes the CPU capacity of a physical machine and PM_j denotes the size of memory. A physical machine can be partitioned or virtualized into a set of virtual machines; each virtual machine has its CPU time $VA = \{VA_1, VA_n, \ldots, VA_v\}$ and memory $VM = \{VM_1, VM_n, \ldots, VM_v\}$.

The decision variable of service allocation is defined as X_n^i. X_n^i is a binary value (e.g. 0 and 1) denoting whether a service i is allocated on a virtual machine n. The decision variable of virtual machine allocation is defined as Y_j^n. Y_j^n is also binary denoting whether a VM n is allocated on a physical machine j.

In this work, we consider homogeneous physical machine which means physical machines have the same size of CPU time and memory. The utilization of a CPU of a virtual machine is denoted as $U = \{U_1, U_n, \ldots, U_v\}$. The utilization can be calculated by Eq.3.5.

$$U_n = \begin{cases} \frac{\sum_{i=1}^{t} R_i \cdot A_i \cdot X_n^i}{VA_n}, \text{If } \frac{\sum_{i=1}^{t} R_i \cdot A_i \cdot X_n^i}{VA_n} < 1 \\ 1 \qquad\qquad\qquad\qquad , \text{otherwise} \end{cases} \tag{3.5}$$

The cost of a type of virtual machine is denoted as $C = \{C_1, C_n \ldots, C_v\}$.

In order to satisfy the performance requirement, Service providers often define Service Level Agreements (SLAs) to ensure the service quality. In this work, we define throughput as a SLA measurement [53]. Throughput denotes the number of requests that a service could successfully process in a period of time. According to *RAT* model, the throughput is equal to the number of requests when the allocated resource is sufficient. Therefore, if a VM reaches its utilization limitation, it means that the services have been allocated exceedingly. Therefore, all services in that VM suffer from performance degradation.

Then we define two objective functions as the total energy consumption and the total cost of virtual machines:

$$\begin{aligned} &\text{minimize} \\ &Energy = \sum_{j=1}^{p} (k \cdot V_{max} + (1-k) \cdot V_{max} \cdot \sum_{n=1}^{v} U_n \cdot Y_j^n) \end{aligned} \tag{3.6}$$

$$Cost = \sum_{j=1}^{p} \sum_{n=1}^{v} C_n \cdot Y_j^n \tag{3.7}$$

Hard constraint

A virtual machine can be allocated on a physical machine if and only if the physical machine has enough available capacity on every resource.

$$\begin{aligned} \sum_{n=1}^{v} VM_n \cdot Y_j^n \leq PM_j \\ \sum_{n=1}^{v} VA_n \cdot Y_j^n \leq PA_j \end{aligned} \tag{3.8}$$

Soft constraint

A service can be allocated on a virtual machine even if the virtual machine does not have enough available capacity on every resource, but the allocated services will suffer from a quality degradation.

$$\sum_{i=1}^{t} M_i \cdot R_i \cdot X_i^n \leq VM_n \tag{3.9}$$

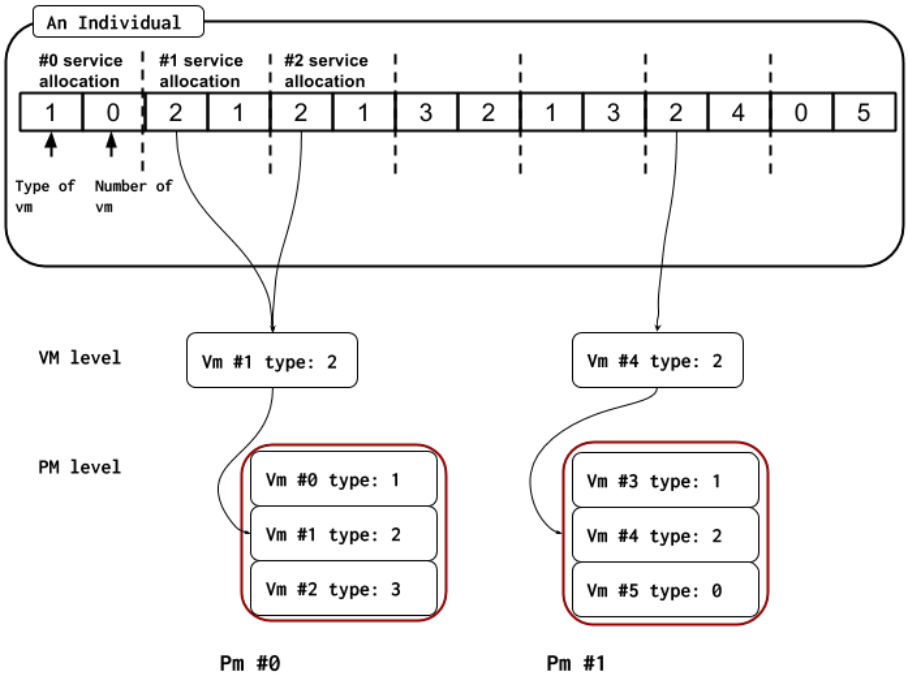


Figure 3.1: An example chromosome representation

3.3 Methods

As we have discussed, Multi-objective Evolutionary Algorithms are good at solving multi-objective problems and NSGA-II [20] has shown his effective and efficiency. NSGA-II is a well-known MOEA that has been widely used in many real-world optimization problems. In this paper we also adopt NSGA-II to solve the SRAC problem. We first propose a representation and then present a NSGA-II based algorithm with novel genetic operators.

3.3.1 Chromosome Representation

SRAC is a two-level bin-packing problem, in the first level, bins represent physical machines and items represent virtual machines. Whereas, in the second level, a virtual machine acts like a bin and web services are items. Therefore, we design the representation in two hierarchies, virtual machine level and physical machine level.

Figure 3.1 shows an example individual which contains seven service allocations. Each allocation of a service is represented as a pair where the index of each pair represents the number of web service. The first number indicates the type of virtual machine that the service is allocated in. The second number denotes the number of virtual machine. For example, in Figure 3.1, service #1 and service #2 are both allocated in the virtual machine #1 while service #1 and service #5 are allocated to different virtual machines sharing the same type. The first hierarchy shows the virtual machine in which a service is allocated by defining VM type and number. Note that, the VM type and number are correlated once they are initialized. With this feature, the search procedure is narrowed down in the range of existing VMs which largely shrinks the search space. The second hierarchy shows the relationship between a physical machine and its virtual machines, which are implicit. The physical machine is dynamically determined according to the virtual machines allocated on it. For example, in Figure 3.1, the virtual machines are sequentially packed into physical machines. The boundaries of PMs are calculated by adding up the resources of VMs until one of the resources researches the capacity of a PM. At the moment, no more VMs can be packed into the PM, then the boundary is determined. The reason we designed this heuristic is because a physical machine is always fully used before launching another. Therefore, VM consolidation is inherently achieved.

Clearly, specifically designed operators are needed to manipulate chromosomes. Therefore, based on this representation, we further developed initialization, mutation, constraint handling and selection method.

3.3.2 Initialization

Algorithm 1 Initialization

```
Inputs:
VM CPU Time $VA$ and memory $VM$,
Service CPU Time $A$ and memory $M$
consolidation factor c
Outputs: A population of allocation of services
 1: for Each service $t$ do
 2:     Find its most suitable VM Type
 3:     Randomly generate a VM type $vmType$ which is equal or better than its most suitable type
 4:     if There are existing VMs with $vmType$ then
 5:         randomly generate a number $u$
 6:         if $u <$ consolidation factor then
 7:             randomly choose one existing VM with $vmType$ to allocate
 8:         else
 9:             launch a new VM with $vmType$
10:         end if
11:     else
12:         Create a new VM with its most suitable VM type
13:     end if
14: end for
```

The initialization (see Alg 1) is designed to generate a diverse population. In the first step, for each service, it is able to find the most suitable VM type which is just capable of running the service based on its resource requirements. In the second step, based on the suitable VM type, a stronger type is randomly generated. If there exists a VM with that type, the service is either deployed in the existing VM or launch a new VM. We design a consolidation factor c which is a real number manually selected from 0 to 1 to control this selection. If a random number u is smaller than c, the service is consolidated in an existing VM.

This design could adjust the consolidation, therefore, controls the utilization of VM.

3.3.3 Mutation

The design principle for mutation operator is to enable individuals exploring the entire feasible search space. Therefore, a good mutation operator has two significant features, the exploration ability and the its ability to keep an individual within the feasible regions. In order to achieve these two goals, firstly, we generate a random virtual machine type which has a greater capacity than the service needs. It ensures the feasible of solutions as well as exploration capability. Then, we consider whether a service is consolidated with the consolidation factor c.

The consolidation is conducted with a roulette wheel method which assigns fitness value to each VM according to the reciprocal of its current utilization. The higher the utilization, the lower the fitness value it is assigned. Therefore, a lower utilization VM has a greater

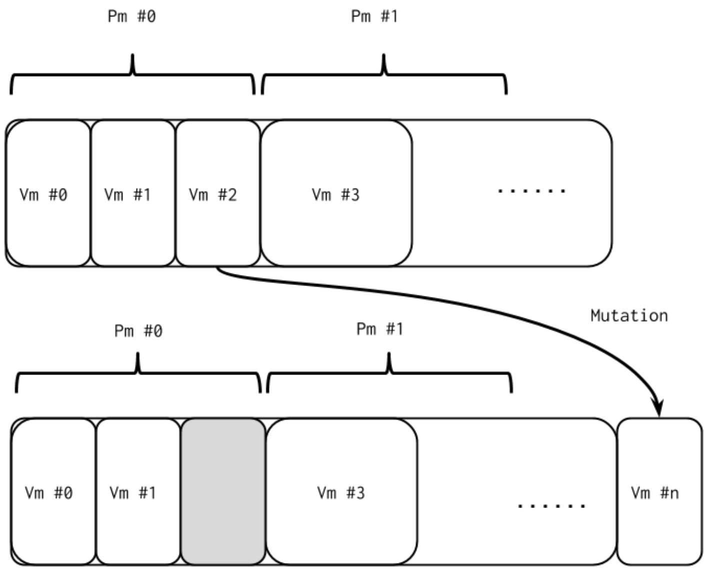


Figure 3.2: An example mutation without insertion that causes a lower resource utilization

probability to be chosen. At last, if a new VM is launched, it will not be placed at the end of VM lists. Instead, it will be placed at a random position among the VMs. The reason is illustrated in Figure 3.2. In the example, VM #2 is mutated into a new type and be placed at the end of the VM list. However, because of the size of VM #3 is too large for PM #0, the hollow in PM #0 will never be filled. This problem can be solved with the random insertion method.

Algorithm 2 Mutation

Inputs:
An individual VM CPU Time VA and memory VM,
Service CPU Time A and memory M
consolidation factor c
Outputs: A mutated individual

```
 1: for Each service do
 2:     Randomly generate a number $u$
 3:     if $u <$ mutation rate then
 4:         find the most suitable VM Type for this service
 5:         Randomly generate a number $k$
 6:         if $k <$ consolidation factor then
 7:             calculate the utilization of used VMs
 8:             assign each VM with a fitness value of 1 / utilization and generate a roulette wheel according to
                their fitness values
 9:             Randomly generate a number $p$, select the VM according to $p$
10:             Allocate the service
11:         else
12:             launch a new VM with the most suitable VM Type
13:             insert the new VM in a randomly choose position
14:         end if
15:     end if
16: end for
```

3.3.4 Violation control method

A modified violation ranking is proposed to deal with the soft constraint, for the hard constraint is automatically eliminated by the chromosome representation. We define a violation number as the number of services which are allocated in the degraded VMs. That is, if there are excessive services allocated in a VM, then all the services are suffered from a degraded in performance. The violation number is used in the selection procedure, where the individuals with less violations are always preferred.

3.3.5 Selection

Our design uses the binary tournament selection with a constrained-domination principle. A constrained-domination principle is defined as following. A solution I is considered constraint-dominate a solution J, if any of the following condition is true:

1. Solution I is feasible, solution is not,
2. Both solutions are infeasible, I has smaller overall violations,
3. Both solutions are feasible, solution I dominates solution J.

An individual with no or less violation is always selected. This method has been proved effective in the original NSGA-II paper [20].

3.3.6 Fitness Function

The cost fitness (Eq.3.7) is determined by the type of VMs at which web service are allocated. The energy fitness is shown in Eq.3.6, the utilizations (Eq.3.5) of VM are firstly converted into the utilizations of PM according to the proportion of VMs and PMs CPU capacity.

3.3.7 Algorithm

The main difference between our approach and the original NSGA-II is that our approach has no crossover operator.

That is, a random switch of chromosome would completely destroy the order of VMs, hence, no useful information will be preserved. Therefore, we only apply mutation as the exploration method. Then, the algorithm becomes a parallel optimization without much interaction between its offspring, which is often addressed as Evolutionary Strategy [42].

3.4 Experiment

3.4.1 Dataset and Problem Design

This project is based on both real-world datasets *WS-Dream* [90] and simulated datasets [12]. The *WS-Dream* contains web service related datasets including network latency and service frequency (request coming rate). In this project, we mainly use the service frequency matrix. For the cost model, we only consider the rental of virtual machines with fixed fees (monthly rent). The configurations of VMs are shown in Table 3.2, the CPU time and memory were selected manually and cost were selected proportional to their CPU capacity. The maximum PM's CPU and memory are set to 3000 and 8000 respectively. The energy consumption is set to 220W according to [12].

We designed six problems shown in Table 3.1, listed with increasing size and difficulty, which are used as representative samples of SRAC problem.

Algorithm 3 NSGA-II for SRAC

```
Inputs:
VM CPU Time $VA$ and memory $VM$,
PM CPU Time $PA$ and memory $PM$,
Service CPU Time $A$ and memory $M$
consolidation factor c
Outputs: A Non-dominated Set of solutions
 1: Initialize a population $P$
 2: while Termination Condition is not meet do
 3:     for Each individual do
 4:         Evaluate the fitness values
 5:         Calculate the violation
 6:     end for
 7:     non-Dominated Sorting of $P$
 8:     calculate crowding distance
 9:     while child number is less than population size do
10:         Selection
11:         Mutation
12:         add the child in a new population U
13:     end while
14:     Combine $P$ and $U$ { for elitism}
15:     Evaluate the combined $P$ and $U$
16:     Non-dominated sorting and crowding distance for combined population
17:     Include the top popSize ranking individuals to the next generation
18: end while
```

Table 3.1: Problem Settings

Problem	1	2	3	4	5	6
Number of services	20	40	60	80	100	200

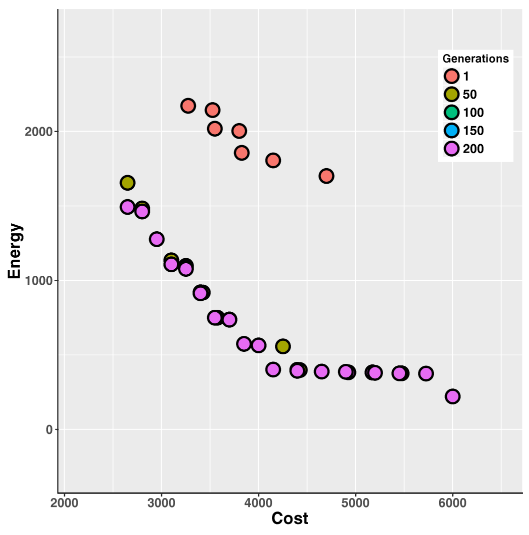
Selection Method with violation Control vs. without violation control

We conducted two comparison experiments. For the first experiment, we make a comparison between NSGA-II with violation control and NSGA-II without violation control. In second experiment, two mutation operators are compared. The first is the roulette wheel mutation, the second is the mutation with greedy algorithm. The mutation with greedy algorithm is a variant of roulette wheel mutation. The only difference is that instead of selecting a VM to consolidate with fitness values, it always selects the VM with the lowest CPU utilization. Therefore, it is a greedy method embedded in the mutation.

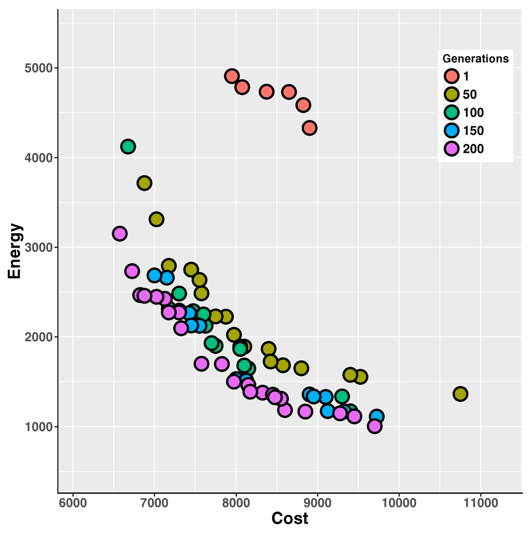
The experiments were conducted on a personal laptop with 2.3GHz CPU and 8.0 GB RAM. For each approach, 30 independent runs are performed for each problem with constant population size 100. The maximum number of iteration is 200. k equals 0.7. We set mutation rate and consolidation factor to 0.9 and 0.01.

Table 3.2: VM configurations

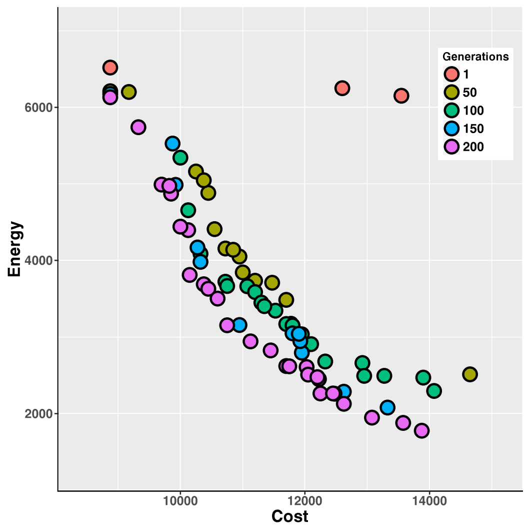
VM Type	CPU Time	Memory	Cost
1	250	500	25
2	500	1000	50
3	1500	2500	150
4	3000	4000	300



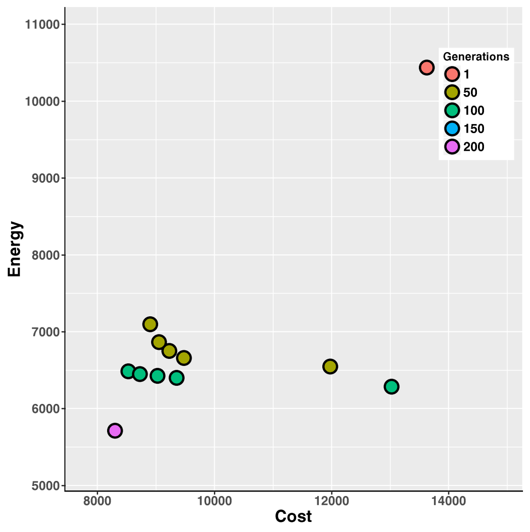
(a) Problem 1



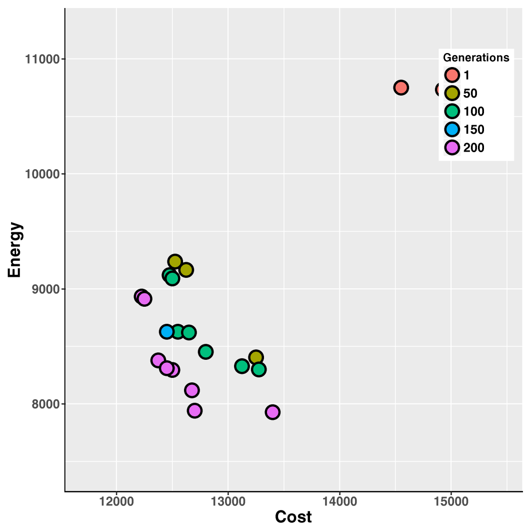
(b) Problem 2



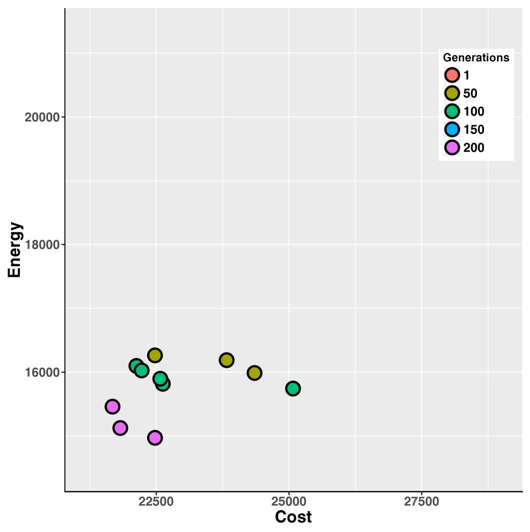
(c) Problem 3



(d) Problem 4



(e) Problem 5



(f) Problem 6

Figure 3.3: Non-dominated solutions evolve along with the generation

3.4.2 Results

Table 3.3: Comparison between two Mutation methods

Problem	roulette wheel mutation		Greedy mutation	
	cost fitness	energy fitness	cost fitness	energy fitness
1	2664.6 ± 66.4	1652.42 ± 18.2	2661.7 ± 56.9	1653.2 ± 18.2
2	6501.1 ± 130.2	4614.0 ± 110.7	6495.37 ± 110.7	4132.5 ± 80.4
3	8939.2 ± 118.5	6140.7 ± 204.0	9020.5 ± 204.0	5739.6 ± 148.6
4	11633.7 ± 301.1	9301.9 ± 254.0	12900.6 ± 243.0	9376.3 ± 120.9
5	14102.0 ± 231.7	10164.8 ± 238.9	14789.2 ± 238.8	9876.3 ± 120.9
6	27194.3 ± 243.0	19914.4 ± 307.5	27654.2 ± 307.5	19187.1 ± 176.6

As we conducted the experiment for 30 runs, we first obtain an average non-dominated set over 30 runs by collecting the results from a specific generation from all 30 runs, and then apply a non-dominated sorting over them.

Firstly, we show the non-dominated solutions evolve along with the evolution process in Figure 3.3. These results come from selection method without violation control. As it illustrated, different colors represent different generations from 0th to 200th. For problem 1, because the problem size is small, the algorithm converged before 100 generations. Therefore, the non-dominated set from the 100th and 150th generations are overlapping with results from the 200th generation. For problem 2 and problem 3, it clearly shows the improvement of fitness values. For problem 4 onwards, the algorithm can only obtain a few solutions as the problem size is large, it is difficult to find solutions.

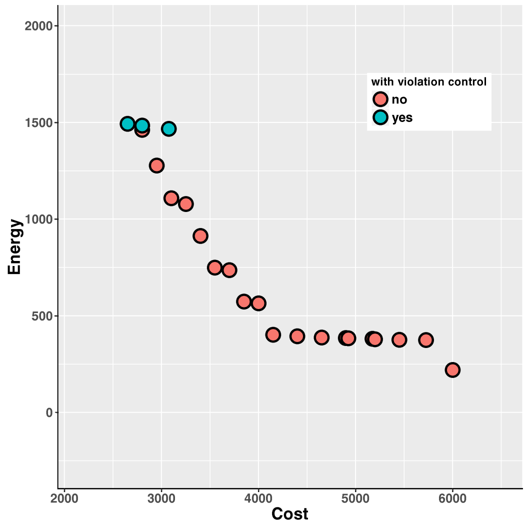
Then, the non-dominated sets of the last generation from two selection methods are compared in Figure 3.4. There are much fewer results are obtained from the violation control method throughout all cases. For the first three problems, the non-dominated set from the violation control method has similar quality as the no violation control method. From problem 4 onwards, the results from selection with violation control are much worse in terms of fitness values. However, most of the results from non-violation control selection have a high violation rate. That is, the method without violation control is stuck in the infeasible regions and provide high-violation rate solutions.

From figure 3.5, we can observe the violation rate between two methods. It proves violation control has a great ability to prevent the individual from searching the infeasible region. On the other hand, without violation control, although, the algorithm can provide more solutions with better fitness values, most of them have a high violation rate over 10% which are not very useful in reality.

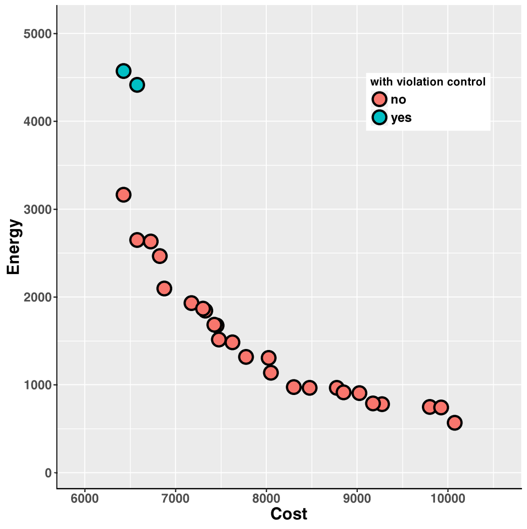
As we mentioned in previous section, the mutation rate and consolidation factor are set differently for the two methods. For the method with violation control, the mutation rate is set to 0.9 and the consolidation factor c is set to 0.01, this is because the feasible region is narrow and scattered. In order to avoid stucking in the local optima, a large mutation rate can help escape local optima. For the factor c, a larger percentage would easily lead the algorithm to infeasible regions. Therefore, it is set to a small number.

Mutation with roulette wheel vs. Mutation with greedy algorithm

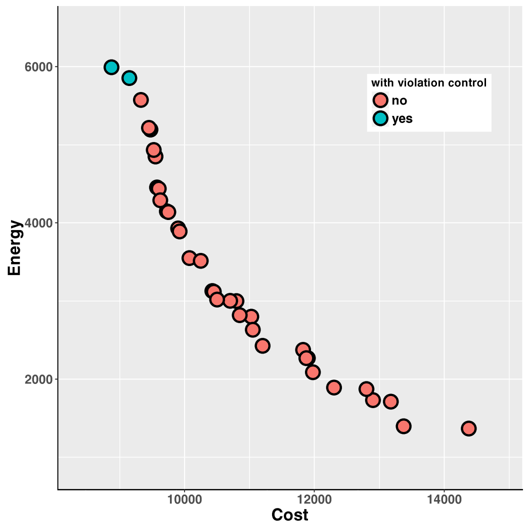
Table 3.3 shows the fitness value comparison between mutation methods. According to statistics significant test, there is little difference between methods. The possible reason is the consolidation factor is set to 0.01. In each mutation iteration, there is only 1% probability that



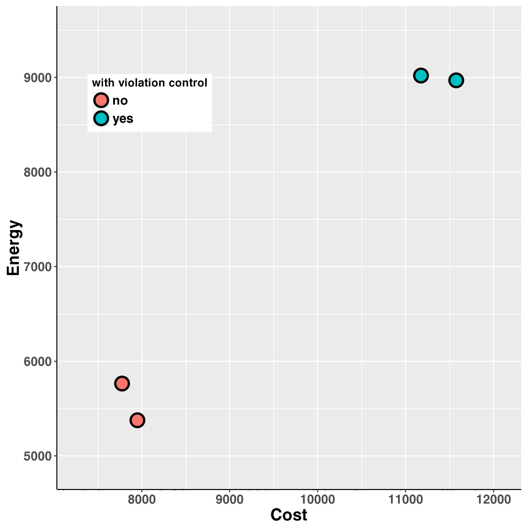
(a) Problem 1



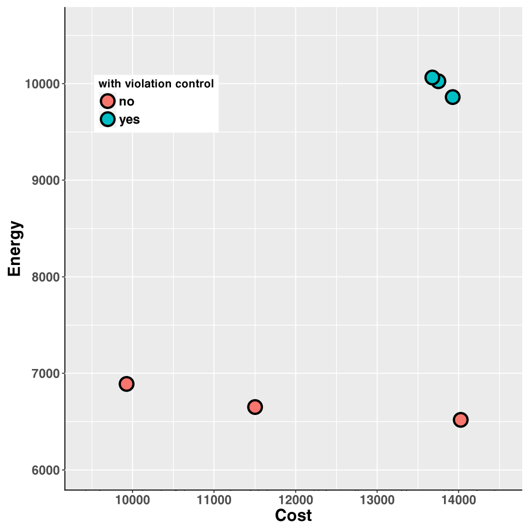
(b) Problem 2



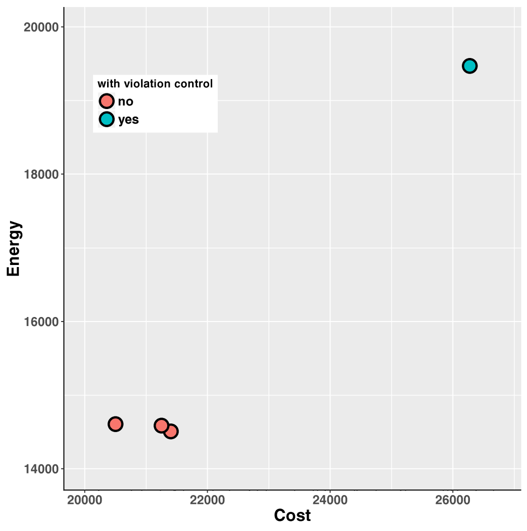
(c) Problem 3



(d) Problem 4



(e) Problem 5



(f) Problem 6

Figure 3.4: non-dominated solutions comparison between selection with violation control and without violation control

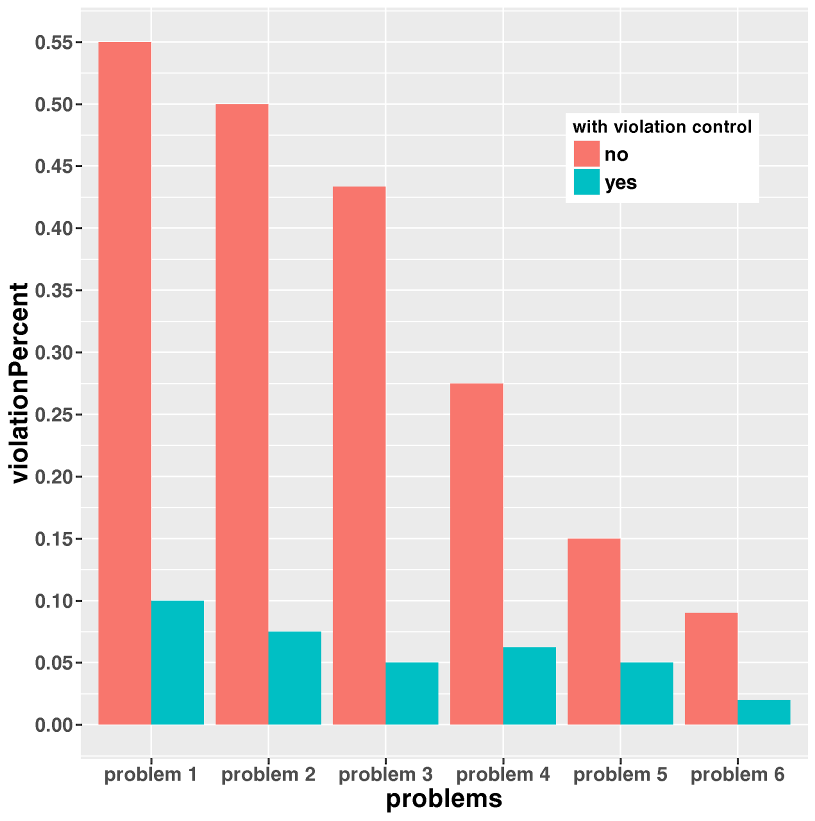


Figure 3.5: Violation Percentage comparison between selection with violation control and without violation control

a service will be consolidated in an existed VM, therefore, the influence between different consolidation strategies is trivial.

3.5 Conclusion

In this paper, we first propose a multi-objective formulation of a two levels of bin packing problem, web service resource allocation in Cloud. It solves the resource allocation in IaaS and SaaS at the same time. Two objectives, minimizing the cost from service providers' perspective and minimizing the energy consumption from cloud provider's objective are achieved. Secondly, we propose a NSGA-II based algorithm with specific designed genetic operators to solve the problem. The results are compared with different variances of the algorithm. The results show our approach can solve the very complicate optimization problem.

With current work as a baseline, in future work, we could improve the quality of solutions as well as provide better violation control mechanisms.

Chapter 4

Proposed Contributions and Project Plan

This thesis will contribute to the field of Cloud Computing by proposing novel solutions for the joint allocation of container and VM, and to the field of Evolutionary Computation by proposing new representations and genetic operators in evolutionary algorithms. The proposed contributions of this project are listed below:

1. Propose a new model for the joint allocation of container and VM problem. New representations will be proposed to problem. The first EC-approach to solve the problem. Reduce the energy consumption of data centers.

2. Propose a new robustness measure for time-dependent server consolidation problem. New optimization algorithms not only consider the two-level of packing problem but also take the previous and next consolidation into considered.

3. Develp a GP-based hyper-heuristic approach to automatically generate dispatching rules for dynamic server consolidation problem. New functional and primitive set for constructing useful dispatching rules will be studied. New genetic operations and representations will be proposed.

4. Develop a preprocessing algorithm to reduce the number of variables in a large scale static consolidation problem without scacrificing too much performance. Useful features of server consolidation problem will be studied.

4.1 Overview of Project Plan

Six overall phases have been defined in the initial research plan for this PhD project, as shown in Figure **??**. The first phase, which comprises reviewing the relevant literature, investigating both VM-based and container-based server consolidation algorithms, and producing the proposal, has been completed. The second phase, which corresponds to the first objective of the thesis, is currently in progress and is expected to be finished on time, thus allowing the remaining phases to also be carried out as planned.

4.2 Project Timeline

The phases included in the plan above are estimated to be completed following the timeline shown in Figure **??**, which will serve as a guide throughout this project. Note that part of the first phase has already been done, therefore the timeline only shows the estimated remaining time for its full completion.

4.3 Thesis Outline

The completed thesis will be organised into the following chapters:

- *Chapter 1: Introduction*
  This chapter will introduce the thesis, providing a problem statement and motivations, defining research goals and contributions, and outlining the structure of this dissertation.

- *Chapter 2: Literature Review*
  The literature review will examine in the existing work on VM-based and container-based server consolidation, discussing the fundamental concepts in this field in order to provide the reader with the necessary background. Multiple sections will then follow, considering issues such as static consolidation, dynamic consolidation, and large-scale of consolidation problem. The focus of this review is on investigating server consolidation techniques that are based on Evolutionary Computation.

- *Chapter 3: An EC Approach to the joint Allocation of Container and Virtual Machine*
  This chapter will establish a new model for the joint allocation of container and virtual machine problem. Furthermore, this chapter will introduce a new bi-level approach to solve this problem. One of the critical aspects of this approach is the representation of solution. Therefore, multiple representations will be proposed, analysed and compared.

- *Chapter 4: An EC Approach to the Time-dependent Global Server Consolidation*
  In this chapter, a robustness measure of time-dependent server consolidation will be proposed. Furthermore, a time-dependent optimization techniques will be employed to solve this problem. In the first, it will only consider the previous allocation results to minimize both cost of migration as well as the overall energy consumption of data center. Then, this approach will be generalized to consider next consolidation.

- *Chpater 5: A Genetic Programming-based Hybrid-heuristic Approach to Dynamic Consolidation*
  This chapter focuses on providing a Genetic Programming-based hybrid heuristic approach to automatic generate dispatching rules to dynamic consolidation problem.

- *Chapter 6: A Preprocessing Algorithm for Large-scale Static Consolidation*
  This chapter proposes a preprocessing algorithm for large scale static consolidation to reduce the number of variables so that the static consolidation can be more scalable.

- *Chapter 7: Conclusions and Future Work* In this chapter, conclusions will be drawn from the analysis and experiments conducted in the different phases of this research, and the main findings for each one of them will be summarised. Additionally, future research directions will be discussed.

4.4 Resources Required

4.4.1 Computing Resources

An experimental approach will be adopted in this research, entailing the execution of experiments that are likely to be computationally expensive. The ECS Grid computing facilities can be used to complete these experiments within reasonable time frames, thus meeting this requirement.

4.4.2 Library Resources

The majority of the material relevant to this research can be found online, using the universitys electronic resources. Other works may either be acquired at the universitys library, or by soliciting assistance from the Subject Librarian for the fields of engineering and computer science.

4.4.3 Conference Travel Grants

Publications to relevant venues in this field are expected throughout this project, therefore travel grants from the university are required for key conferences.

Bibliography

[1] Al-Roomi, M., and Al-Ebrahim, S. Cloud computing pricing models: a survey. *Computing* (2013).

[2] Angelo, J. S., Krempser, E., and Barbosa, H. J. C. Differential evolution for bilevel programming. In *2013 IEEE Congress on Evolutionary Computation (CEC)* (2013), IEEE, pp. 470–477.

[3] Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. Genetic programming: an introduction, 1998.

[4] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T. L., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. Xen and the art of virtualization. *SOSP* (2003), 164.

[5] Barroso, L. A., and Hölzle, U. The Case for Energy-Proportional Computing. *IEEE Computer 40*, 12 (2007), 33–37.

[6] Beloglazov, A., Abawajy, J., and Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems 28*, 5 (2012), 755–768.

[7] Beloglazov, A., Abawajy, J. H., and Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Comp. Syst. 28*, 5 (2012), 755–768.

[8] Beloglazov, A., and Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation - Practice and Experience 24*, 13 (2012), 1397–1420.

[9] Beloglazov, A., and Buyya, R. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints. *IEEE Transactions on Parallel and Distributed Systems 24*, 7 (2013), 1366–1379.

[10] Bernstein, D. Containers and Cloud - From LXC to Docker to Kubernetes. *IEEE Cloud Computing* (2014).

[11] Bohra, A. E. H., and Chaudhary, V. VMeter: Power modelling for virtualized clouds. In *Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on* (2010), Ieee, pp. 1–8.

[12] Borgetto, D., Casanova, H., Da Costa, G., and Pierson, J.-M. Energy-aware service allocation. *Future Generation Computer Systems 28*, 5 (2012), 769–779.

[13] BURKE, E. K., HYDE, M. R., AND KENDALL, G. Evolving Bin Packing Heuristics with Genetic Programming. *PPSN 4193*, Chapter 87 (2006), 860–869.

[14] CHAISIRI, S., LEE, B.-S., AND NIYATO, D. Optimization of Resource Provisioning Cost in Cloud Computing. *IEEE Trans. Services Computing 5*, 2 (2012), 164–177.

[15] CHO, J., AND KIM, Y. Improving energy efficiency of dedicated cooling system and its contribution towards meeting an energy-optimized data center. *Applied Energy 165* (Mar. 2016), 967–982.

[16] CLARK, C., FRASER, K., HAND, S., HANSEN, J. G., JUL, E., LIMPACH, C., PRATT, I., AND WARFIELD, A. Live migration of virtual machines. 273–286.

[17] COFFMAN JR, E. G., GAREY, M. R., AND JOHNSON, D. S. *Approximation algorithms for bin packing: a survey*. PWS Publishing Co., Aug. 1996.

[18] COLSON, B., MARCOTTE, P., AND SAVARD, G. An overview of bilevel optimization. *Annals OR 153*, 1 (2007), 235–256.

[19] DAYARATHNA, M., WEN, Y., AND FAN, R. Data Center Energy Consumption Modeling - A Survey. *... Surveys & Tutorials* (2016).

[20] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (Apr. 2002), 182–197.

[21] DEB, K., AND SINHA, A. An Efficient and Accurate Solution Methodology for Bilevel Multi-Objective Programming Problems Using a Hybrid Evolutionary-Local-Search Algorithm. *dx.doi.org 18*, 3 (Aug. 2010), 403–449.

[22] DESAI, S., BAHADURE, S., KAZI, F., AND SINGH, N. Article: Multi-Objective Constrained Optimization using Discrete Mechanics and NSGA-II Approach. *International Journal of Computer Applications 57*, 20 (2012), 14–20.

[23] DUA, R., RAJA, A. R., AND KAKADIA, D. Virtualization vs Containerization to Support PaaS. In *2014 IEEE International Conference on Cloud Engineering (IC2E)* (2014), IEEE, pp. 610–614.

[24] FARAHNAKIAN, F., PAHIKKALA, T., LILJEBERG, P., PLOSILA, J., AND TENHUNEN, H. Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing. *CLOUD* (2015).

[25] FELTER, W., FERREIRA, A., RAJAMONY, R., AND RUBIO, J. An updated performance comparison of virtual machines and Linux containers. *ISPASS* (2015), 171–172.

[26] FERDAUS, M. H., MURSHED, M. M., CALHEIROS, R. N., AND BUYYA, R. Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic. *Euro-Par 8632*, Chapter 26 (2014), 306–317.

[27] FORSMAN, M., GLAD, A., LUNDBERG, L., AND ILIE, D. Algorithms for automated live migration of virtual machines. *Journal of Systems and Software 101* (2015), 110–126.

[28] GAO, Y., GUAN, H., QI, Z., HOU, Y., AND LIU, L. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *J. Comput. Syst. Sci. 79*, 8 (2013), 1230–1242.

[29] Ghodsi, A., Zaharia, M., Hindman, B., and Konwinski, A. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. *NSDI* (2011).

[30] Grimes, D., Mehta, D., OSullivan, B., Birke, R., Chen, L., Scherer, T., and Castineiras, I. Robust Server Consolidation: Coping with Peak Demand Underestimation. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)* (2016), IEEE, pp. 271–276.

[31] Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q. M., Tziritas, N., Vishnu, A., Khan, S. U., and Zomaya, A. Y. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing 98*, 7 (2016), 751–774.

[32] He, S., Guo, L., Guo, Y., Wu, C., Ghanem, M., and Han, R. Elastic Application Container: A Lightweight Approach for Cloud Resource Provisioning. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications (AINA)* (Feb. 2012), IEEE, pp. 15–22.

[33] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R. H., Shenker, S., and Stoica, I. Mesos - A Platform for Fine-Grained Resource Sharing in the Data Center. *NSDI* (2011).

[34] Hines, M. R., Deshpande, U., and Gopalan, K. Post-copy live migration of virtual machines. *SIGOPS Oper. Syst. Rev. 43*, 3 (July 2009), 14–26.

[35] Jennings, B., and Stadler, R. Resource Management in Clouds: Survey and Research Challenges. *Journal of Network and Systems Management 23*, 3 (2015), 567–619.

[36] Jung, G., Hiltunen, M. A., Joshi, K. R., Schlichting, R. D., and Pu, C. Mistral - Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. *ICDCS* (2010), 62–73.

[37] Kanagarajan, D., Karthikeyan, R., Palanikumar, K., and Davim, J. P. Optimization of electrical discharge machining characteristics of WC/Co composites using non-dominated sorting genetic algorithm (NSGA-II). *The International Journal of Advanced Manufacturing Technology 36*, 11-12 (2008), 1124–1132.

[38] Kaplan, J. M., Forrest, W., and Kindler, N. Revolutionizing data center energy efficiency, 2008.

[39] Khanna, G., Beaty, K. A., Kar, G., and Kochut, A. Application Performance Management in Virtualized Server Environments. *NOMS* (2006).

[40] Kivity, A., Kamay, Y., Laor, D., and Lublin, U. kvm: the Linux virtual machine monitor. *... of the Linux ...* (2007).

[41] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. Power and performance management of virtualized computing environments via lookahead control. *Cluster computing 12*, 1 (2009), 1–15.

[42] Lee, K. Y., and Yang, F. F. Optimal reactive power planning using evolutionary algorithms: A comparative study for evolutionary programming, evolutionary strategy, genetic algorithm, and linear programming. *IEEE Transactions on power systems 13*, 1 (1998), 101–108.

[43] LEGILLON, F., LIEFOOGHE, A., AND TALBI, E.-G. CoBRA - A cooperative coevolutionary algorithm for bi-level optimization. *IEEE Congress on Evolutionary Computation* (2012), 1–8.

[44] LI, X., TIAN, P., AND MIN, X. A Hierarchical Particle Swarm Optimization for Solving Bilevel Programming Problems. In *Service-Oriented and Cloud Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1169–1178.

[45] MAGALHÃES, D., CALHEIROS, R. N., BUYYA, R., AND GOMES, D. G. Workload modeling for resource usage analysis and simulation in cloud computing. *Computers & Electrical Engineering 47* (2015), 69–81.

[46] MANN, Z. Á. Approximability of virtual machine allocation: much harder than bin packing.

[47] MANN, Z. Á. Interplay of Virtual Machine Selection and Virtual Machine Placement. In *Service-Oriented and Cloud Computing*. Springer International Publishing, Cham, Aug. 2016, pp. 137–151.

[48] MELL, P. M., AND GRANCE, T. The NIST definition of cloud computing. Tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, Gaithersburg, MD, 2011.

[49] MISHRA, M., DAS, A., KULKARNI, P., AND SAHOO, A. Dynamic resource management using virtual machine migrations. *IEEE Communications . . . 50*, 9 (2012), 34–40.

[50] MOENS, H., FAMAEY, J., LATRÉ, S., DHOEDT, B., AND DE TURCK, F. Design and evaluation of a hierarchical application placement algorithm in large scale clouds. *Integrated Network Management* (2011), 137–144.

[51] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Automatic Design of Scheduling Policies for Dynamic Multi-objective Job Shop Scheduling via Cooperative Coevolution Genetic Programming. *IEEE Transactions on Evolutionary Computation 18*, 2 (2014), 193–208.

[52] PANIGRAHY, R., TALWAR, K., UYEDA, L., AND WIEDER, U. Heuristics for vector bin packing. *research microsoft com* (2011).

[53] PASCHKE, A., AND SCHNAPPINGER-GERULL, E. A Categorization Scheme for SLA Metrics. *Service Oriented Electronic Commerce 80*, 25-40 (2006), 14.

[54] PIRAGHAJ, S. F., CALHEIROS, R. N., CHAN, J., DASTJERDI, A. V., AND BUYYA, R. Virtual Machine Customization and Task Mapping Architecture for Efficient Allocation of Cloud Data Center Resources. *Comput. J. 59*, 2 (2016), 208–224.

[55] PIRAGHAJ, S. F., DASTJERDI, A. V., AND CALHEIROS, R. N. A survey and taxonomy of energy efficient resource management techniques in platform as a service cloud. *. . . of Research on End-to . . .* (2017).

[56] PIRAGHAJ, S. F., DASTJERDI, A. V., CALHEIROS, R. N., AND BUYYA, R. Efficient Virtual Machine Sizing for Hosting Containers as a Service (SERVICES 2015). *SERVICES* (2015).

[57] RAGHAVENDRA, R., RANGANATHAN, P., TALWAR, V., WANG, Z., AND ZHU, X. No power struggles: Coordinated multi-level power management for the data center. In *ACM SIGARCH Computer Architecture News* (2008), ACM, pp. 48–59.

[58] RONG, H., ZHANG, H., XIAO, S., LI, C., AND HU, C. Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews 58* (May 2016), 674–691.

[59] ROSEN, R. Resource management: Linux kernel namespaces and cgroups. *Haifux* (2013).

[60] SARIN, S. C., VARADARAJAN, A., AND WANG, L. A survey of dispatching rules for operational control in wafer fabrication. *Production Planning and ... 22*, 1 (Jan. 2011), 4–24.

[61] SHAFER, J. I/O virtualization bottlenecks in cloud computing today. In *Proceedings of the 2nd conference on I/O virtualization* (2010).

[62] SHEN, S., VAN BEEK, V., AND IOSUP, A. Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters. *CCGRID* (2015), 465–474.

[63] SHI, W., AND HONG, B. Towards Profitable Virtual Machine Placement in the Data Center. *UCC* (2011), 138–145.

[64] SINHA, A., MALO, P., AND DEB, K. A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. *IEEE Transactions on Evolutionary Computation* (2017), 1–1.

[65] SOLTESZ, S., PÖTZL, H., FIUCZYNSKI, M. E., BAVIER, A. C., AND PETERSON, L. L. Container-based operating system virtualization - a scalable, high-performance alternative to hypervisors. *EuroSys 41*, 3 (2007), 275–287.

[66] SOMANI, G., AND CHAUDHARY, S. Application Performance Isolation in Virtualization. *IEEE CLOUD* (2009), 41–48.

[67] SOTELO-FIGUEROA, M. A., SOBERANES, H. J. P., CARPIO, J. M., HUACUJA, H. J. F., REYES, L. C., AND SORIA-ALCARAZ, J. A. Evolving Bin Packing Heuristic Using Micro-Differential Evolution with Indirect Representation. *Recent Advances on Hybrid Intelligent Systems 451*, Chapter 28 (2013), 349–359.

[68] SRIKANTAIAH, S., KANSAL, A., AND ZHAO, F. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems* (2008), San Diego, California, pp. 1–5.

[69] SVARD, P., LI, W., WADBRO, E., TORDSSON, J., AND ELMROTH, E. Continuous Datacenter Consolidation. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)* (2015), IEEE, pp. 387–396.

[70] TAKOUNA, I., ALZAGHOUL, E., AND MEINEL, C. Robust Virtual Machine Consolidation for Efficient Energy and Performance in Virtualized Data Centers. In *2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing(CPSCom)* (2014), IEEE, pp. 470–477.

[71] TAN, B., MA, H., AND MEI, Y. A NSGA-II-based approach for service resource allocation in Cloud. *CEC* (2017), 2574–2581.

[72] TAN, J., DUBE, P., MENG, X., AND 0002, L. Z. Exploiting Resource Usage Patterns for Better Utilization Prediction. *ICDCS Workshops* (2011), 14–19.

[73] TOMÁS, L., AND TORDSSON, J. Improving cloud infrastructure utilization through overbooking. *CAC* (2013), 1.

[74] UHLIG, R., NEIGER, G., RODGERS, D., SANTONI, A. L., MARTINS, F. C. M., ANDERSON, A. V., BENNETT, S. M., KÄGI, A., LEUNG, F. H., AND SMITH, L. Intel Virtualization Technology. *IEEE Computer 38*, 5 (2005), 48–56.

[75] UHLIG, R., NEIGER, G., RODGERS, D., SANTONI, A. L., MARTINS, F. C. M., ANDERSON, A. V., BENNETT, S. M., KAGI, A., LEUNG, F. H., AND SMITH, L. Intel virtualization technology. *Computer 38*, 5 (May 2005), 48–56.

[76] VARASTEH, A., AND GOUDARZI, M. Server Consolidation Techniques in Virtualized Data Centers: A Survey. *IEEE Systems Journal* (2015), 1–12.

[77] VERMA, A., AND DASGUPTA, G. Server Workload Analysis for Power Minimization using Consolidation. *USENIX Annual Technical Conference* (2009), 28–28.

[78] VICENTE, L., SAVARD, G., AND JÚDICE, J. Descent approaches for quadratic bilevel programming. *Journal of Optimization Theory and Applications 81*, 2 (May 1994), 379–399.

[79] WALDSPURGER, C. A. Memory resource management in VMware ESX server. *ACM SIGOPS Operating Systems Review 36*, SI (Dec. 2002), 181–194.

[80] WANG, G. G., AND SHAN, S. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design 129*, 4 (Apr. 2007), 370–380.

[81] XAVIER, M. G., NEVES, M. V., ROSSI, F. D., FERRETO, T. C., LANGE, T., AND DE ROSE, C. A. F. Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments. In *2013 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2013)* (2013), IEEE, pp. 233–240.

[82] XIAO, Z., JIANG, J., ZHU, Y., MING, Z., ZHONG, S.-H., AND CAI, S.-B. A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory. *Journal of Systems and Software 101* (2015), 260–272.

[83] XU, J., AND FORTES, J. A. B. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. In *Int'l Conference on Cyber, Physical and Social Computing (CPSCom)* (2010), IEEE, pp. 179–188.

[84] YAU, S. S., AND AN, H. G. Adaptive resource allocation for service-based systems. In *Proceedings of the First Asia-Pacific Symposium on Internetware* (2009), ACM, p. 3.

[85] YAZIR, Y. O., MATTHEWS, C., FARAHBOD, R., NEVILLE, S. W., GUITOUNI, A., GANTI, S., AND COADY, Y. Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis. *IEEE CLOUD* (2010), 91–98.

[86] YIN, Y. Genetic-algorithms-based approach for bilevel programming models. *Journal of Transportation Engineering* (2000).

[87] YUSOH, Z. I. M., AND TANG, M. A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud. In *Evolutionary Computation (CEC), 2010 IEEE Congress on* (2010), IEEE, pp. 1–8.

[88] ZHANG, J., HUANG, H., AND WANG, X. Resource provision algorithms in cloud computing - A survey. *J. Network and Computer Applications 64* (2016), 23–42.

[89] ZHANG, Q., CHENG, L., AND BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications 1*, 1 (2010), 7–18.

[90] ZHANG, Y., ZHENG, Z., AND LYU, M. R. Exploring Latent Features for Memory-Based QoS Prediction in Cloud Computing. In *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on* (2011), pp. 1–10.

[91] ZHU, X., YU, Q., AND WANG, X. A Hybrid Differential Evolution Algorithm for Solving Nonlinear Bilevel Programming with Linear Constraints. In *2006 5th IEEE International Conference on Cognitive Informatics* (2006), IEEE, pp. 126–131.