VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Energy-efficient Server Consolidation in Container-based Clouds with EC approaches

Boxiong Tan

Supervisors: Dr.Hui Ma, Dr.Yi Mei, Prof. Mengjie Zhang

Submitted in partial fulfilment of the requirements for PhD.

Abstract

Container-based Cloud is a new trend in Cloud computing. Compare to virtual machine (VM)-based Cloud, containers provide a new architecture for allocating applications and a finer resource management which has the potential to further improve the energy consumption in data centers. Current VM-based server consolidation strategies cannot be used in container-based cloud because the container-based cloud has two levels of placement: container to VM and VM to PM. Existing research lacks energy model and optimization algorithms that consider the joint allocation of container and VM. This work aims at reducing energy consumption by proposing a bilevel energy model and Evolutionary Computation (EC)-based optimization algorithms. We resolve the server consolidation problems on all three placement decision scenarios: application initial placement, periodic optimization, and dynamic placement. The proposed research aims to improve the energy consumption in container-based Cloud with various types of workload. A novel energy model and three optimization algorithms are proposed for three placement decision scenarios.

Contents

Figures

Chapter 1

Introduction

This chapter introduces this research proposal. It starts with the problem statement, then outlines the motivations, research goals and the organisation of this proposal.

1.1 Problem statement

Cloud computing has made a huge impact on modern software industry by offering on-demand computing capacity (e.g storage and computing) [18]. Compare with traditional software industry, web service providers such as Google and Neflix do not need to build data centers and manage hardware resources. They can develop, deploy and maintain applications (e.g. google drive) on cloud [2] without worrying about scalability issue (e.g. dynamically increases the capacity of application) and availability of applications (e.g services are accessible 99.99% of the time). For application users, they can enjoy applications without experiencing breakdown and access the applications from anywhere in the world.

A major issue in cloud computing is the huge energy consumption generated by data centers - a typical data center consumes as much energy as 25,000 households [27]. Huge energy consumption has become the major expense of cloud providers. The reduction of energy bill will be further beneficial to the profit in software industry as well as most people who access the Internet on a daily basis with a lower cost.

Generally, reducing the energy consumption of a data center can be achieved by reducing the number of live physical machines (PMs) (e.g. servers). Studies shows [6, 87], among several energy consuming components such as cooling systems, PMs, and network devices, PMs account for the majority - more than 40% - of energy consumption while these PMs are not used effectively. The proportion of PMs' average utilization is quite low - from 10% to 50% due to some disadvantages in resource management. Therefore, it is possible to reduce energy by improving the utilization of PMs.

The common way to improve the utilization of PMs of a data center is through resource management of PMs [66] (see Figure 1.1). The cloud resource management of PMs is a centralized system [48] that allocates resources such as CPUs and memories of PMs to cloud users' applications and handles the workload fluctuations. Four major steps in resource management are listed as follows: Collecting utilization data from PMs, analyzing available resources on PMs, deciding the placement of applications on PMs, and executing the application placement decisions. As better allocation of CPUs and memories leads to the reduction of energy consumption, the placement decision is the most important step.

Placement decision is applied in three scenarios to improve the utilization of PMs. Three placement decision scenarios are common in a data center: application initial placement handles new arrival of applications; periodic optimization adjusts the placement globally

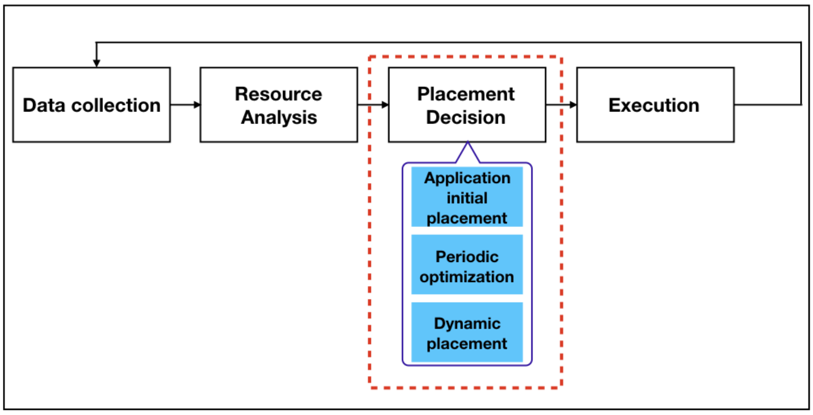


Figure 1.1: A workflow of resource management [71]

and periodically; dynamic placement adjusts application in a fast manner. All three decision scenarios rely on an optimization strategy called server consolidation.

Server consolidation [101] is a strategy to manage the resources of PMs by placing applications in fewer PMs and lead to lower energy consumption. For three resource management scenarios, they are generally considered as two types of problem: static and dynamic. Static problem is solved in an off-line fashion which includes application initial placement and periodic optimization, while dynamic placement problem is solved in an on-line fashion. In order to solve static and dynamic problems, server consolidation strategy must also be designed in static and dynamic fashions.

Currently, resource management in data centers is based on *virtualization* technology [99]. Such virtualization separates the resources (e.g. CPUs and RAMs) of a PM into several parts called *virtual machines (VMs)*, each of VMs runs an isolated operating system. Compare with virtualization, the traditional data center assigns a PM for each application; it leads to the low reserved utilization of PMs. After the technique of VM being introduced, the multiplexing of PMs largely improve the utilization and reduce the energy consumption.

However, in recent years, resource management with VMs cannot catch up with a new trend in software industry - Service Oriented Architecture (SOA) [93]; SOA has become widely used in modern software industry because of its agility and re-usability [93]. SOA separates a centralized application into multiple distributed components called web services. As most of web services only require a small amount of resources, using a VM for a web service causes resource wastage inside a VM. Consequently, the low utilization of PMs decreases the energy efficiency. To support the architecture of SOA and further reduce energy consumption, a new virtualization technology: containers [38, 91] has been proposed. The container technology provides a new architecture for allocating applications and a finer resource management which has the potential to further improve the energy consumption.

A container is an operating system (OS) level of virtualization which runs on top of VMs. Similar to VM, a container provides performance and resource isolation for a single application. Different to VMs, multiple containers can run in the same VM without interfering each other. In addition, containers naturally support vertical scaling (change its size during runtime) [100] which is resilient to fluctuate workloads. However, vertical scaling requires the VMs to reserve enough resource to support the increasing size of container.

Although the efficient use of containers can improve the utilization of VMs, it brings new challenges and difficulties to server consolidation [1]. Current VM-based server consolidation strategy cannot be directly on container-based cloud because of the different application placement structure. This research aims at improving the utilization of PMs in container-based cloud by proposing new bilevel energy models and server consolidation algorithms for three placement decision scenarios: application initial placement, periodic optimization, and dynamic placement.

1.2 Motivation

This section identifies the research gaps in resource management of container-based data centers. We will discuss the gaps from three placement decision scenarios: application initial placement, periodic optimization, and dynamic placement.

1.2.1 Application initial placement

Application initial placement deploys applications when data centers receive a number of requests. The placement can be seen as a static server consolidation problem. To reduce energy, the strategy allocates applications to a minimum number of physical machines (PMs). For VM-based and container-based cloud, the energy models are different. In a VM-base Cloud, the energy model of application initial placement can be seen as a vector bin packing problem [60], applications (wrapped with VMs) are packed in PMs (detailed discussion is in Section 2.2.2).

In contrast, in a container-based Cloud, the energy model can be seen as a bilevel optimization problem [24] where each level is treated as a bin packing problem. The lower level optimizes the placement of containers to VMs and the upper level optimizes the placement of VMs to PMs. The advantage the bilevel model is that the interaction of container-VM and VM-PM placement is considered, so that the global optimal can be achieved.

Two reasons motivate us to solve the application initial placement in container-based Cloud. First, currently, no research has considered the application initial placement as a bilevel optimization problem. Therefore, it needs to propose a new bilevel energy model. A bilevel model - includes energy model, workload model and prices model - represents the relationship between container, VMs, and energy consumption. Current VM-based models cannot be directly applied because container-based model has two levels of placement. In addition, current VM-based models do not consider the overhead of VM hypervisor because for VM-based cloud there is no better ways to avoid the overhead. However, in container-based cloud, the overhead of VMs can be mitigated by reducing the number of VMs. This can be achieved by consolidating containers to fewer VMs. Furthermore, many VM-based models do not consider a balance between CPUs and memories. The balance is crucial [98] in improving utilization of PMs, because the balance in a PM increases the probability of being able to allocating a new application.

Second, bilevel optimization is known to be strongly NP-hard [89]. Even in the simplest case of linear bilevel programs, where the lower level problem has a unique optimal solution for all the parameters, it is not likely to find a polynomial algorithm that can find the global optimum solution. The proof for the non-existence of a polynomial time algorithm for linear bilevel problems can be found in [32]. In contrast, evolutionary computation (EC) is a population-based search mechanism which has been proposed to solve bilevel optimization problems [3,105,108]. EC algorithms have shown promising performance bilevel problems. Therefore, we will investigate EC-based approaches on bilevel problem.

1.2.2 Periodic optimization

After initial placement, periodic optimization is a routine process that takes existing applications' placement, re-placing them to PMs to optimize the energy consumption. A technology called live migration can be used to re-place the applications' placement from one PM to another. Live migrations are very expensive since they consume network bandwidth and use the resources on both host PM and targeted PM. Therefore, periodic optimization is a multi-objective task which considers minimizing migration of applications and minimizing the overall energy consumption. Resolving the bi-level multi-objective problem will lead to a high utilization of PMs but it is very challenging.

Two reasons motivate us to explore solutions for periodic optimization in a container-based cloud. **First**, no research has considered the periodic optimization in a container-based cloud as a bilevel multi-objective optimization problem. The multi-objective bilevel problem has two potentially conflicting objectives: reducing the number of migration and minimizing the energy consumption. **Second**, not many research have considered the robustness of placement. The robustness of placement means the placement is resilient enough to handle the fluctuate workloads without making too many further adjustments. To achieve robustness of placement, periodic optimization must consider the combination of various workloads and the reserved resources on PMs. Currently, most research simplified workloads as static (remains a constant value throughout its life cycle) [21, 37, 103] for the sake of simplicity. The placement is easy to affect by fluctuations which leads to higher energy consumption. Therefore, we will consider various types of workload to improve the robustness of placement.

1.2.3 Dynamic placement

Dynamic placement is applied on overloading and underloading scenarios which need an immediate reaction on the placement of an application [9]. Overloadding and underloading happened when applications are released from PMs or applications are facing a burst of workload. In overloading scenario, PMs are running out of resources which means the applications' performance degraded. The degradation will bring financial punishment for Cloud providers. In underloading scenario, PMs are running in a low utilization which leads to the waste of energy. At these states, it is ideal that applications inside the PM will be placed to other PMs to quickly resolve the problems.

Three reasons motivates us to explore solutions for dynamic placement. **Firstly**, no research has considered dynamic placement in container-based cloud as a bilevel problem. Current approaches [79] consider two placements as separate tasks. **Secondly**, current VM-based dynamic placement approaches typically applied either simple bin-packing algorithms such as First Fit or manually designed heuristics. Simple bin-packing algorithms may perform poorly because application placement is more complicated than bin-packing [64]. Therefore, even for the placement of VMs to PM, it is difficult to reach a global optimal solution. **Thirdly**, manually designed heuristics may not be general (in terms of decision variables and constraints) because they are designed for specific conditions and constraints [50], e.g considered the network topology in data centers. We want to develop a hyper-heuristic approach which can automatically generate good heuristics based the knowledge learned from previous placement solutions.

In summary, shortcomings of current resource management in container-based cloud increase the energy consumption of a cloud data center. Therefore, our goal is to reduce the energy consumption by overcoming these limitations in each of placement decision scenario.

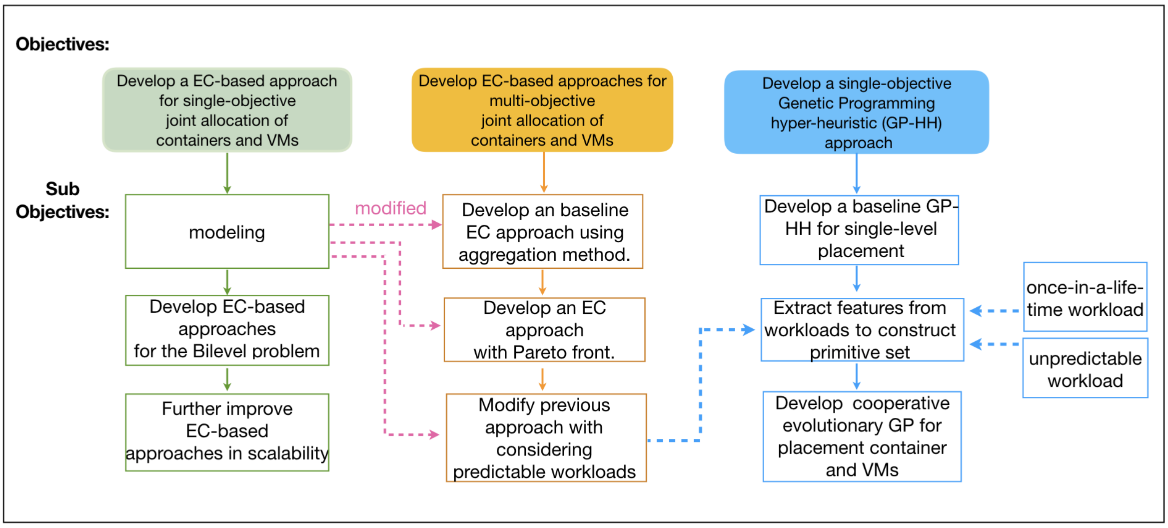


Figure 1.2: Relationship between objectives

1.3 Research Goals

The overall goal of this research is to optimize energy consumption of a container-based Cloud data center using EC-based approaches for three placement decision scenarios: application initial placement, periodic optimization, and dynamic placement. The specific research objectives of this work can be itemized as follows.

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

The goal is to reduce the energy consumption in application initial placement considering container-based cloud data center. To achieve this goal, the first step is to propose a new bilevel model for the joint placement of containers and VMs problem. We will explore evolutionary computation based approaches to solve the bievel problem (For the sake of simplicity, we will use "the bilevel model/problem" to replace "the joint placement of containers and VMs model/problem" in the following content.). The research goal leads to three objectives as follows.

1. Develop a new bilevel model to capture the relationship between containers allocation and energy consumption in container-based cloud data center. The goal of the first sub objective is to propose a bilevel model for the joint placement of container and VM.

   The major challenge is that no previous research considers the joint placement container and VM as a bilevel problem while the relationship between container, VM and energy consumption is unclear. Specifically, three issues remain unsolved. The first issue is that it is still unclear that which energy function is the best to capture the relationship between container and VM so that the overall energy is low. Specifically, the objective for the lower level - placing container to VM, is still unclear. This is because the minimum number of VM does not necessary lead to the minimum number PMs; the types of VM also play an important role. The second issue is that previous VM-based research do not consider the overhead of VM. However, the overhead of VMs is a major source of resource wastage (addressed in Section 2.2.2). Therefore, how to represent the impact of VMs remains unsolved. The third issue is related to a VM-based research, Mishra [72] discovers that when multiple resources are considered in the model, the balance between resources has a heavy impact on the optimization results. Therefore, in the bilevel model, the balance of resources should also be considered.

In order to establish a bilevel model, 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. We will start from the simplest case - single dimension of resource - to more general multi-dimensional resources model by reviewing a number of VM-based approaches. Specifically, we focus on their variables, constraints and objective function. Objective function is mainly related to energy consumption. Hence, energy model is another major issue to study. In addition, in the multi-dimensional resource model, we will address the balance of CPU and memory problem by investigating several resource wastage models [39, 43, 116]. In this objective, we consider the static workload of applications, this is because the initial resource demand is often provided by the Cloud users.

2. Propose a new EC based bilevel optimization approach to solve the application initial placement.
   Based on the proposed bilevel model, the goal of this sub-objective is to develop an approach for the bilevel optimization problem using nested Evolutionary algorithms [90].

   Three challenges need to be solved. First one is to understand the interaction between bilevel's placement. In the bilevel problem, placing containers into a minimum number of VMs does not necessary lead to the minimum energy consumption. Therefore, it is still unclear that relation among the selection type of VM, placement of container and placement of VM will affect the energy consumption. Second challenge is how to design the search operators and representation. Currently two types of representation: direct and indirect representation can be considered. However, it is unclear that which one is more suitable for the nature of the bilevel problem. Third, bilevel optimization is strongly NP-hard [68], the solution space can be non-linearity, discreteness, no-differentiability, and non-convexity. Therefore, it is extremely difficult to design a proper search mechanism to find near optimal solutions.

   In order to discover the relation among the selection type of VM, placement of container and placement of VM, we will first use one type of VMs and one type of container. By controlling these variables, the effect of different types of VMs and containers will be eliminated. Therefore, the relationship between bilevel placement would be clear. We will gradually add up variables and constraints. For the representation of bilevel problem, we will develop direct binary representation [116], and indirect continuous probability representation [115]. Genetic operators are also designed along with the proposed representation. Current nested methods have been used in solving bilevel problem, however, there is no research focus on bilevel bin-packing problem. We will investigate several approaches such as Nested Particle Swarm Optimization [61], Differential evolution (DE) based approach [3, 121] and Co-evolutionary approach [59].

3. Investigate methods to improve the scalability of the EC-based bilevel optimization approach.

   Based on proposed EC-based approach, the goal of this sub-objective is to improve scalability of the approach. Although nested approaches have been reported effective, they are very time consuming [90]. Therefore, this sub objective intends to explore other directions to improve the execution time.

Three approaches can be potentially used in improving the scalability. The first one is single-level reduction [90], which reduces the bilevel problem into a single dimensional problem. Containers can be categorized into VMs which is then placed into PMs. The combination of container must be based on the knowledge of two-level placement interaction which we discover in the previous objective. Clustering approaches such as K-means [114] or decision tree can be useful in categorizing containers. Then, complementary containers can be grouped to reduce the variables of placement. The challenge is to identify the features of static workload so that different workloads can be combined to fill a VM. Another way is use reinforcement learning to learn the pattern of energy-efficient combination of containers. The second approach is using a divide and conquer method to split the large number of containers into smaller chunks. The main challenge is that how to split the problem is unknown. Randomly dividing is very likely lead to a sub-optimal solution. The third approach is combine heuristics into the EC algorithm, for example, develop a representation which is embedded with a simple heuristic (e.g First Fit). The heuristic is expected to reduce the search space so that the EC algorithm can find solution more efficiently. However, design a heuristic which embedded inside an EC algorithm is extreme difficult since evaluation of heuristic is indirect.

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

The goal is to develop multi-objective EC-base approaches for container-based cloud in periodic optimization with considering various types of workload to reduce the overall energy consumption.

1. Modify the proposed model to adapt to the multi-objective problem with various types of workload.
   The goal of this sub-objective is to modify previous proposed bilevel model so that it adapts to the multi-objective problem.

   There are mainly two challenges, the first one is to add an migration model to the existing model, and the second is to adapt the model to various types of workload. The migration model is distinct with VM-based model. Because both container and VM can be migrated, it is unclear that migration model should be added to both layer or just one. One possible solution is to represent all VM migration with container migration. It may reduce the complexity. In addition, majority traditional migration models only consider the migration number without including the size of the applications which is unrealistic. Because the size will affect the overhead on the networking. The second challenge is to adapt the model to various types of workload. Previously, we simply the problem as applications can be represented as static workloads. In this sub objective, we consider three types of predictable types of workload: static, linear continuous changing and periodic workload. These workloads may be represented as a function of time. Other models may also be changed accordingly.

   We will further develop an EC-based multi-objective algorithm with aggregation approach to test the model. The aggregation approach turns a multi-objective problem into a single-objective problem by combining objectives into a single one. Therefore, we may use previous developed algorithm to solve periodic optimization problem. We will use static workload to test the model.

2. Propose an EC-based multi-objective algorithm for periodic optimization with Pareto front approach.

The goal of this sub-objective is to develop an EC-based approach to solve the multi-objective joint allocation problem with Pareto front approach.

The major challenge in this sub-objective is to design genetic operators so that the proposed algorithm can steer the search close to the correct Pareto front. The aggregation approach proposed in previous sub objective has some defects such as it cannot find the non-convex solution. A Pareto front approach is able to find a set of trade-off between objectives, therefore, we decide to explore this direction. Currently only a few research [28, 30, 118] focus on bilevel optimization problem. This will be the first time that bilevel optimization with Pareto front approach is applied on a bilevel bin packing problem.

3. Propose an EC-based multi-objective algorithm for periodic optimization considering various types of predictable workload.
The goal of this sub objective is to propose an approach for three predictable workloads [36]: static, linear continuously changing, and periodic. The major challenge for this objective is that the change of problem model from static to changing workload may lead to a different representation. In order to achieve using a uniform representation for various workload, we are going to explore various workload. Accordingly, new search mechanisms must be proposed to adapt to the representation.

1.3.3 Objective Three: Develop a hyper-heuristic single-objective Cooperative Genetic Programming (GP) approach for automatically generating dispatching rules for dynamic placement.

The goal for this objective is to develop a cooperative GP-based hyper-heuristic algorithm so that the generated dispatching rules can achieve both fast placement and global optimization with various workloads.

1. Develop a GP-based hyper-heuristic (GP-HH) algorithm for the placement of container to VM.
In order to develop a cooperative GP-based hyper-heuristic to the bilevel problem, it is necessary to develop a GP-HH for the single level of the problem. Therefore, the goal of this sub-objective is to develop a GP-HH algorithm for placing containers to VMs. This task is none trivial since no GP-HH has been dynamic placement problem. Therefore, we may start from considering the features such as the status of VMs (e.g resource utilization), features of workloads (e.g resource requirement) that will affect the placement decision. We will construct primitive set with the selected features. Other unsolved issues are the functional set and search mechanism. We will use the functional set by using the general operators. The original genetic programming will be used as the search mechanism.

   To train the GP-based hyper-heuristic, we will use the solutions in the first objective as the model solution.

   In order to evaluate the automatically generated heuristics. We will use a widely used simulator called CloudSim [19]. Since our proposed algorithm is focus on one level of placement, it is equivalent to the VM-based placement problem. We will compare our heuristic to a highly cited work [7] from Beloglazov who propose a Best Fit Decreasing heuristic for the energy consumption problem.

2. Conduct feature extraction on the predictable workloads and unpredictable workloads.

The goal of this sub-objective is to construct a GP primitive set by applying feature extraction on various types of application workload. In previous sub-objective, we develop a baseline GP-HH on static workload. In order to develop a general GP-HH that can handle all kinds of workloads, we will extract features from predictable workloads such as linear continuous changing workloads, periodic workloads, and from unpredictable workloads: once-in-a-lifetime workloads.

The first challenge is to find suitable representation for workloads. Currently, representation time-series are classified into three categories: temporal, spectral and others. It is still unclear which pattern extraction technique and representation that is best for workload data. The second challenge the high dimensionality of dataset which requires a dimensionality reduction technique to reduce the number of data point. Some possible techniques are sampling, extrema extraction.

We will test the extracted features by applying classification on the training and test set. The final features will be used in the primitive set.

3. Develop a Cooperative GP-HH approach to evolve dispatching rules for placing container and VMs.
The goal of this sub-objective is to develop a cooperative GP approach to evolve dispatching rules. In the baseline approach, we develop a GP-HH approach for single-level of placement. However, there is a case that no current VM is suitable for a container to place in; a new VM is needed to place at this moment. This case incurs a second level of placement.

   Therefore, to construct a complete placement dispatching rule, we will develop a cooperative GP-HH approach to solve the two-level of placement problem. We may reuse the single-level GP-HH in both level or develop a new GP-HH in the VMs to PMs level.

1.4 Published Papers

During the initial stage of this research, some investigation was carried out on the model of container-based server consolidation [97].

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 2 provides a fundamental background of the resource management in cloud data center and its the energy consumption problem. It also performs a literature review covering a range of works in this field; Chapter 3 discusses the preliminary work carried out to explore the techniques and EC-based techniques for the joint allocation of container and VMs; Chapter 4 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 brief introduction of Cloud computing and the energy consumption problem in cloud data centers. Section 2.2 introduces resource management in cloud including the placement decision scenarios (Section 2.2.1) and the main strategy - server consolidation (Section 2.2.2).The second part (Section 2.3) gives a literature review on resource management in both container and VM-based cloud.

2.1 Cloud Computing Background

Cloud computing is a computing model offers a network of servers to their clients in a on-demand fashion. From NIST's definition [69], *"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."*

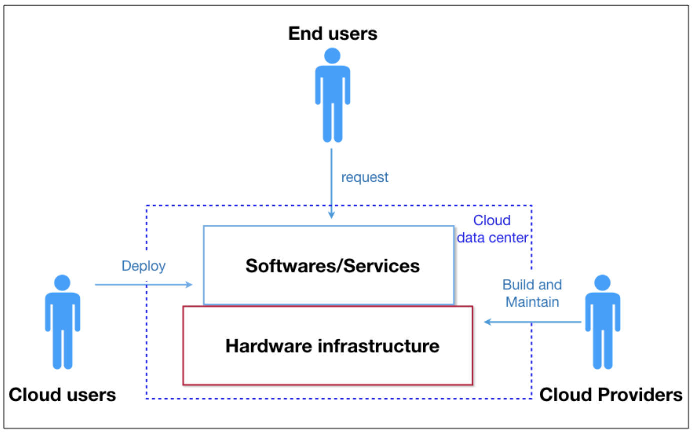


Figure 2.1: Stakeholders of Cloud computing [48]

To give an example of how Cloud computing works (see Figure 2.1, considering the case: a *Cloud provider* builds a data center which contains thousands of servers connected with network. A web-based application provider is also a *Cloud user*, can deploy and access their applications (e.g Endnote, Google Drive and etc.) in these servers from anywhere in the world. Once the applications start serving, *End users* can use them without installing on their local computers. Cloud providers charge fees from Cloud users for infrastructure and Cloud users charge fees from End users for using their services. Therefore, from cloud

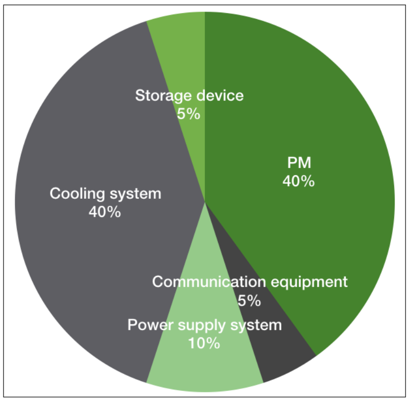


Figure 2.2: Energy consumption distribution of data centers [84]

providers' perspective, they can increase their profit by accommodating more applications and reducing the expense of data centers.

The major expense of data centers is energy consumption [51]. Energy consumption is derived from several parts as illustrated in Figure 2.2. Cooling system and physical machines (PMs) account for a majority of the consumption. A recent survey [22] shows that the recent development of cooling techniques have reduced its energy consumption and now PM consumption has become the dominate energy consumption component.

According to Hameed et al [44], PMs are far from energy-efficient. The main reason for the wastage of energy is that the energy consumption of servers remains high even when the utilization of PMs are low. For example, the average CPU utilization of a data center is only 15% for most of the time, however, the energy consumption of PM remains 70% of their peak time. The major reason is that each application is allocated to PMs for the sake of application environment isolation while the resources in the PM are far more than the application needs.

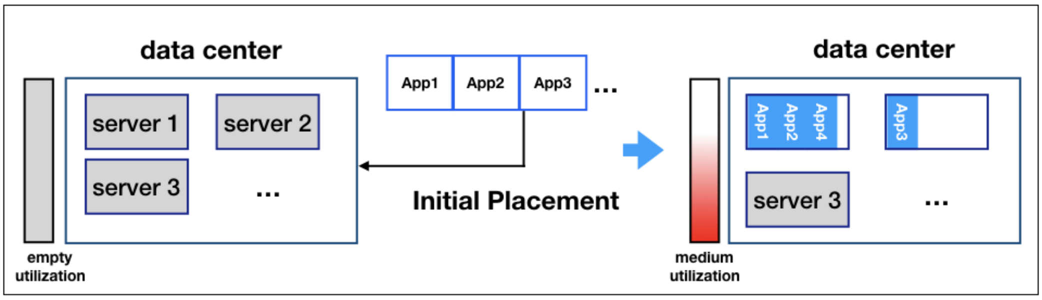
In order to solve the low utilization of PMs, a concept of *energy proportional computing* [6] raised to address the disproportionate between utilization and energy consumption by using resource management.

2.2 Cloud Resource Management

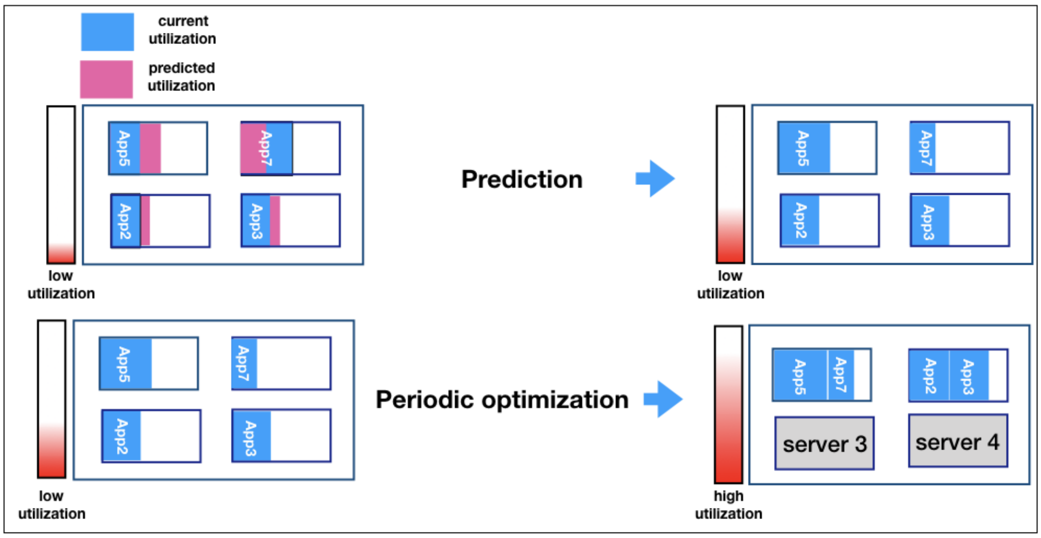
Cloud resource management is a process of allocating computing, storage, networking and indirectly energy resources to a set of applications, to meet the objectives of the *End users*, *Cloud users* and *Cloud providers* [48]. The objectives of cloud provider are efficient and effective resource use, specifically, the high utilization of PMs and energy. To achieve this goal, a strategy of server consolidation is used to enable multiplexing of PMs' resources across applications. Server consolidation is typically implemented through virtualization technologies. This section will first discusses the server consolidation strategy. Then two types of virtualization technologies are introduced and compared. Finally, the research gaps in three placement decision scenarios will be discussed.

2.2.1 Placement Decision Scenarios

The server consolidation in data center can be applied to three placement decision scenarios [71, 96]: Application initial placement, periodic consolidation, and dynamic placement (see Figure 2.3).



(a) Application Initial Placement



(b) Periodic Optimization

(c) Dynamic Placement

Figure 2.3: Three scenarios of placement decision

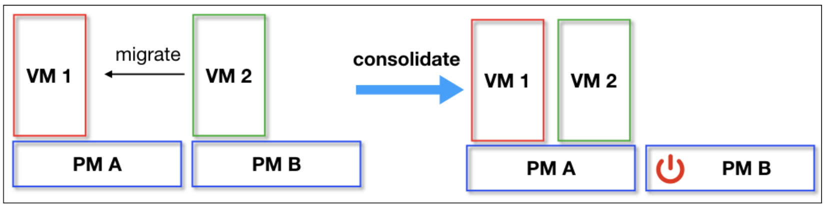


Figure 2.4: A Server Consolidation example: Initially, each PM runs an application wrapped with a VM in a low resource utilization state. After the consolidation, both VMs are running on PM A, so that PM B is turned off to save energy [6].

Application initial placement

Application initial placement is applied when new applications arrived. The task is to place them into a set of PMs [71] so that all applications' resource demand are satisfied and the energy consumption is minimized.

Periodic Optimization

Periodic Optimization is applied periodically to adjust the current placement of applications. The task is to re-place the current applications so that the energy consumption is minimized and the cost of migration is minimized. This problem is often modeled as a multi-objective static server consolidation problem.

Dynamic Placement

Dynamic placement is applied in two scenarios [71]: Overloading and underloading. Overloading is a scenario where the actual demand of applications in a PM is higher than the PM's resources. Therefore, one or more applications' performance degraded. Underloading is a scenario where the PM is running in low utilization. In both scenarios, the applications running inside the PM will be moved to other PMs in an online fashion which requires a dynamic placement approach [15].

2.2.2 Server consolidation

Server consolidation is a resource management strategy aims at improving the resource utilization and decreasing the energy consumed by PMs. A general step of server consolidation is shown in Figure 2.4, a number of VMs is migrated to fewer number of PMs. Server consolidation is often applied to solve the problem of PM sprawl [52]: a situation that more PMs are used in a low-utilized way. Server consolidation is usually model as a vector bin packing problem.

Types of Server consolidation

Server consolidation can be done in two ways: Static and dynamic [102, 113] which are applied in different placement decision 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.

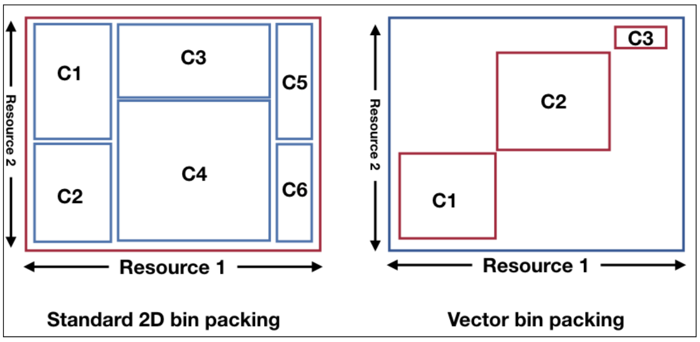


Figure 2.5: A comparison between standard bin packing and vector bin packing

Vector Bin Packing Model

Server consolidation is typically modeled as a Vector bin packing problem which is a variant of standard bin packing problem (see Figure 2.5). Vector bin packing is also referred as multi-capacity [60] or multi-dimensional bin packing problem [115]. Vector bin packing is particularly suitable for modeling resource allocation problems where there is a set of bins with known capacities and a set of items with known demands [76]. The optimization objective is to minimize the number of 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 or discrete number $I^i \in R^d$. A valid solution is packing I into bins $B^1, B^2, \ldots, B^k$. For each bin j and each dimension i, 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. Notice that, the items assigned to bins do not consider the positions in the bins, that is, there is no geometric interpretation of the items or bins [49]. Vector bin packing reduces to the classic *bin-packing* problem when $d = 1$. Vector bin packing is an NP-hard problem in strong sense, as it is a generalized bin packing problem.

Virtualization Technologies

Two technologies can be used to achieve server consolidation: clustering and virtualization. Clustering is used in a situation that the applications running in bare-mental PMs are I/O intensive. This is because current virtualization technologies such as KVM [54] or Xen [5] are not suitable for data intensive application because they have a 20% to 55% of reduction of I/O bandwidth (e.g disk reads and writes, network bandwidth) in comparison with bare-mental PM [86]. In our research, we consider the applications which require small CPU utilization (e.g 15%) and low I/O needs such as web services, therefore, this section mainly discusses the virtualization technology.

Virtualization [99] partitions a PM'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' was originally invented to enable isolated software testing, because VMs can provide good isolation so that multiple applications can run in separated VMs within the same PM without interfering each other [92]. 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 2.6): Hypervisor-based or VM-based and container-based virtualization.

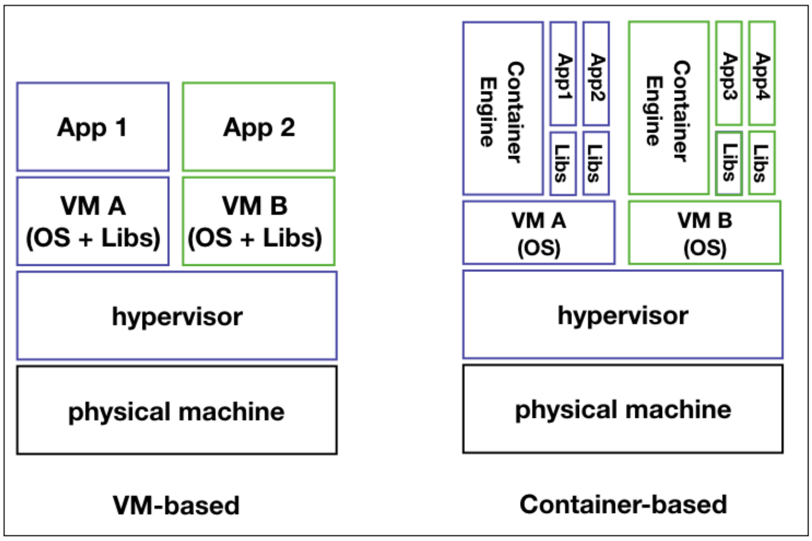


Figure 2.6: A comparison between VM-based and Container-based virtualization [79]

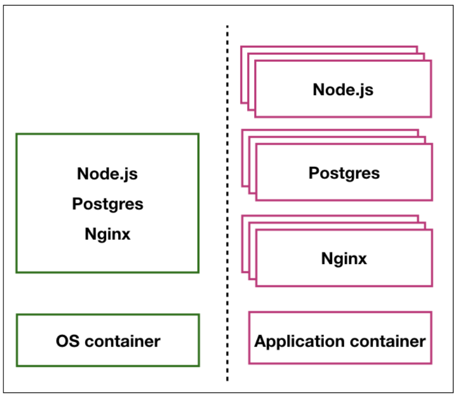


Figure 2.7: A comparison between OS container and Application container [?]

VM-based Virtualization A VM-based virtualization uses VM as its fundamental resource management unit. A VM allows independent Operating system runs on it. The system includes a new layer of software - the hypervisor or the virtual machine monitor (VMM). The VMM arbitrates accesses to the PM's resources so that OS from VMs can share. In the previous decades VM-based hypervisors such as Xen [5], KVM [54], and VMware ESX [104] dominate this field. VM-based Server consolidation [119] utilized dynamic migration techniques (e.g pre-copy [23] and post-copy [46]) to resolve the low utilization problem.

Container-based Virtualization Container-based virtualization uses both VM and container as its fundamental resource management unit. Container-based virtualization is also often addressed as operating-system-level virtualization. It includes two types of container: OS container and application container [?]. OS container (as shown in Figure 2.7) can run in both PM and VM which allows multiple applications running inside. There are mainly three implementations of OS-level of containers: OpenVZ, Google's control groups, and namespace [85]. Google and Facebook have been using OS container for years and being beneficial for its lightweight and fast communication among applications.

An application container, such as Docker, Rocket and Kubernetes [10], accommodates a single application in it. Application container is much more flexible than OS container because each container to have its separated environment (e.g. libraries) so that the container can be deployed in different OSs. Application container also provides a finer granularity of resource management by enabling an application level of operations including deployment, scaling, and migration.

Comparison between Container-based and VM-based virtualization This section mainly discussed the differences between application container and VM in terms of their characteristics. Os container is not discussed because it is very similar to a VM. Several research [33, 38, 111] have compared VM-based and container-based virtualization. The advantages of container-based virtualization are mainly from three aspects:

- Container-based virtualization has a lightweight management which generates much less overhead than a VM hypervisor.
- Container-based virtualization shares OSs which reduces the overhead of multiple OSs while VMs have to run separated OSs.
- 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.

These advantages together with fine granularity resource management of container, can be used to improve the energy efficiency by overcoming three main defects in traditional VM-based cloud data centers.

The defects are listed as follows:

- Resource over-provisioning
  Cloud users tend to reserve over more resources for ensuring the Quality of Service at peak hours [20] which leads to low resource utilization. Cloud users do not completely rely on auto-scaling 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.

  Application container overcomes this defects by dynamically adjusting its size. With vertical scaling, resource reservation can be performed in a minimum level, which leads to a high utilization of resources.

- Unbalanced usage of resources
  Specific applications consume unbalanced resources which leads to vast amount of resource wastage [98]. 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.

  Application container overcomes this defects by combining complementary containers into a VM. The size of container reflects the requirement of the application. By using appropriate placement algorithm, the balanced usage of resources is achievable.

- Heavy overhead of VM hypervisors and redundant operating systems (OSs)
  Heavy overhead of hypervisors and redundant OSs running in the PM causes huge resource wastage. Traditional VM provides a complete independent environment for deploying software which includes its own OS and libraries. However, as most applications only require a general OS such as Windows or Linux, multiple duplicate OSs running in the system is a waste of resource.

Application container overcomes this defects by running on top of a VM.

In summary, container-based cloud data center has the potential to improve further improve the energy efficiency than VM-based cloud. However, the placement decision of containers and VMs are quite different. Most of the current research focus on VM-based cloud while their approaches cannot be directly used on container-based cloud. Next section discusses the detailed literature review on three placement decision scenarios in container-based and VM-based cloud.

2.3 Related Work

In this section, we summarize the related works in the terms of three resource management processes: application initial placement, periodic consolidation and dynamic consolidation, corresponding to our proposed objectives. In each sub section, we analyze the container-based and VM-based approaches, discuss their models and methodologies.

2.3.1 Application Initial Placement

Application initial placement is one of the major process in resource management as discussed in Section 2.2.1. The task of application initial placement for cloud provider is to deploy a number of VMs or containers to a minimum number of empty physical machines (PMs), because energy consumption is proportional to the PM number.

This section first discusses container-based approaches including modeling and methodology. The modeling includes power model and wastage model. Then, we will discuss and reason their disadvantages in design. In order to remedy their problems, we will review a number of VM-based placement including traditional bin packing and more advanced approaches.

Container-based Application Initial Placement

This section will briefly discuss the models of container-based approaches. Furthermore, our focus is to discuss the design of separate concern of container placement and VM placement as well as the collaborative placement.

There are only few container-based research in the literature which mainly follow the VM-based modeling. Piraghaj et al [79] and Mann [65] both consider a linear energy model which is widely adopted in the literature [112]. They consider two resources: CPU and memory. In addition, the constraints are allocated resources cannot exceed the PMs' capacity. In terms of wastage model, so far, they did not use any measure on the residual resources. In contrast, in most VM-based approaches, wastage has been widely considered. It will be discussed in the next section.

The design of three-tier structure: container-VM-PM raises a placement concern, optimize each tier separately and collaboratively. Piraghaj et al [79] consider it separately. Their placement strategy replies on fast and simple heuristic: in each tier of placement, they apply First Fit algorithm. While their main contribution is not the placement strategy but an architecture for container-based resource management. In this architecture, it allows the cloud providers to customize the size or type of VM when allocating applications instead of traditional fixed size VM. They determine the size of VM by two steps, in a pre-execution phase, they perform clustering technique on historical workload data from Google Cluster Data. In this way, they assert that the applications with similar workload pattern can be categorized into the same group. In the execution phase, when a number of deployment requests of container arrives, if there are enough available resources in the existing VMs, they perform First Fit on containers. If new VMs need to be initiated, they first determines the size of VMs by designed policies. Then, the containers are allocated to certain size of VMs.

Mann's research [65] is the earliest study which realizes this two level of placement are interact with each other, therefore, they must be considered collaboratively. Different from Piraghaj's assumption, they consider the container is based on an existing IaaS where fixed types of VM are provided. Therefore, the problem becomes three folds: 1. VM size selection for containers, 2. Place containers, 3. VM placement, and they should be considered together. However, another concern is "why not placement as many containers as possible in a single VM which minimizes the overhead of hypervisor". The paper gives an answer of "Too big VMs limit the consolidation possibilities". In order to prove their interaction, they apply a fixed VM placement algorithm and considering a series of VM selection algorithms such as simple selection [42], Multiple selection, Maxsize, Consolidation-friendly. They discover that the final energy consumption varies with the selection algorithms. They claim that the performance is better when VM selection has more knowledge of the PMs' capacity. However, their study only focuses on the partial placement with fixed VM placement algorithm. The answer of "How these two level of placement interact ?" is still undiscovered.

There are mainly three reasons for us to propose a distinct approach from Piraghaj [79] to solve the container-based placement problem. First, their proposed architecture assumes arbitrary size of VM can be created when requests arrive. While our assumption is that the container-based architecture is based on traditional IaaS, where fix-size VMs provide the fundamental resources. Second, from the perspective of energy efficiency, the allocation of container and VM interact with each other. That is, the minimum number of VMs does not necessary lead to the minimum number of PMs, because the type of VMs also affect the results. Therefore, their approach cannot guarantee the energy consumption is near optimal. This inspires us to simultaneously allocate containers and VMs. Third, both approaches did not consider the OS requirement of containers or applications. We argue that this is another critical reason for deploying containers into different VMs besides the "limit the consolidation possibilities" argument states by Mann [65].

In order to solve the two-level of optimization problem, we believe one of the promising way is to model the problem as a bilevel optimization [24] as described in detailed in Section 2.3.1. Bilevel optimization naturally models the interaction so that this problem can be tackled by an optimization algorithm. The main challenge is that Bilevel optimization is a strongly NP-hard problem [89], therefore, traditional approaches such as Branch-and-bound [4] can only be applied in very small problems, or convert the problem into a single-level optimization problem. Evolutionary algorithms, on the other hand, becoming popular approaches in this field [89, 107].

VM-based Application Initial Placement

This section first reviews a number of traditional approaches for the VM placement problem. Their drawbacks will be presented. Then, a number of advanced approaches will be examined. We mainly study the following five aspects: resources, power model, wastage model, objective, and algorithm.

Most of the works model VM placement problem as variants of bin packing problem and propose extensions of greedy-based heuristics such as First Fit Decreasing (FFD) [110], Best Fit, Best Fit Decreasing [8] etc. However, as VM placement is an NP-hard problem, greedy-based approaches can not guaranteed to generate near optimal solutions. Mishra and Sahoo's paper [72] further analyzes and discusses the drawbacks of these approaches. They found that, instead of standard bin packing, only vector bin packing is suitable for

Table 2.1: A Comparison of different models and approaches

Research	Resources	Algorithm	Power model	Wastage model	Objective
Xu et al [116]	CPU and RAM	GGA and Fuzzy multi-objective	Linear	balance resources	three
Gao et al [43]	CPU and RAM	Ant Colony Optimization	Linear	balance resources	Two
Ferdaus et al [39]	CPU, RAM, and IO	Ant Colony Optimization	Linear	Sum of resources	Single
Wang and Xia [106]	CPU and RAM	MIP	Cubical	No	Single
Wilcox et al [109]	CPU and RAM	GGA	Linear	Sum of resources	Single
Xiong and Xu [115]	CPU,RAM,Bandwidth,Disk	PSO	Non-linear	Sum of resources	Single

modeling resource allocation (see Section 2.2.2). Another drawback of traditional bin packing heuristic is that they do not consider the balance among resources which is a critical issue for vector bin packing problem. Their main contribution is that they list five principles for a good design of objective function, specially, the core idea is to capture the balance among resources.

Based on this insight, Gao et al [43] and Ferdaus et al [39] both propose an Ant Colony Optimization based metaheuristic using a vector algebra complementary resource utilization model proposed by Mishra [72]. They considered three resources CPU, memory, and network I/O with two objectives: minimizing power consumption and resource wastage. They apply the *Resource Imbalance Vector* to capture the imbalance among three resources. Meanwhile, they use a linear energy consumption function to capture the relationship between CPU utilization and energy [35]. Their solution was compared with four algorithms: Max-Min Ant System, a greedy-based approach, and two First Fit Decreasing-based methods. The results show that their proposed algorithm has much less wastage than other algorithms.

Xu and Fortes [116] propose a multi-objective VM placement approach with three objectives: minimizing total resource wastage, power consumption and thermal dissipation costs. They applied an improved grouping genetic algorithm (GGA) with fuzzy multi-objective evaluation. Their wastage by calculating as differences between the smallest normalized residual resource and the others. They also applied a linear power model to estimate the power consumption [62].They conduct experiments on synthetic data and compare with six traditional approaches including First Fit Decreasing (FFD), Best Fit Decreasing (BFD) and single-objective grouping GA. The results showed the superior performance than other approaches.

Wilcox et al [109] also propose a reordering GGA approach because GGA can effectively avoid redundancy [34]. They use a indirect representation [82] which represents the packing as a sequence. In order to transform the sequence into a packing, they applied an ordering operator which, in essence, is a first fit algorithm. This design naturally avoids infeasible solution, therefore, there is no need for constraint handling.

Wang and Xia [106] develop a MIP algorithm for solving large-scale VM placement problem under a *non-linear* power consumption model. Instead of considering the power consumption as a linear model like most researchers, they consider the CPU frequency can be adjust by dynamic voltage and frequency scaling (DVFS), therefore, the power consumption is a cubical power function of frequency. In order to solve the non-linear problem, they first use a linear function to approximate the cubical function. Then, they first use the Gruobi MIP solver to solve the relaxed linearized problem. Then, they apply an iterative rounding algorithm to obtain the near optimal solution.

$$\delta = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{d} (u_j^i - ubest_i)^2} \tag{2.1}$$

Xiong and Xu [115] propose a PSO based approach to solve the problem. Their major contribution is using a total Euclidean distance δ to represent the distance between current resource utilization and the optimal resource utilization (see equation 2.1) where d is the dimension of resources, u_j^i is the current resource utilization of j in a PM i, $ubest_i$ is the predefined optimal resource utilization (e.g 70% CPU utilization). Another contribution is their representation used in PSO. They represent the allocation of each VM to a PM as a probability and let particles search through the indirect solution space.

In summary, most of VM-based placement approaches consider two or three resources (I/O has not been considered in many approaches because they assume that network attached storage (NAS) is used as a main storage along the cluster [73]). After Mishra unreal the principles of vector bin packing, most research apply a balance-measure among resources as their objectives. EC approaches are widely used because they are better performed than traditional heuristics and faster than ILP methods.

Bilevel Optimization

The joint allocation of container and VM can be modeled as a bilevel optimization. A bilevel optimization [24] is a kind of optimization where one problem is embedded within another. The general formulation of a bilevel optimization problem can be defined as:

$$\begin{aligned} min_{x \in X, y} \quad & F(x, y) && \text{(2.2a)} \\ s.t \quad G(x, y) \leq 0&, && \text{(2.2b)} \\ min_y \quad & f(x, y) && \text{(2.2c)} \\ s.t \quad & g(x, y) && \text{(2.2d)} \end{aligned}$$

The lower-level problem is the function $f(x, y)$, where the decision variable is $y \in \mathbb{R}^{n_2}$. The upper-level problem is the function Fx, y where the decision variable is $x \in \mathbb{R}^{n_1}$. The function $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}$ and $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}$ are the *upper-level* and *lower-level objective functions* respectively. The function $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}^{m_1}$ and $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}^{m_2}$ are called the *upper-level* and *lower-level constraints* respectively.

Bilevel optimization problem has a hierarchical structure which may introduce difficulties such as non-convexity and disconnectedness even for simple cases such as bilevel linear programming problems is strongly NP-hard [89].

In practice, there are a number of problems that are bilevel in nature. For example, transportation related: work design, optimal pricing [16, 25], management: network facility location [95], and engineering related: optimal design [53].

Techniques for Bilevel optimization

A number of studies have been conducted on bilevel optimization [24, 31]. Approximation algorithms such as Karush-kuhn-Tucker approach [11, 45], branch-and-bound [4] are often applied to solve the problem. Most of these approaches are not applicable when the problem size increases.

Evolutionary methods have been applied to bilevel optimization problem since 90s. Mathieu et al [67] proposed an genetic algorithm (GA) based approach. It uses a nested strategy - the lower level is optimized with a linear programming method and the upper level apply a GA.

Oduguwa and Roy [75] proposed a co-evolutionary approach for bilevel problems. Two population are co-operated to find the optimal solution, where each population handles a sub-problem.

Wang et al [107] proposed an evolutionary algorithm based approach with a constraint handling technique. Their approach is able to handle non-differentiability at the upper level objective function, but not in constraints and lower level objective function. Later on, Wang proposed an improved version [108] that shows better performance than the previous version.

Particle Swarm Optimization [61] was also used in solving bilevel problems. A recent work is from Sinha et al [89], they propose a bilevel evolutionary algorithm (BLEAQ) works by approximating the optimal solution mapping between the lower level optimal solutions and the upper level variables. BLEAQ was tested on two sets of test problems and the results were compared with WJL [107] and WLD [108]. The results show BLEAQ is much faster than previous approaches. One major drawback of evolutionary algorithms is its high computation cost which limits the problem size from growing bigger.

In conclusion, as the complexity of the problem, practical problems with bilevel nature are often simplified into a single-level optimization problem which can achieve a satisfactory level instead of optimal. Classic algorithms often fail because of the nature of bilevel problem such as non-linearity, discreteness, no-differentiability, non-convexity etc. EC algorithms have been successfully applied on bilevel problems.

2.3.2 Periodic consolidation

Periodic consolidation (see Section 2.2.1) is an process that optimizes the current allocation of resources in a periodic fashion [71]. This is because the cloud datacenter is a dynamic environment with continuous deployment and releases that causes degradation of the resource utilization, thus, the allocation needs to be adjusted when the performance degrades to a certain level. In comparison with application initial placement (see Section 2.3.1), the similarity is that they are both static approaches which consider a batch of applications and PMs. The difference is that periodic consolidation needs to take the cost of application migration into account, therefore, it is often considered as a multi-objective optimization problem.

Although periodic consolidation has been applied in VM-based Cloud for years [40,73], it has not been studied in the new context of container-based Cloud. Therefore, this section will first discuss VM-based models, especially the migration models. Secondly, we will discuss the approaches in periodic consolidation in the VM-context, specifically, in terms of the prediction of workload, these gaps existed in both VM and container context.

Murtazaev and Oh [73], Beloglazov et al [7] and Ferreto et al [40] realize that the migration process generates a large overhead so that it should be used as few as possible. In their migration model, they use the number of migration as the optimization objective. Murtazaev's approach minimize this number by developing an algorithm which always chooses a VM in the least loaded PM and attempts to allocate them on the most loaded PMs. Based on this idea, they develop a heuristic based on First and Best Fit. They select a candidate VM based on a surrogate weight of the resources it used. Beloglazov, on the other hand, considers different criteria for selecting candidate VMs. They not only considers the utilization of VMs but also the utilization of the original PM and target PMs. They also propose a simple heuristic: a modified Best Fit Decreasing to solve the problem. However, these two approaches develop their selection criteria in a greedy fashion which may lead to a local optimal. Ferreto proposes a preprocessing step before the placement algorithm. It first orders the VMs according to their workload variation. Then, it only performs placement on those VMs with the highest variability. These three papers provide some insight that a good placement algorithm should consider more than the utilization of host and target PMs, but also the variation of workload.

Most previous consolidation approaches [21,37,103] only consider static workload. That is, they use a peak or average workload as a represented value as the consolidation input. In most of cases, this will lead to either low utilization: peak time only account for small proportion of the total time, or more migrations: extra migration are performed on workload changes. Therefore, the consolidation is more than aggressively concentrate workload on as few PM as possible, but also considers the robustness. The robustness is referred to the capability of enduring the variation of workload without make too many changes.

In order to achieve robustness, the workload variation must be taken into account. Boboff [12] analyzed a large number of traces from real world datacenter. They categorize workloads into three main groups:

- Weak variability.
- Strong variability with weak periodic behavior.
- Strong variability with strong periodic behavior.

Workload with weak variability can be directly packed. The only problem is that their long-term workload can also be changed. For the second type of workload, it is hard or even impossible to predict its behavior. The third type of workload can be predicted. However, it is hard to find the applications with compensated workload patterns.

Meng et al [70] proposed a standard time series technique to extract the deterministic patterns (e.g trends, cycles and seasonality) and irregular fluctuating patterns from workloads' CPU utilization; they assume the periodic behavior of workload will preserve in the future and predict the irregular parts with two approaches: with and without explicit forecast error models. Then, applications are paired according to their negative correlation. They evaluate the workload prediction and application selection with a server consolidation task. They use First Fit to allocate paired applications. During the consolidation, The consolidation results show that they use 45% less PMs for hosting the same number of VMs. Furthermore, their approach is more robust since the variation of workload is considered. However, they only consider two complementary applications at a time.

2.3.3 VM-based Dynamic Consolidation Techniques

Forsman et al [41] propose two distributed migration strategies to balance the load in a system. *The push* strategy is applied on overloaded PM; it attempts to migrate *One* VM at a time to less loaded PMs. *The pull* strategy is applied on underutilized PMs request workload from heavier loaded PMs. Each of the strategy is executed on each PM as an intelligent agent. They share their status with each other through a communication protocol. There are several interesting features of their approach. First, they apply an adaptive high-load threshold (e.g 0.7 of overall CPU utilization) so that it considers the environment changes. Second, they use an EWMA algorithm to reduce the unnecessary migration because EWMA [47] is useful in smoothing out variations in the average load. Third, they applied an entropy to model the load distribution which is also applied in some previous approaches [56, 81]. Their system is agent-based which means large amount of communication may occur between nodes, this would certainly cost extra network resources which are not discussed. Therefore, we expect to design a centralized system, where all nodes are controlled by a controller.

Xiao et al [113] make two contributions, first, they build a quadratic energy model for the energy consumption of PM and a linear model for the energy consumption of migration [63]. Second, they propose an algorithm based on Multiplayer random evolutionary game theory to solve the problem. In their approach, VMs are mapped into players that take part in the

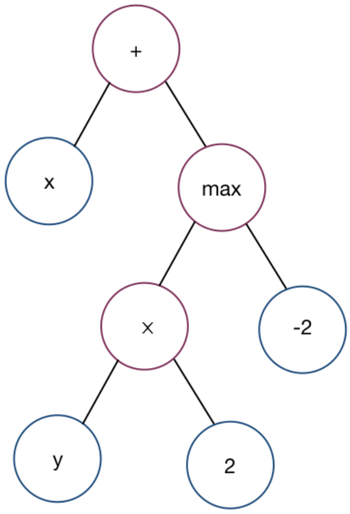


Figure 2.8: GP program that represents x + max(y × 2, -2)

evolutionary game. In each iteration, all the players choose their best feasible action, i.e, Migrate to a PM which can minimize the energy consumption. Some players will randomly choose PM to avoid being stuck at a local optimal. Their approach is compared with First Fit, Best Fit Increasing, Best Fit Decreasing, Greedy and Load Balance rule. The solutions show their approach can improve energy consumption greatly, especially in the scenario that the distributions of VMs are very centralized.

Genetic Programming

Genetic programming [55] is an evolutionary computation technique, inspired by biological evolution, to automatically find computer programs for solving a specific task. In a GP population, each individual represents a computer program. In each generation, these programs are evaluated by a predefined fitness function, which accesses the performance of each program. Then, individuals will go through several genetic operators such as selection, crossover, and mutation. A number of top individuals will survive to the next generation while others will be discarded. The major difference between GA and GP is that, each GP individual is represented as a tree with variant depth instead of a string. This representation is particular suitable for a program. For example, a GP individual is showed in Figure 2.8 which is a program x + max(y × 2, -2). The variables {x, y} and constraint {-2, 2} are called terminal of the program. The arithmetic operations {+, ×, max } are called functions in GP. A GP individual is a specific combination of elements in terminal set and functional set. In order to observe the relationship between a function and its subtrees, the GP programs are usually presented to human users by using the $prefix$ notation similar to a Lisp expression, for example, x + max(y × 2, -2) can be expressed as (+ (x (min (× y 2) -2))).

GP approach based hyper-heuristics (GP-HH) has been applied in many applications such as Job shop scheduling to evolve dispatching rules [74]. The term hyper-heuristics [26] means "heuristics to choose heuristics". *Dispatching rule* is essentially a heuristics [77] used in a scheduling context. Resource allocation problems are also in the scheduling category and they are often modeled as bin packing problems. GP-HH has been applied in generating heuristics for bin-packing problems [17,80,88]. These research have shown that GP-HH can generate excellent heuristics which have equal or better performance than human designed heuristics.

In the Cloud computing context, Cloud resource allocation usually has extra constraints such as multi-dimensional resources, migration costs, heterogeneous PMs etc. These constraints make the Cloud resource allocation problem much harder than original bin packing [64]. Therefore, traditional bin packing approaches such as First Fit Decreasing, Best Fit etc cannot perform well in this context. GP-HH, therefore, is a promising technique can be used to automatically generate heuristics under multiple constraints.

2.4 Summary

This chapter reviewed the main concepts of cloud resource management and server consolidation. The challenges of server consolidation in container-based cloud data center were discussed. This chapter also discussed the limitations of existing work on three placement decision scenarios in both container-based and VM-based data center.

- Current research lacks appropriate model to capture the relationship between containers, VMs and PMs. Hence, most research on container-based application initial placement conducts the placement in two independent steps: container-VM and VM-PM. These approaches neglected the interaction between two levels of placement, hence, they cannot reach a near optimal energy consumption. A bilevel model for the joint placement of containers and VMs need to be proposed. Related sub models such as energy model, workload model, variables and constraints need to be further investigated.

- Periodic optimization has not been studied in the container context. A bilevel multi-objective energy model needs to be proposed which considers minimizing migration cost as well as minimizing energy consumption.

- Traditional periodic optimization mostly consider static workload. Thus, it is very likely lead to large number of adjustment of applications' placement in the future because the fluctuation of workloads. These adjustment will increase the cost for Cloud providers. In order to provide a robust placement of application, various predictable workload patterns such as linear continuously changing can be considered. It needs more investigation on how to represent various workloads and how to combine them in a compact structure.

- Current dynamic placement approaches are based on simple bin-packing algorithms and manually designed heuristics. These heuristics are either perform poorly or cannot be applied with specific constraints. A hyper-heuristic approach can learn from previous good placement patterns and automatically generate heuristics. In order to design a hyper-heuristic, features of various workload need to be investigated.

This research aims to address the above-mentioned issues. The next chapter will focus on the initial work conducted in investigating NSGA-II for bilevel application initial placement.

Chapter 3

Preliminary Work

This chapter presents the initial work conducted in investigating NSGA-II for the joint placement of container and VM. In this work, we consider the web services are deployed in containers, therefore, "web service" is used in the content instead of container. This work investigates the bilevel model including sub models (e.g power model), variables, and constraints. Two optimization objectives were initially considered: minimizing the energy consumption and the total price for the used VMs. A NSGA-II approach is applied for optimizing the problem. The result covers the evaluation of the proposed algorithm along with analysis, and concluding remarks and discussion of future work is given in conclusion (Section 3.5).

3.1 Related models

3.1.1 Workload model

Xavier el al [117] 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.

3.1.2 Power Model

Shekhar's research [94] 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 [13] 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.

Beloglazov et al. [**?**] 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 [57,83].

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

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^i_n. X^i_n 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^n_j. Y^n_j 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.3.3.

$$U_n = \begin{cases} \frac{\sum_{i=1}^{t} R_i \cdot A_i \cdot X^i_n}{VA_n}, \text{If } \frac{\sum_{i=1}^{t} R_i \cdot A_i \cdot X^i_n}{VA_n} < 1 \\ 1 \qquad\qquad\qquad\quad, \text{otherwise} \end{cases} \tag{3.3}$$

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 [78]. 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:

$$\text{minimize} \quad Energy = \sum_{j=1}^{p}(k \cdot V_{max} + (1-k) \cdot V_{max} \cdot \sum_{n=1}^{v} U_n \cdot Y_j^n) \tag{3.4}$$

$$Cost = \sum_{j=1}^{p}\sum_{n=1}^{v} C_n \cdot Y_j^n \tag{3.5}$$

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.

$$\begin{aligned} \sum_{n=1}^{v} VM_n \cdot Y_j^n \leq PM_j \\ \sum_{n=1}^{v} VA_n \cdot Y_j^n \leq PA_j \end{aligned} \tag{3.6}$$

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.

$$\sum_{i=1}^{t} M_i \cdot R_i \cdot X_i^n \leq VM_n \tag{3.7}$$

3.3 Methods

As we have discussed, Multi-objective Evolutionary Algorithms are good at solving multi-objective problems and NSGA-II [29] 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

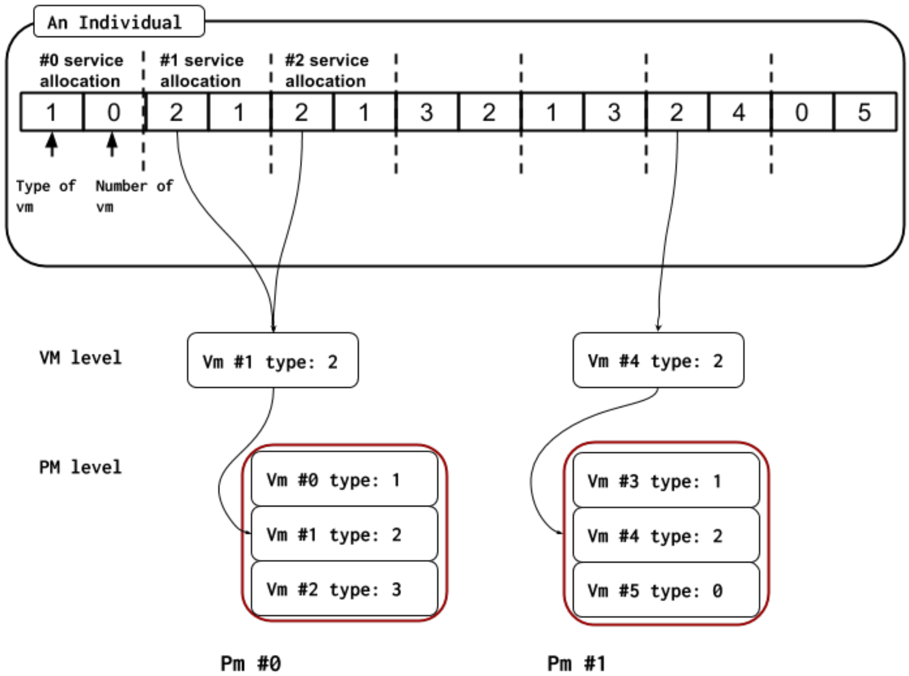


Figure 3.1: An example chromosome representation

#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

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.

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

```
for Each service t do
    Find its most suitable VM Type
    Randomly generate a VM type vmType which is equal or better than its most suitable type
    if There are existing VMs with vmType then
        randomly generate a number u
        if u < consolidation factor then
            randomly choose one existing VM with vmType to allocate
        else
            launch a new VM with vmType
        end if
    else
        Create a new VM with its most suitable VM type
    end if
end for
```

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 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.

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:

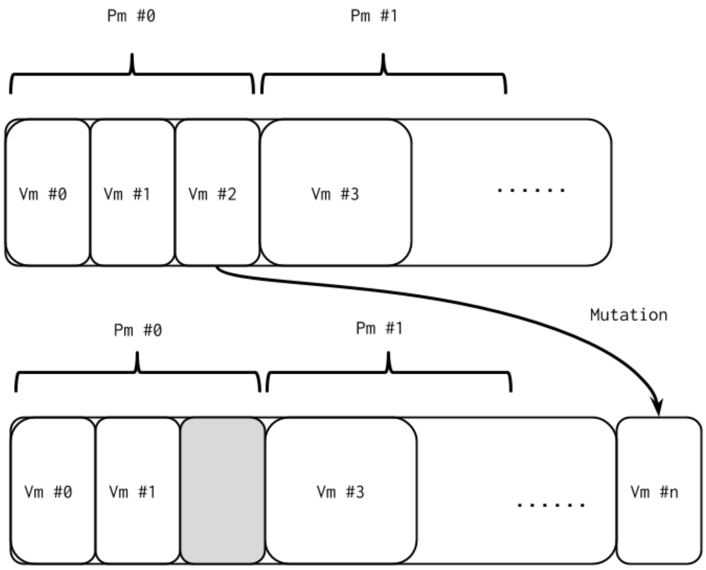


Figure 3.2: An example mutation without insertion that causes a lower resource utilization

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

```
for Each service do
    Randomly generate a number u
    if u < mutation rate then
        find the most suitable VM Type for this service
        Randomly generate a number k
        if k < consolidation factor then
            calculate the utilization of used VMs
            assign each VM with a fitness value of 1 / utilization and generate a roulette wheel according to
            their fitness values
            Randomly generate a number p, select the VM according to p
            Allocate the service
        else
            launch a new VM with the most suitable VM Type
            insert the new VM in a randomly choose position
        end if
    end if
end for
```

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 [29].

3.3.6 Fitness Function

The cost fitness (Eq.3.5) is determined by the type of VMs at which web service are allocated. The energy fitness is shown in Eq.3.4, the utilizations (Eq.3.3) 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 [58].

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

```
Initialize a population P
while Termination Condition is not meet do
    for Each individual do
        Evaluate the fitness values
        Calculate the violation
    end for
    non-Dominated Sorting of P
    calculate crowding distance
    while child number is less than population size do
        Selection
        Mutation
        add the child in a new population U
    end while
    Combine P and U { for elitism}
    Evaluate the combined P and U
    Non-dominated sorting and crowding distance for combined population
    Include the top popSize ranking individuals to the next generation
end while
```

3.4 Experiment

3.4.1 Dataset and Problem Design

This project is based on both real-world datasets *WS-Dream* [120] and simulated datasets [14]. 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 [14].

Table 3.1: Problem Settings

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

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

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

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.

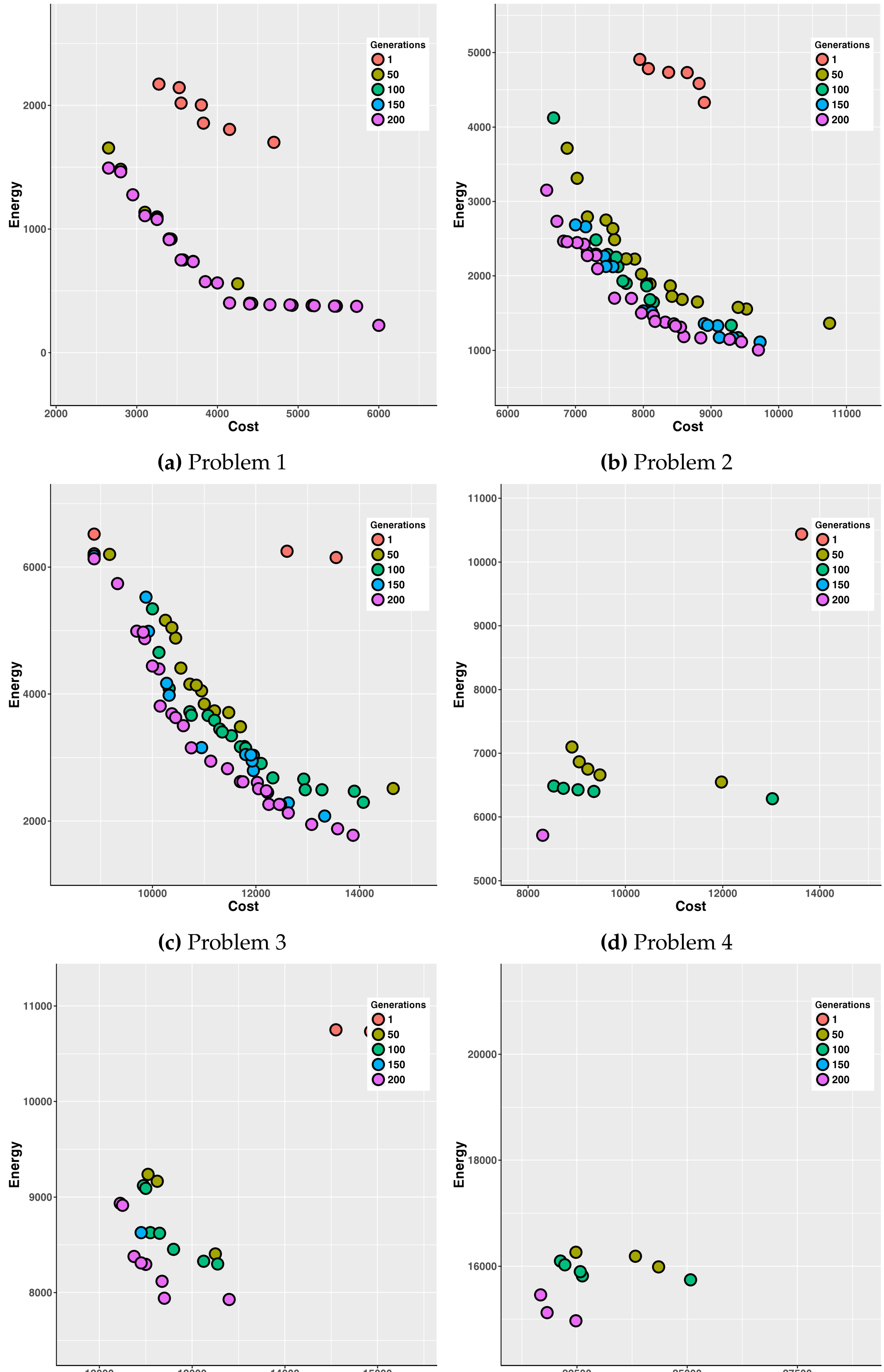
3.4.2 Results

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,



(a) Problem 1

(b) Problem 2

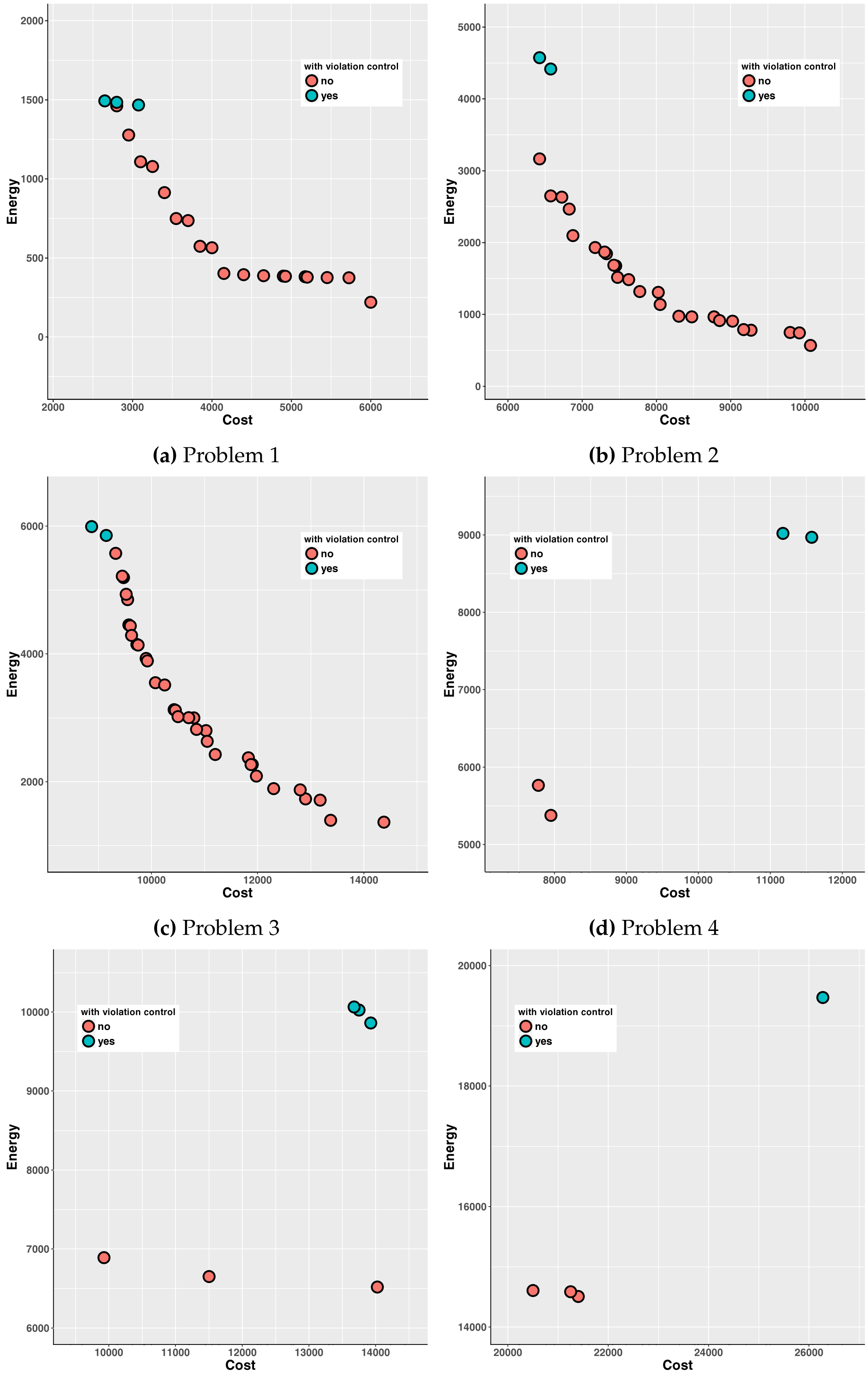
(c) Problem 3

(d) Problem 4

(e) Problem 5

(f) Problem 6

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



(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

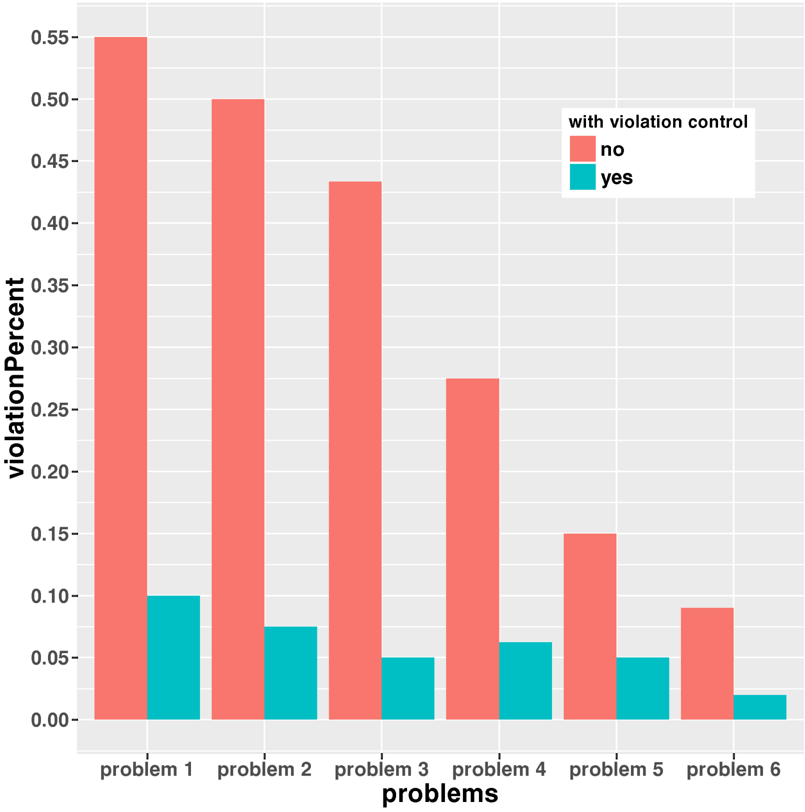


Figure 3.5: Violation Percentage comparison between selection with violation control and without violation control

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

3.5 Conclusion and Future work

This work investigates the bilevel model for the joint placement of container and VM. Several sub models such as workload model, power model were considered. A multi-objective formulation of the bilevel problem were established. Two objectives, minimizing the cost of used VMs and minimizing the energy consumption are achieved. In order to optimize the problem, we propose a NSGA-II based algorithm with specific designed representation which reduces the bilevel placement into one single-level. Genetic operators such as population initialization, mutation, selection were also designed for generating valid solutions and handling the constraints. 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 bilevel optimization of 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. A new bilevel model for the joint placement of container and VM problem. The bilevel model will address the relationship between containers, VMs, PMs, and energy consumption. The bilevel model can be used in optimizing the energy consumption in a container-based cloud data center.

2. An EC-based bilevel optimization algorithm for the application initial placement based on previous proposed bilevel model. This algorithm is expected to achieve better performance than existing VM-based approaches in terms of energy consumption.

3. An EC-based bilevel optimization algorithm with a clustering-based pre-processing approach to improve the scalability of previous proposed algorithm. This work can be used to reduce the complexity of the bilevel problem by combining containers into larger groups.

4. EC-based bilevel multi-objective approaches for periodic optimization with considering various types of workload. This work will modify previous model to adapt to the multi-objective problem. It will also propose several multi-objective approaches with aggregation and Pareto front methods. In addition, a uniform representation for various workloads will be proposed to make the algorithm more resilient.

5. A new genetic programming hyper-heuristic (GP-HH) approach for single-objective dynamic placement with various types of workload. The proposed GP-HH is expected to learn good placement patterns from previous placement solutions output from previous proposed algorithms. After the learning process, GP-HH can automatically generate heuristics. These generated heuristics can achieve fast dispatching containers to suitable PMs. In addition, the containers and VMs can achieve a near-optimal solution in energy consumption.

4.1 Overview of Project Plan

Six overall phases have been defined in the initial research plan for this PhD project, as shown in Table 4.1. The first phase, which comprises reviewing the relevant literature, in-

Table 4.1: Phases of project plan

Phase	Task	Duration (Months)
1	Reviewing literature, overall design, selection of datasets and writing the proposal	12 (Complete)
2	Develop a single-objective EC-based approach for the joint allocation of containers and VMs	7
3	Develop multi-objective EC-based approaches for container-based cloud in periodic optimization with considering various types of workload	7
4	Develop a cooperative Genetic programming based hyper-heuristic approach for dynamical placement.	7
5	Writing the thesis	6

Table 4.2: Time Line

Task	Months											
	2	4	6	8	10	12	13	16	18	20	22	24
Literature Review and Updating	x	x	x	x	x	x	x	x	x	x	x	x
Develop a new model for the joint placement of VMs and containers	x											
Develop an EC-based bilevel optimization approach for the joint placement of VMs and containers		x										
Improve the scalability of the EC-based approach with a pre-processing method			x									
Modify the model and develop a baseline aggregation approach for the multi-objective problem				x	x							
Develop an EC-based approach to solve the multi-objective joint allocation problem with Pareto front approach						x	x					
Propose an EC-based multi-objective algorithm for periodic optimization considering various types of predictable workload								x				
Develop a baseline GP-based hyper-heuristic approach									x	x		
Construct a GP primitive set by applying feature extraction on various types of application workload									x	x		
Develop a Cooperative GP-HH approach to evolve dispatching rules for placing container and VMs									x	x		
Writing the first draft of the thesis										x	x	
Editing the final draft										x	x	x

vestigating 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 Table 4.2, 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 organized into the following chapters:

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

- *Chapter 2: Literature Review*
  The literature review will illustrate the fundamental background of Cloud computing, resource management, and server consolidation. It will examine the existing work on VM-based and container-based server consolidation, discuss concepts in this field in order to provide readers with the necessary background. Multiple sections will consider the issues such as application initial placement, periodic placement, and dynamic placement. The focus of this review is on investigating server consolidation techniques.

- *Chapter 3: Develop EC-based approaches for the single objective joint placement of containers and VMs for application initial placement.*
  This chapter will establish a new bilevel model for the joint placement of container and virtual machine problem. Based on this model, this chapter will introduce a new EC-based bi-level approach to solve the application initial placement problem.

- *Chapter 4: Develop EC-based approaches for the multi-objective joint allocation problem for periodic optimization*
  This chapter will first modify the previous proposed bilevel model to adapt to multi-objective problem with various workload. It will also propose new EC-based approaches for the bilevel multi-objective joint placement of containers and VMs with considering various types of workload. It is then followed by algorithm performance evaluation that contains an experiment design, setting, results and analysis.

- *Chpater 5: Develop a hyper-heuristic single-objective Genetic Programming (GP) approach for automatically generating dispatching rules for dynamic placement*
  This chapter focuses on providing a Genetic Programming-based hybrid heuristic approach to automatic generate dispatching rules to dynamic consolidation problem. A cooperative GP-HH will be proposed in this chapter for generating dispatching rules for two-level of placement.

- *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 summarized. 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 on-line, using the university electronic resources. Other works may either be acquired at the university 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] 0002, Y. S., WHITE, J., LI, B., WALKER, M., AND TURNER, H. A. Automated QoS-oriented cloud resource optimization using containers. *Autom. Softw. Eng. 24*, 1 (2017), 101–137.

[2] ADHIKARI, V. K., GUO, Y., HAO, F., VARVELLO, M., HILT, V., STEINER, M., AND ZHANG, Z.-L. Unreeling netflix - Understanding and improving multi-CDN movie delivery. *INFOCOM* (2012).

[3] 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.

[4] BARD, J. F., AND FALK, J. E. An explicit solution to the multi-level programming problem. *Computers & Operations Research 9*, 1 (Jan. 1982), 77–100.

[5] 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.

[6] BARROSO, L. A., AND HÖLZLE, U. The Case for Energy-Proportional Computing. *IEEE Computer 40*, 12 (2007), 33–37.

[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. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. *MGC@Middleware* (2010), 1–6.

[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] BIANCO, L., CARAMIA, M., AND GIORDANI, S. A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: ... 17*, 2 (Apr. 2009), 175–196.

[12] BOBROFF, N., KOCHUT, A., AND BEATY, K. A. Dynamic Placement of Virtual Machines for Managing SLA Violations. *Integrated Network Management* (2007), 119–128.

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

[14] BORGETTO, D., CASANOVA, H., DA COSTA, G., AND PIERSON, J.-M. Energy-aware service allocation. *Future Generation Computer Systems 28*, 5 (2012), 769–779.

[15] BORODIN, A., AND EL, R. Yaniv (1998): Online Computation and Competitive Analysis.

[16] BROTCORNE, L., LABBÉ, M., MARCOTTE, P., AND SAVARD, G. A Bilevel Model for Toll Optimization on a Multicommodity Transportation Network. *Transportation Science 35*, 4 (Nov. 2001), 345–358.

[17] BURKE, E. K., HYDE, M. R., KENDALL, G., AND WOODWARD, J. Automating the Packing Heuristic Design Process with Genetic Programming. *Evolutionary Computation 20*, 1 (Mar. 2012), 63–89.

[18] BUYYA, R., BELOGLAZOV, A., AND ABAWAJY, J. H. Energy-Efficient Management of Data Center Resources for Cloud Computing - A Vision, Architectural Elements, and Open Challenges. *PDPTA cs.DC* (2010), arXiv:1006.0308.

[19] BUYYA, R., RANJAN, R., AND CALHEIROS, R. N. Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit - Challenges and Opportunities. *PDPTA cs.DC*, Chapter 4 (2009), 24–.

[20] 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.

[21] CHEN, M., ZHANG, H., SU, Y.-Y., WANG, X., JIANG, G., AND YOSHIHIRA, K. Effective VM sizing in virtualized data centers. In *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)* (2011), IEEE, pp. 594–601.

[22] 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.

[23] 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.

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

[25] CONSTANTIN, I. Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. *International Transactions in Operational Research 2*, 2 (Apr. 1995), 149–164.

[26] COWLING, P., KENDALL, G., AND SOUBEIGA, E. A Hyperheuristic Approach to Scheduling a Sales Summit. In *Practice and Theory of Automated Timetabling III*. Springer, Berlin, Heidelberg, Berlin, Heidelberg, Aug. 2000, pp. 176–190.

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

[28] DEB, K., AND 0001, A. S. Solving Bilevel Multi-Objective Optimization Problems Using Evolutionary Algorithms. *EMO 5467*, 1 (2009), 110–124.

[29] 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.

[30] 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.

[31] DEMPE, S., DUTTA, J., AND LOHSE, S. Optimality conditions for bilevel programming problems. *... with Multivalued Mappings 55*, 5-6 (Oct. 2006), 505–524.

[32] DENG, X. Complexity Issues in Bilevel Linear Programming. In *Multilevel Optimization: Algorithms and Applications*. Springer, Boston, MA, Boston, MA, 1998, pp. 149–164.

[33] 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.

[34] FALKENAUER, E. A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics 2*, 1 (1996), 5–30.

[35] FAN, X., WEBER, W.-D., AND BARROSO, L. A. Power provisioning for a warehouse-sized computer. *ISCA* (2007), 13.

[36] FEHLING, C., LEYMANN, F., RETTER, R., SCHUPECK, W., AND ARBITTER, P. Cloud Computing Patterns - Fundamentals to Design, Build, and Manage Cloud Applications.

[37] FELLER, E., RILLING, L., AND MORIN, C. Energy-Aware Ant Colony Based Workload Placement in Clouds. *GRID* (2011).

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

[39] 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.

[40] FERRETO, T. C., NETTO, M. A. S., CALHEIROS, R. N., AND DE ROSE, C. A. F. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems 27*, 8 (Oct. 2011), 1027–1034.

[41] 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.

[42] GANESAN, R., SARKAR, S., AND NARAYAN, A. Analysis of SaaS Business Platform Workloads for Sizing and Collocation. *IEEE CLOUD* (2012), 868–875.

[43] 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.

[44] 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.

[45] HERSKOVITS, J., LEONTIEV, A., DIAS, G., AND SANTOS, G. Contact shape optimization: a bilevel programming approach. *Structural and Multidisciplinary Optimization 20*, 3 (2000), 214–221.

[46] 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.

[47] HOLT, C. C. Author's retrospective on 'Forecasting seasonals and trends by exponentially weighted moving averages'. *International journal of forecasting 20*, 1 (Jan. 2004), 11–13.

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

[49] JOHNSON, D. S. Vector Bin Packing.

[50] 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.

[51] KAPLAN, J. M., FORREST, W., AND KINDLER, N. Revolutionizing data center energy efficiency, 2008.

[52] KHANNA, G., BEATY, K. A., KAR, G., AND KOCHUT, A. Application Performance Management in Virtualized Server Environments. *NOMS* (2006).

[53] KIRJNER-NETO, C., POLAK, E., AND DER KIUREGHIAN, A. An Outer Approximations Approach to Reliability-Based Optimal Design of Structures. *Journal of Optimization Theory and Applications 98*, 1 (1998), 1–16.

[54] KIVITY, A., KAMAY, Y., LAOR, D., AND LUBLIN, U. kvm: the Linux virtual machine monitor. *... of the Linux ...* (2007).

[55] KOZA, J. R. Genetic programming - on the programming of computers by means of natural selection. *Complex adaptive systems* (1993).

[56] KUNKLE, D., AND SCHINDLER, J. A Load Balancing Framework for Clustered Storage Systems. *HiPC 5374*, 1 (2008), 57–72.

[57] 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.

[58] 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.

[59] 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.

[60] LEINBERGER, W., KARYPIS, G., AND KUMAR, V. Multi-Capacity Bin Packing Algorithms with Applications to Job Scheduling under Multiple Constraints. *ICPP* (1999), 404–412.

[61] 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.

[62] LIEN, C.-H., BAI, Y.-W., AND LIN, M.-B. Estimation by Software for the Power Consumption of Streaming-Media Servers. *IEEE Trans. Instrumentation and Measurement 56*, 5 (2007), 1859–1870.

[63] LIU, H., JIN, H., XU, C.-Z., AND LIAO, X. Performance and energy modeling for live migration of virtual machines. *Cluster computing 16*, 2 (2013), 249–264.

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

[65] 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.

[66] MANVI, S. S., AND SHYAM, G. K. Resource management for Infrastructure as a Service (IaaS) in cloud computing - A survey. *J. Network and Computer Applications 41* (2014), 424–440.

[67] MATHIEU, R., PITTARD, L., AND ANANDALINGAM, G. Genetic algorithm based approach to bi-level linear programming. *RAIRO-Operations Research* (1994).

[68] MATHIEU, R., PITTARD, L., AND ANANDALINGAM, G. Genetic algorithm based approach to bi-level linear programming. *RAIRO-Operations Research 28*, 1 (Mar. 2011), 1–21.

[69] 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.

[70] MENG, X., ISCI, C., KEPHART, J. O., 0002, L. Z., BOUILLET, E., AND PENDARAKIS, D. E. Efficient resource provisioning in compute clouds via VM multiplexing. *ICAC* (2010), 11.

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

[72] MISHRA, M., AND SAHOO, A. On Theory of VM Placement - Anomalies in Existing Methodologies and Their Mitigation Using a Novel Vector Based Approach. *IEEE CLOUD* (2011), 275–282.

[73] MURTAZAEV, A., AND OH, S. Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing. *IETE Technical Review 28*, 3 (Sept. 2014), 212.

[74] 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.

[75] ODUGUWA, V., AND ROY, R. Bi-level optimisation using genetic algorithm. In *2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002)* (2002), IEEE Comput. Soc, pp. 322–327.

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

[77] PANWALKAR, S. S., AND ISKANDER, W. A Survey of Scheduling Rules. *Operations Research 25*, 1 (1977), 45–61.

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

[79] 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.

[80] POLI, R., WOODWARD, J., AND BURKE, E. K. A histogram-matching approach to the evolution of bin-packing strategies. In *2007 IEEE Congress on Evolutionary Computation* (2007), IEEE, pp. 3500–3507.

[81] QIN, X., ZHANG, W., 0049, W. W., WEI, J., ZHAO, X., AND HUANG, T. Towards a Cost-Aware Data Migration Approach for Key-Value Stores. *CLUSTER* (2012).

[82] RADCLIFFE, N. J. Forma Analysis and Random Respectful Recombination. *ICGA* (1991).

[83] 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.

[84] 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.

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

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

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

[88] SIM, K., AND HART, E. Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. *GECCO* (2013), 1549.

[89] SINHA, A., MALO, P., AND DEB, K. Efficient Evolutionary Algorithm for Single-Objective Bilevel Optimization.

[90] 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.

[91] 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.

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

[93] SPROTT, D., AND WILKES, L. Understanding service-oriented architecture. *The Architecture Journal* (2004).

[94] 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.

[95] SUN, H., GAO, Z., AND WU, J. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied mathematical modelling 32*, 4 (Apr. 2008), 610–616.

[96] 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.

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

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

[99] 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.

[100] VAQUERO, L. M., RODERO-MERINO, L., AND BUYYA, R. Dynamically scaling applications in the cloud. *Computer Communication Review 41*, 1 (2011), 45.

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

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

[103] VISWANATHAN, B., VERMA, A., AND DUTTA, S. CloudMap - Workload-aware placement in private heterogeneous clouds. *NOMS* (2012), 9–16.

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

[105] WANG, G., WAN, Z., WANG, X., AND LV, Y. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications 56*, 10 (Nov. 2008), 2550–2555.

[106] WANG, Y., AND 0002, Y. X. Energy Optimal VM Placement in the Cloud. *CLOUD* (2016), 84–91.

[107] WANG, Y., JIAO, Y.-C., AND LI, H. An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme. *IEEE Trans. Systems, Man, and Cybernetics, Part C 35*, 2 (2005), 221–232.

[108] WANG, Y., LI, H., AND DANG, C. A New Evolutionary Algorithm for a Class of Nonlinear Bilevel Programming Problems and Its Global Convergence. *INFORMS Journal on Computing 23*, 4 (2011), 618–629.

[109] Wilcox, D., McNabb, A. W., and Seppi, K. D. Solving virtual machine packing with a Reordering Grouping Genetic Algorithm. *IEEE Congress on Evolutionary Computation* (2011), 362–369.

[110] Wood, T., Shenoy, P. J., Venkataramani, A., and Yousif, M. S. Sandpiper - Black-box and gray-box resource management for virtual machines. *Computer Networks 53*, 17 (2009), 2923–2938.

[111] 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.

[112] Xavier, M. G., Rossi, F. D., De Rose, C. A. F., Calheiros, R. N., and Gomes, D. G. Modeling and simulation of global and sleep states in ACPI-compliant energy-efficient cloud environments. *Concurrency and Computation - Practice and Experience 29*, 4 (Feb. 2017), e3839.

[113] 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.

[114] Xie, J., Jiang, S., Xie, W., and Gao, X. An Efficient Global K-means Clustering Algorithm. *JCP 6*, 2 (2011).

[115] Xiong, A.-p., and Xu, C.-x. Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center. *Mathematical Problems in Engineering 2014*, 6 (2014), 1–8.

[116] Xu, J., and Fortes, J. A. B. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. *GreenCom/CPSCom* (2010).

[117] 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.

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

[119] 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.

[120] 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.

[121] 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.