Energy-efficient Server Consolidation in Container-based Clouds with Evolutionary Computation A.pproaches

Boxiong Tan

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

Submitted in partial fulfilment of the requirements for PhD.

Abstract

A container-based cloud is a new trend in Cloud computing that introduces more granular management of applications and reduces overheads of virtual machine (VM). Compared with VM-based clouds, container-based clouds can further improve the energy efficiency with a finer granularity server consolidation in data centers. Current VM-based single level of server consolidation cannot be used in container-based clouds because container-based clouds have two levels of placement: container to VM and VM to Physical machine. Existing research lacks energy model and optimization algorithms that consider the joint allocation of container and VM. This work aims to improve energy efficiency in container-based cloud by proposing two bilevel energy models and three Evolutionary Computation (EC)-based optimization algorithms for three placement decision scenarios: initial placement of applications, periodic placement of applications, and dynamic placement of applications. The two novel bilevel energy models and three EC algorithms will contribute to better management of resources for energy efficiency in container-based clouds.

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 organization 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) [11]. Compared with the traditional software industry, where applications run on individual hardware, on-demand cloud computing provides a cluster of servers for the software industry. For example, web service providers such as Google and Neflix deploy their applications on the cloud. These web service providers do not need to purchase and maintain hardware resources. In addition, they do not need to worry about scalability and availability issues when demands of their applications increase. Cloud dynamically increases the capacity of applications and cloud computing services can be accessible 99.99% of the time [1]. Moreover, application users 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 [31]. This huge energy consumption has become the major expense of cloud providers. It is necessary to find ways to reduce the energy bills. The reduction of energy bills would benefit cloud providers, web service providers and environment. Furthermore, people would pay less to applications on the cloud.

Generally, reducing the energy consumption in clouds depends on improving the resource utilization of live physical machines (PMs) such as servers. Studies show [8, 102], PMs account for the majority – more than 40% – of energy consumption in clouds among other components such as cooling systems and network devices. Moreover, these PMs have been not used effectively. Some studies analyzed the PMs' average utilization. They found that the average utilization is quite low – from 10% to 50% [142]. Therefore, we can reduce energy by improving the utilization of PMs and reducing the usage of PMs in clouds.

The common way to improve the utilization of PMs in clouds is through resource management of PMs [79] (see Figure 1.1). A centralized resource management system in clouds has two main functionalities. First, the management system allocates resources such as CPUs and memories of PMs for cloud users to run applications. Second, the management system handles the workload fluctuations to reduce the number of potential migrations. These two main functionalities deploy and maintain applications in clouds.

The two main functionalities of resource management of PMs in clouds involve four

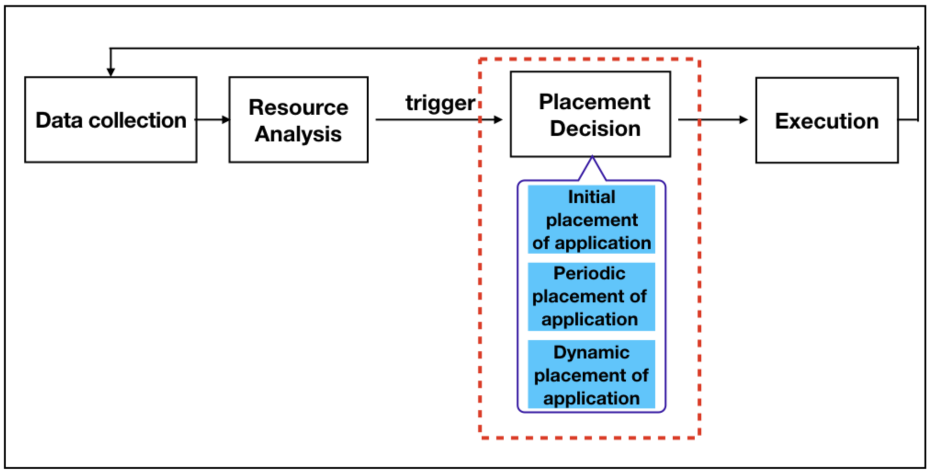

Data collection
Resource Analysis
trigger
Placement Decision
Execution
Initial placement of application
Periodic placement of application
Dynamic placement of application

Figure 1.1: A workflow of resource management [83]

steps. First, the management system collects the utilization information of PMs and then analyzes the usage of PMs and the resources needed by applications. Next, triggered by resource analysis, the management system determines the placement of applications. The placement of applications includes three scenarios: initial placement of applications for new applications; periodic placement of applications for adjusting the running applications periodically; and dynamic placement of applications for adjusting applications in a fast manner when abnormal events such as overloading happen. Finally, the management system executes the placement decision of applications to PMs. Hence, better management of resources in clouds contributes towards a better utilization of PMs and thus a reduction of energy consumption.

The core strategy of resource management is server consolidation [118]. Two different types of server consolidation are used in clouds: static and dynamic. Static server consolidation manages resources in an off-line fashion. It is mainly used in the initial placement of applications and periodic placement of applications. Dynamic server consolidation manages resources in an on-line fashion, which is used in the dynamic placement of applications. Both static and dynamic server consolidation aim to place applications in fewer PMs. This leads to fewer PMs with higher utilization and lower energy consumption.

Currently, the resource management in clouds is based on *virtualization* technology [116] and the mainstream is virtual machine-based virtualization. Such virtualization separates the resources (e.g. CPUs and RAMs) of a PM into several parts called *virtual machines (VMs)*. Each VM runs an isolated operating system. This VM-based technology is very different from traditional clouds that place each application to a single PM and lead to the low reserved utilization of PMs. Compared with traditional clouds, current VM-based clouds significantly improve the utilization of PMs and reduce energy consumption.

However, in recent years, virtual machine-based virtualization cannot catch up with a new trend in the software industry – Service Oriented Architecture (SOA) [108]. This SOA is widely used in the modern software industry because of its agility and re-usability [108]. SOA separates a centralized application into multiple distributed components called web services. As web services only require a small amount of resources (e.g. 15% of a typical CPU), 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 SOA and further reduce energy consumption, a new container-based virtualization [40, 106] has been proposed. Containers running on top of VMs are called an operating system (OS) level of virtualization [106]. 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 with each other. In addition, containers naturally support *vertical scaling* (change size during runtime) [117]. The vertical scaling provides resilient resources to fluctuate workloads. The container technology provides a new architecture for allocating applications and a finer granularity of resource management. Hence, containers have the potential to further improve the energy efficiency.

Although the efficient use of containers can improve the utilization of VMs, containers bring new challenges and difficulties to server consolidation [109]. We cannot directly apply current VM-based server consolidation strategies on container-based clouds because of the different placement structures of applications. Moreover, to support the increasing size of containers, the vertical scaling in clouds requires the VMs to reserve sufficient resources. The interaction between VMs and containers changes the server consolidation into a bilevel problem: VMs-to-PMs, and VMs-to-containers. Bilevel problems are NP-hard [104]. NP-hard means there is no polynomial time algorithm to find the global optimum solution. Therefore, we need to develop better algorithms for the bilevel problem.

This research aims to improve the energy efficiency in container-based clouds by proposing new bilevel energy models and server consolidation algorithms for three placement decision scenarios: initial placement of applications, periodic placement of applications, and dynamic placement of applications.

1.2 Motivation

A container-based cloud is a promising technology to support SOA, the new trend in software industry. container-based clouds provide a finer granularity management of applications compared with VM-based clouds [109]. Such finer granularity management improves the utilization of resources and minimizes energy consumption of data centers. Therefore, container-based clouds not only increase the profit for cloud providers but reduces the cost of cloud users as well.

Even though container-based clouds has many advantages to achieve energy-efficiency, it is more challenging in optimizing the placement of applications than in VM-based clouds. **First**, we cannot use existing VM-based energy models for the container-based clouds. Current VM-based energy models represent a single level of placement: VM-PM while the container-based clouds need a two-level energy model representing the placement between container-VM and the placement between VM-PM. However, the two-level energy model is more complicated than the single-level energy model. **Second**, current VM-based optimization approaches [10, 76] formulate the placement of applications as a bin packing problem. Most VM-based approaches use bin-packing heuristics such as First Fit [50] that are designed for the problem of single-level placement. Since container-based clouds have two-level placement, bin-packing heuristics can rarely achieve the global optimal solution in container-based clouds.

In recent years, Evolutionary Computation (EC)-based approaches have been applied to bilevel problems in other areas such as logistics distribution centers and have shown promising results [4, 33, 105]. However, EC-based approaches have not been used to solve the placement of applications in container-based clouds. Therefore, we need to study how to design representations and specific genetic operators for adapting EC algorithms to container-based clouds. The above three issues exist in all three main scenarios of placement in container-based clouds.

Besides the above issues, other specific issues exist in three main scenarios of placement

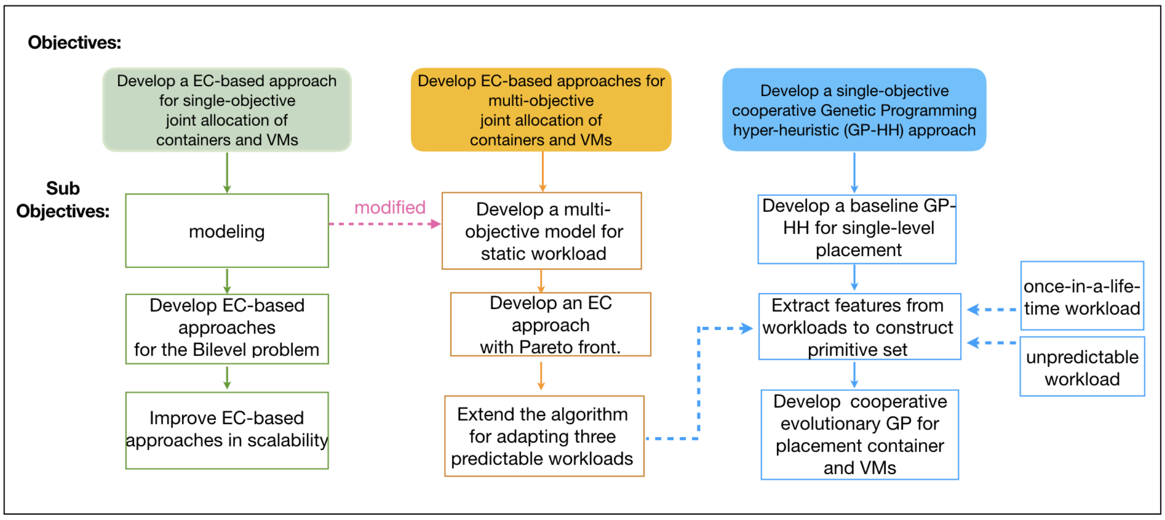

Objectives:
Develop a EC-based approach for single-objective joint allocation of containers and VMs
Develop EC-based approaches for multi-objective joint allocation of containers and VMs
Develop a single-objective cooperative Genetic Programming hyper-heuristic (GP-HH) approach
Sub Objectives:
modeling
modified
Develop a multi-objective model for static workload
Develop a baseline GP-HH for single-level placement
Develop EC-based approaches for the Bilevel problem
Develop an EC approach with Pareto front.
Extract features from workloads to construct primitive set
once-in-a-life-time workload
unpredictable workload
Improve EC-based approaches in scalability
Extend the algorithm for adapting three predictable workloads
Develop cooperative evolutionary GP for placement container and VMs

Figure 1.2: Relationship between objectives

of applications. **Initial placement of applications** deploys applications when data centers receive a number of requests. We need to investigate a scalable EC-based optimization algorithm to further minimize the energy consumption when placing over one thousand of applications in PMs. **Periodic placement of applications** migrates applications to fewer PMs to optimize energy consumption and migration cost with consideration of workload. We need to take workload into account in a new model for multi-objective (energy and migration cost) optimization. Meanwhile, we will investigate how to design an EC-based approach to solve the multi-objective bilevel placement. **Dynamic placement of applications** is capable of dynamically allocating applications when abnormal events such as overloading and under-loading [12] happen in data centers. Unlike the previous two scenarios, dynamic placement requires fast allocation for optimizing energy consumption. A genetic programming [5] hyper-heuristic (GP-HH) is a promising technology that has been used in many dynamic problems such as dynamic job shop scheduling [86]. GP-HH is based on learning patterns from previous good solutions and can automatically generate heuristics. Hence, GP-HH can allocate applications quickly and optimally in terms of energy consumption. However, the complex learning patterns from previous good solutions are challenging in dynamic placement because different types of workload exist and some of them are unpredictable. In summary, all these issues are critical and need solving in order to achieve energy efficiency in container-based clouds. Therefore, we need to design bilevel energy models and applying EC-based optimization algorithms for the placement of applications in container-based clouds.

1.3 Research Goals

The overall goal of this research is to optimize energy consumption of container-based clouds using EC-based approaches for three placement decision scenarios: initial placement of applications, periodic placement of applications, and dynamic placement of applications. 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 initial placement of applications

The goal of objective one is to minimize energy consumption in initial placement of applications at container-based clouds. We set three sub-objectives to achieve this goal. For the sake of brevity and without loss of clarity, we will use "the bilevel model/problem" to replace "the joint placement of containers and VMs model/problem" in the following content.

1. Sub-objective 1: Develop a new bilevel energy model to represent the relationship between five factors and energy consumption. The five factors involve locations of container, types of VM, locations of VM, overheads of VM, and the balance between memory and CPU. We need to consider the interaction between these factors.

   The major challenge of this sub-objective is that the bilevel energy model is more complicated than a single-level energy model. **First**, in a VM-based model, the only factor is the locations of VMs in PMs. In contrast, in a container-based model, the energy model involves four more variables. Specifically, locations of container decide the utilization of VMs. Types of VM constrain the placement of container. Overheads of VM bring extra workloads to the PMs. The balance of CPU and memory may indirectly impact on the energy consumption. Mishra [84] finds better balance leads to a higher probability of allocating more applications. The above mentioned five factors are very likely to affect the energy consumption. **Second**, two pairs of interaction have not been considered. The first pair is the interaction between containers and types of VM. The second pair is the interaction between locations of container and locations of VM. Therefore, the bilevel energy model is very complicated.

   In order to establish a bilevel energy model, we propose to follow three steps and gradually add factors to the model. Because the correlations between the five factors are complicated, we will study their correlation pair by pair. **First**, we will consider the relationship between overheads, types of VM, and numbers of VM because they are closely related. In order to study the overheads of VM, we will review some research about VM hypervisors and study the impact of VMs. Our hypothesis is that the overheads have a linear relationship with the number of VMs and have no correlation with the type of VM. The outcome of the study of overheads will be a formulation that describes the relationship between overheads, types of VM, and numbers of VM.

   Second, we will consider the balance between CPU and memory in both VM and PM levels. No one has yet considered the balance in the bilevel problem. We will first consider formulating the balance in both levels. However, this formulation may have a high computation complexity. Hence, we will consider an aggregation of resources from both levels. Our hypothesis is that the aggregation can also achieve high utilization in PMs. The outcome of the study will be a formulation that describes the balance between CPU and memory in two levels.

   Third, we will consider the energy consumption with container, VM, and PM. We will review some pieces of VM-based research [41, 46, 134]. The VM-based research generally represents resource demands as resource utilization. We can use the similar idea to represent containers. The outcome of the study of overheads will be a formulation that describes the relationship between container, VM, PM, and energy consumption.

   This sub-objective 1 is expected, therefore, to formulate a bilevel energy model to represent the relationship between five factors.

2. Sub-objective 2: Propose a new EC-based bilevel optimization approach to solve the initial placement of applications to achieve better energy efficiency than VM-based ap-

proaches.

We need to solve two challenges. **The first challenge** is to design a representation of multiple variable types for EC-based optimization approach. Current VM-based representations only contain one type of variable while container-based representations contain three types of variables: locations of container, locations of VM and types of VM. Furthermore, current VM-based research generally applies binary representation (use 0 and 1 to represent placement). However, in container-based clouds, we cannot apply binary representation because it is not straightforward to represent bilevel of placement. In addition, it is difficult to design a representation that narrows down the search space of solutions. **The second challenge** is to design genetic operators and search mechanisms for an EC-based optimization approach. Since a bilevel optimization problem is known as a NP-hard problem [80], the landscape of search space is ragged because of non-linearity, non-differentiability, non-convexity etc. Therefore, we need to further explore an effective search mechanism that can quickly locate feasible solutions.

In order to solve the above challenges, we can learn from some existing approaches and design our bilevel approaches. For the representation, we can review some VM-based research that uses direct discrete representation [134] and indirect continuous probability representation [133]. We can learn from their approaches and then design our own such as the representation we design in preliminary work (Section 3.3.1). Furthermore, in order to quickly locate feasible solutions, we can embed a heuristic in the representation like in Section 3.3.1. Based on our designed representation, we need to first investigate a few EC-based bilevel algorithms such as Genetic Algorithm (GA)-based approach: Cooperative coevolution algorithm [71], Particle Swarm Optimization (PSO)-based approach: Nested PSO [73] and others [4, 139]. Then, we will choose an algorithm and design the genetic operators of the algorithm. For example, for GA-based algorithm, initialization operators should generate a diverse set of population and, ideally, the population contains feasible solutions. Mutation operator should allow the population to explore the entire solution space. Crossover operator creates better solutions based on the good genes in parents (previous good solutions).

We will evaluate our approaches in two ways. **First**, in order to find the most suitable representation and EC-based algorithm, we will conduct experiments on different approaches and compare their performances in terms of energy efficiency. **Second**, we will compare our algorithm with existing VM-based approaches on the same benchmark dataset [102].

This sub-objective 2 is expected to propose an EC-based approach for the initial placement of applications. We expect this approach to achieve better energy efficiency than existing VM-based approaches [126, 134].

3. Sub-objective 3: Improve the scalability of the proposed EC-based approach a large number of applications.

   The major challenge of this task is the scalability of bilevel problems and the long computation time of EC-based approaches [105].

   We can use two methods: bottom-up and top-down to improve the scalability and decrease the computation time. **First**, for the bottom-up method, we can reduce the number of variables by combining small containers into large combinations of containers. In the first step, we can use clustering approaches such as K-means [131] and decision tree to categorize containers into major groups and label them as “CPU intensive”, “Memory intensive” etc. An open question is “what kind of features should we use in clustering”. In the second step, we will choose containers from these groups to construct combinations. An open question is “what combination rule should we use”. Greedy-based rules may be useful to quickly construct combinations. In the third step is to decide the size of the constructed combination. Intuitively, large combinations (fewer variables) lead to lower computation complexity. On the other hand, larger combinations may lead to worse energy efficiency because the combinations are local. Therefore, a trade-off exists in computation complexity and energy efficiency.

Second, for the top-down method, we can separate a large problem into several sub-problems so that they can be executed in parallel. Similar to the bottom-up method, the top-down method also has a trade-off in computation complexity and energy efficiency. The divide-and-conquer approach is a top-down method. After the divide-and-conquer approach splits a large problem into sub-problems, the smaller sub-problems are easy to solve individually. However, the aggregation of results from sub-problems may lead to local optima. Therefore, we will investigate how to split the problem and the size of sub-problems.

We will evaluate our approach by comparing the energy efficiency and execution time with the previously proposed EC-based approach.

This sub-objective 3 is expected to improve the scalability of the previously proposed EC-based approach for up to one thousand applications without decreasing the energy efficiency.

1.3.2 Objective Two: Develop an EC-based approach for the multi-objective periodic placement of applications

The goal of the second research objective is to develop a multi-objective EC-based approach for periodic placement of applications at container-based clouds with consideration of various types of workload. The two objectives to be minimized are the energy consumption and the migration cost. We set three sub-objectives to achieve this goal.

1. Sub-objective 1: Propose a multi-objective bilevel model for periodic placement with consideration of static workloads. It has two objectives to be optimized: minimizing the energy consumption and minimizing the migration cost.

   The main challenge is that the migration model in container-based clouds is much complicated than the migration model in VM-based clouds. **First**, in VM-based clouds, many research only considers the number of migrations [85]. In container-based clouds, because the sizes of container vary in a wide range (e.g from MBs to GBs), it is inappropriate to simplify the migration cost as the number of migrations. Therefore, we will consider more factors such as network bandwidth to model the migration cost. This is a difficult task. **Second**, container-based migration model not only considers migration of VM, but also migration of container. Therefore, adding the migration model into our previous bilevel model is difficult.

   In order to solve these challenges, we will thoroughly study the differences of migration between VMs and containers. For example, VM contains an OS and all applications' data while containers only include data of an application. Then, we will explore some VM-based migration models that consider the size of VM and network bandwidth. We will consider to use a uniform model to represent the cost of migrating VM and container.

The sub-objective 1 is expected to formulate the bilevel model to represent both energy consumption and migration cost in container-based clouds.

2. Sub-objective 2: Propose a multi-objective bilevel EC-based algorithm for periodic placement of applications with Pareto front approach to achieve better performance than VM-based approach in both energy efficiency and migration cost.

   We need to solve three challenges. **First**, the relationship between the two optimization objectives is very complicated. Since both migration cost and energy consumption can be non-convex, we cannot easily imagine the trade-off between two objectives. **Second**, we will design genetic operators to adapt the bilevel multi-objective optimization. Specifically, genetic operators in both levels collaborate with each other. Hence, it is more difficult to design the collaboration of operators between two levels. **Third**, after obtaining a set of non-dominated solutions, we need to further design a fitness function assessment. The assessment selects one solution from a large number of non-dominated solutions based on some predefined preferences from Cloud providers. The selection of a solution can be a difficult task for a large size of non-dominated set [141].

   We will follow three steps to solve the above challenges. **First**, to study the relationship between two objectives, we will first analyze the fitness functions of the objectives. For example, it is likely that in the migration function, the migration cost is monotonically increasing with the number of migrations and size of containers. After analyzing the fitness functions, we can visualize the relationship by using a technique called Trade-off region map [91]. The trade-off region map can identify local and complex relationships between objectives, gaps in the fitness landscape, and regions of interest. **Second**, we will first choose an EC-based algorithm based on our previous experience from the objective one. Different from objective one, we will add trade-off control mechanisms such as non-dominated sorting into the genetic operators. **Third**, in order to design a fitness function assessment, we will review some priori and posteriori decision making support methods. On one hand, for priori methods, we assume Cloud providers have preferences according to the energy efficiency and migration cost. On the other hand, for posteriori, we can design a clustering algorithm on the solution set and further narrow down solutions [140].

   We will evaluate our algorithm by comparing with existing VM-based approaches on the same benchmark dataset in terms of the energy consumption and migration cost.

   This sub-objective 2 is expected to propose a multi-objective bilevel EC-based approach that can achieve better energy efficiency and migration cost than VM-based approaches.

3. Sub-objective 3: Extend the bilevel multi-objective algorithm for adapting three types of predictable workloads [38]: static, linear continuously changing, and periodic.

   Several research mention the fluctuation of workloads has a heavy impact on the periodic placement [38, 119]. Verma [119] states that treating workloads as static leads to a performance degradation of applications and low utilization of PMs. Verma considers three types of workloads: static, periodic, and unpredictable while Fehling [38] considers three types of predictable workload: static, periodic and linear continuously changing, and two types of unpredictable workload: once-in-a-life-time and irregular. We believe Fehling's classification has a better coverage of existing workloads. Thus, we extend periodic placement on predictable workloads: static, periodic, and linear continuously changing to reach a more stable consolidation (less future migrations). Furthermore, because periodic and linear continuously changing workloads have distinct characteristics, for example, peak vs. no-peak, stable vs. unstable (constant mean value). Hence, we will study periodic and linear continuously changing workloads separately.

To consider various types of workload, three challenges exist in this sub-objective. First, it is difficult to reduce the dimensionality of raw workload dataset. Since a typical workload dataset may contain thousands of data point, the dimensionality reduction is critical before applying any similarity measure between workloads. Second, it is non-trivial to define the similarity between two workloads. The current measurement [119] only gives a correlation based on two workloads while we will measure the resource demand of an arbitrary number of workloads. Third challenge is to design representations and genetic operators to adapt to three types of workload.

To tackle these challenges, **first**, we can apply time-series analysis techniques on workloads such as Independent component analysis (ICA). ICA can extract features from large dimensionality of raw data so that three types of workload can be represented according to their features. **Then**, we can use a Pearson correlation to represent the similarity between two periodic workloads. **Third**, we will first try to design a uniform representation for all workloads so that all types of workload can be treated the same. However, if we cannot design a uniform representation, we may define a unique representation for each type of workload. Accordingly, we will design the genetic operators to suit the representations.

The sub-objective 3 is expected to extend a multi-objective bilevel EC-based approach to three types of workload.

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

The goal of 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 five types of workloads. We set three sub-objectives to achieve this goal step by step.

1. Sub-objective 1: Develop a GP-based hyper-heuristic (GP-HH) algorithm for automatically generating dispatching rules for the single-level of placement: VM to PM with static workloads.

   There are three reasons for us to develop a GP-HH for single-level placement. **First**, previous dynamic consolidation methods, including both VM-based and container-based, are mostly based on bin-packing heuristics such as First Fit Descending. As Mann's research [77] shown, server consolidation is much harder than a bin-packing problem because of multi-dimensional of resources and many constraints. Therefore, general bin-packing algorithms do not perform well with many constraints and specific designed heuristics only perform well in a very narrow scope. **Second**, GP has been used in automatically generating dispatching rules in many areas such as job shop scheduling [86]. GP also has been successfully applied in bin-packing problems [21]. Therefore, we will investigate GP approaches for solving the dynamic consolidation problem. **Third**, we will first design a GP-HH for single-level of placement so that the two GP-HHs for bilevel of placement can cooperate.

Two major challenges, **first**, it is difficult to design a hyper-heuristics to generate dispatching rules. We will explore meaningful features of static workload, VM, and PM to construct a primitive set. It is uncertain that which feature can contribute to the dispatching rules. **Second**, it is even difficult for the generated dispatching rules to reach a global optimal solution because of the trade-off between training accuracy and over-fitting.

To gradually solve above challenges, **first**, we start from reviewing the literature about manually designed heuristics and simple bin-packing heuristics. These heuristics contain some criteria, for example, First Fit algorithm normally considers the remaining resources of PM as the criteria of selecting a placement. We will also consider other features such as the status of VMs (e.g resource utilization), features of workloads (e.g resource requirement) that will affect the placement decision. The above features are represented as resource utilization. We will construct a primitive set with the selected features. **Second**, we will use the default function set and the basic tree representation for constructing the dispatching rules. **Third**, to train the GP-HH, we will use the solutions from the first objective as the learning sets. We can use ten folds cross-validation to prevent from over-fitting.

In order to evaluate the automatically generated heuristics, we will use a widely used simulator called CloudSim [23]. Since our proposed algorithm is equivalent to a VM-based placement algorithm because they both focus on single-level of placement. We will compare our heuristic to a highly cited work [9] from Beloglazov who proposes a Best Fit Decreasing heuristic for the energy consumption problem.

The sub-objective 1 is expected to propose a GP-HH algorithm for automatically generating dispatching rules for the single-level placement with static workloads. The generated dispatching rules are expected to achieve an equal or better performance than manually designed heuristics in terms of energy efficiency.

2. Sub-objective 2: Conduct feature extraction on the three predictable workloads and two unpredictable workloads to construct a new primitive set.

The major challenge is to discover useful features that can represent various workloads. The first challenge is to identify features from high dimensionality of datasets. The raw workload dataset may contain a large number of data points. It is difficult to extract meaningful features from the data points. The second challenge is to select useful features from a large number of raw features. It is very time consuming to manually examine raw features. The correlations between these raw features bring additional complexities.

To solve these challenges, **first**, we will review a number of dimensionality reduction techniques. Some possible techniques are sampling, extrema extraction. **Second**, we may employ the principle component analysis (PCA) technique on the feature set to reduce the number of features.

Classification of various workloads is one of the way to test the extracted features. If we can differentiate workloads based on the extracted features with a high accuracy (e.g. 95%), we can use the features to construct new dispatching rules.

The sub-objective 2 is expected to extract a number of features to represent different types of workload. We can apply these features to construct a new primitive set so that the dispatching rules can deal with all kinds of workload.

3. Sub-objective 3: Develop a cooperative GP-HH approach for automatically generating dispatching rules for placing both containers and VMs.
   The major challenge is to develop a cooperating procedure between two GP-HHs on two levels. As the first step, we will consider using two sub-populations to represent two-levels of placement. Each sub-population would be trained to minimize their objectives. The second step,the sub-population in the container-VM level will be used in training the VM-PM level of placement.

   This sub-objective 3 is expected to propose a cooperative GP-HH approach for automatically generating dispatching rules for dynamic placement in container-based clouds.

1.4 Published Papers

During the initial stage of this research, some investigation was carried out on the model of container-based server consolidation [112].

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

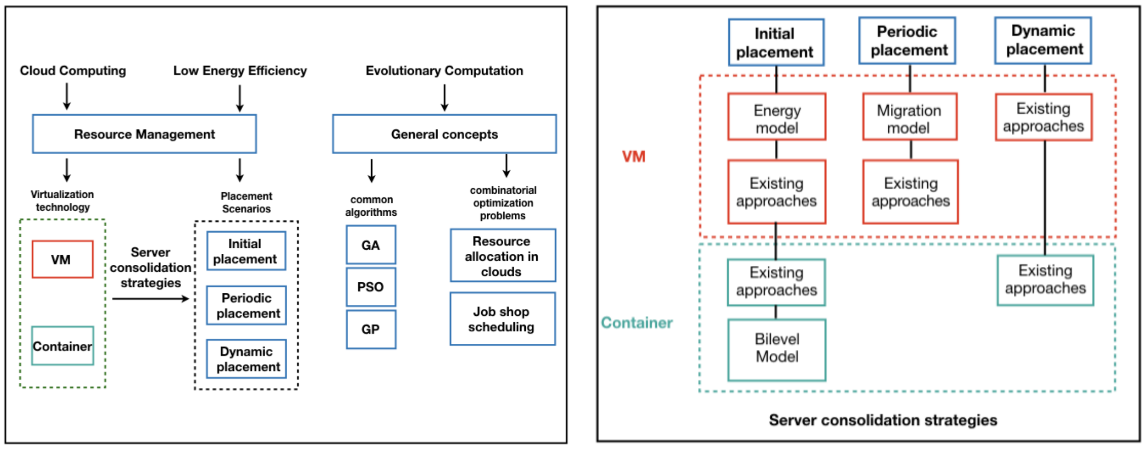
1.5 Organisation of Proposal

The remainder of the proposal is organised as follows: Chapter 2 provides a fundamental background of the resource management in cloud data centers and its the energy consumption problem. It also conducts 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

Background and Related Work

Figure 2.1: The scope of this chapter



(a) Background **(b)** Related Work

This chapter introduces the background of resource management in cloud computing, server consolication strategies and Evolutionary Computation (EC). In addition, the related work discusses server consolidation strategies related to three placement decision scenarios (see Figure 2.1). **Background** is divided into two parts. The first part introduces cloud computing, the energy efficiency problem, and resource management in cloud. We also explain virtualization technologies with three common placement scenarios and corresponding server consolidation strategies. The second part discusses three common EC algorithms: Genetic algorithm (GA), Particle Swarm Optimization (PSO), and Genetic Programming (GP). We also reviews several literature that applies EC algorithms in combinatorial optimization problems. **Related Work** discusses server consolidation strategies in terms of three placement scenarios with two virtualization technologies, VMs and containers.

2.1 Cloud Computing and Energy Efficiency Problem

Cloud computing is a computing model that offers a network of servers to their clients in a on-demand fashion. From NIST's definition [81], *"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 re-*

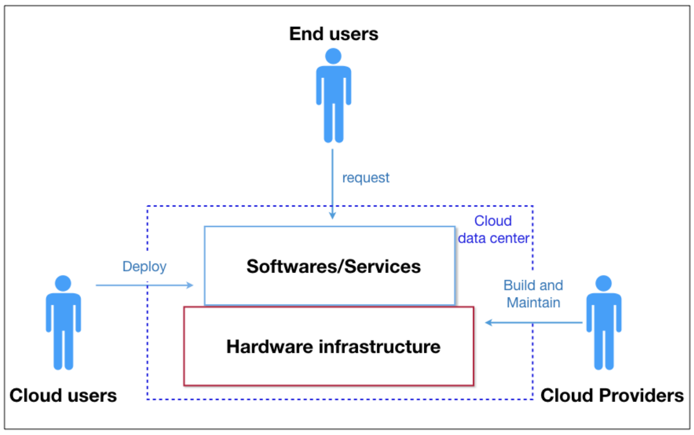


Figure 2.2: Stakeholders of Cloud computing [57]

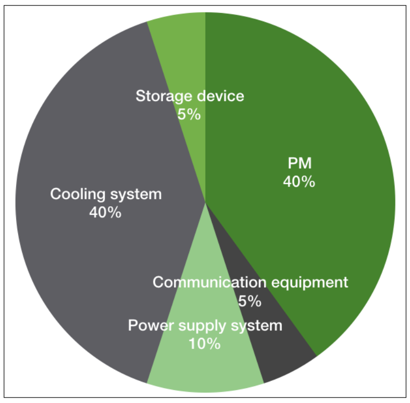


Figure 2.3: Energy consumption distribution of data centers [100]

leased with minimal management effort or service provider interaction." Hence, Cloud computing has major functionalities for providing facilities to cloud users.

To give an example of how Cloud computing works (see Figure 2.2), consider the case: a *Cloud provider* builds a data center containing thousands of servers. These servers connect with a network. To use these remote servers, *Cloud user* (e.g an application provider), 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. Cloud users charge fees from End users for using applications. Therefore, from cloud providers' perspective, both accommodating more applications and reducing energy consumption lead to the increase of profit.

The major expense of a cloud provider is energy consumption [59] and physical machines (PMs) (e.g servers) contribute to a majority of the energy. As shown in Figure 2.3, both the cooling system and physical machines (PMs) account for 40% of energy. However, PMs' energy efficiency are low on average [142]. The reason for low energy efficiency is the disproportion between the utilization of PMs and the energy consumption of PMs. For example, when CPU utilization of a PM is 15%, the energy consumption of the PM is 70% of its peak time. Therefore, cloud providers can reduce the energy consumption by improving the utilization of PMs.

In order to solve the low energy efficiency caused by low utilization of PMs, cloud providers use a resource management system to manage the applications.

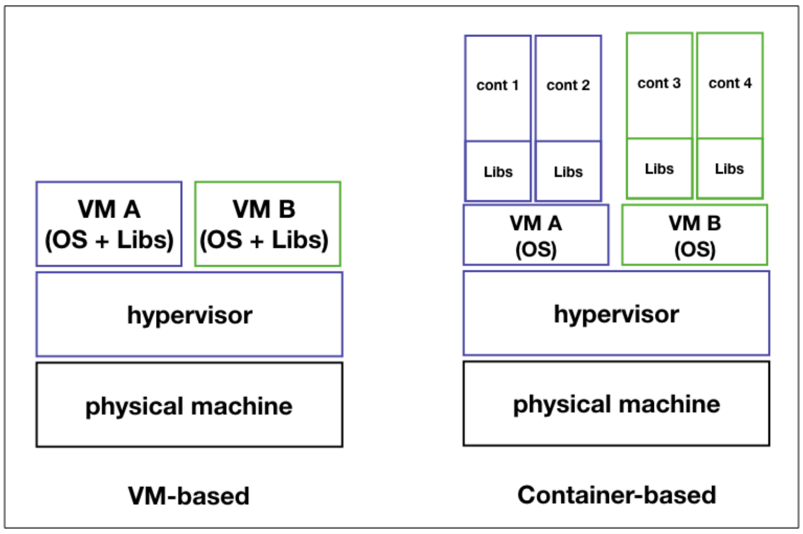


Figure 2.4: A comparison between VM-based and Container-based virtualization [92]

2.2 Cloud Resource Management

Cloud resource management is a process of allocating computation resources (e.g CPU and memory) to Cloud users to run their applications. Meanwhile, resource management aims to achieve low energy consumption in cloud [57].

Resource management applies two types of virtualization: virtual machine (VM) and container on four management steps (see Section 1.1): collecting PMs' utilization data, analyzing available PMs, deciding the placement of applications, and executing the placement. Furthermore, in the third step of placement decision, resource management has three common scenarios: initial placement, periodic placement, and dynamic placement. Resource management uses distinct server consolidation strategies on these placement scenarios based on different virtualiazation technologies to achieve energy efficiency.

This section will first introduce and compare two types of virtualization: VM-based and container-based and then illustrate three placement decision scenarios.

2.2.1 Virtualization Technologies

Resource management uses virtualization technologies [116] to achieve a finer granularity management than the traditional way. In comparison with traditional management – allocating an application to a PM – virtualized management partitions PM's resources (e.g. CPU, memory and disk) into several independent units and allocates applications into these units. The most common units are virtual machines (VMs) and containers.

Virtualization technology rooted back in the 1960s' was originally invented to enable isolated software testing. Because VMs can provide good isolation for applications running without interfering with each other [107]. Soon, people realized that virtualization can improve the utilization of hardware resources: with each application deployed in a VM, a PM can run multiple applications.

The next two sections illustrate two classes of virtualization (see Figure 2.4): VM-based and container-based virtualization and then compares the two virtualizations.

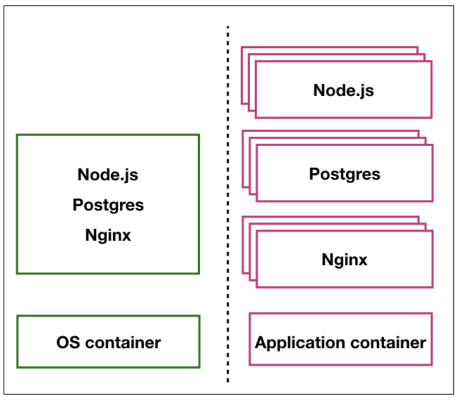


Figure 2.5: A comparison between OS container and Application container [94]

VM-based Virtualization

A VM-based virtualization has three-layers of structure: PM-Hypervisor-VM (see Figure 2.4 left-hand side). The hypervisor or the virtual machine monitor (VMM) is a software layer on top of PM. A hypervisor arbitrates accesses to the PM's resources so that VMs can share resources on the PM. Some implementations of VM-based hypervisors such as Xen [7], KVM [64], and VMware ESX [122] dominate this field in recent years. On of top of hypervisor, **VMs** are the fundamental resource management units. A VM allows independent Operating System (OS) to run on it.

In addition, hypervisors support dynamic migration techniques (e.g pre-copy [25] and post-copy [24]) that can move VMs from one PM to another. Therefore, resource management can improve the utilization of a PM by migrating VMs to that PM.

Container-based Virtualization

A Container-based virtualization has four-layers of structure: PM-Hypervisor-VM-Container (see Figure 2.4 right-hand side). Container-based virtualization is also addressed as operating-system-level virtualization because containers run on top of VMs. Specifically, container-based virtuliazation includes two classes: OS container and application container [94].

OS containers (Figure 2.5 left-hand side) have a one-on-one relationship with VM. Multiple applications run inside an OS container. Three implementations of OS-level of containers: OpenVZ, Google's control groups, and namespace [101] are widely used in Google and Facebook.

Application containers (Figure 2.5 right-hand side) have a many-to-one relationship with VM. A single application runs on an application container. Major implementations such as Docker, Rocket and Kubernetes [13] are very popular in the software industry.

In comparison with OS container, an application container is much more flexible because each container has its separated environment (e.g. libraries) for applications. Furthermore, application containers provide a finer granularity of resource management by enabling an application level of operations including deployment, scaling, and migration.

Notice that, in the following content, we use "container" to represent "application container". We do not discuss Os container because it is very similar to a VM.

Comparison between Container-based and VM-based Virtualization

This section summarizes three disadvantages of VM-based virtualization in terms of energy efficiency and shows how container-based can overcome these disadvantages based on several research [35, 40, 128].

Some main disadvantages in VM-based virtualization are listed as follows:

- Resource over-provisioning
  Cloud users tend to reserve more resources for ensuring the Quality of Service at peak hours [30] 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, for most of the time, resources are wasted.

- Unbalanced usage of resources
  Specific applications consume unbalanced resources which leads to a vast amount of resource wastage [114]. For example, computation intensive tasks consume much more CPU than RAM; a fixed type of VM provides much more RAM than it needs. Because the tasks use too much CPU, they prevent other tasks from co-allocating. This also causes wastage.

- Heavy overhead of VM hypervisors and redundant operating systems (OSs)
  Heavy overhead of hypervisors and redundant OSs running in the PM also 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.

Some key characteristics of containers can help overcome the above disadvantages of VMs.

The following two characteristics can overcome the heavy overhead of VM hypervisors and reduce the redundant OSs.

- Container-based virtualization has 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 separate OSs.

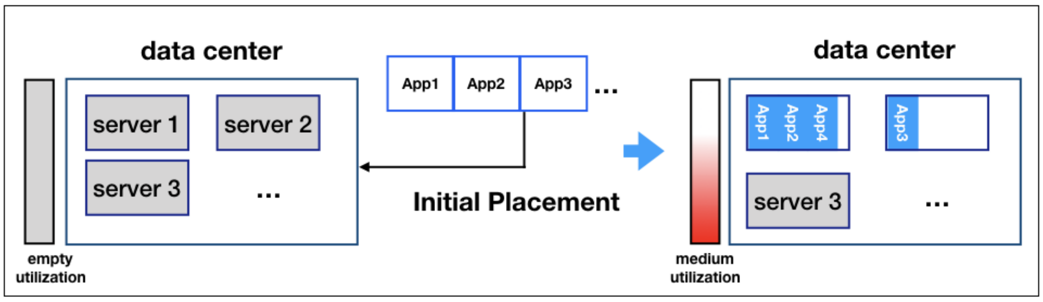
- Container-based virtualization naturally supports vertical scaling while VM-based virtualization does not. Vertical scaling means a container can dynamically adjust its resources under the host's resource constraint. This feature offers a fine granularity management of resources.

Vertical scaling can overcome the resource over-provisioning by dynamically adjusting the size of containers. Furthermore, the size of container reflects the requirement of the application. We can achieve a balanced usage of resources by using appropriate placement algorithm.

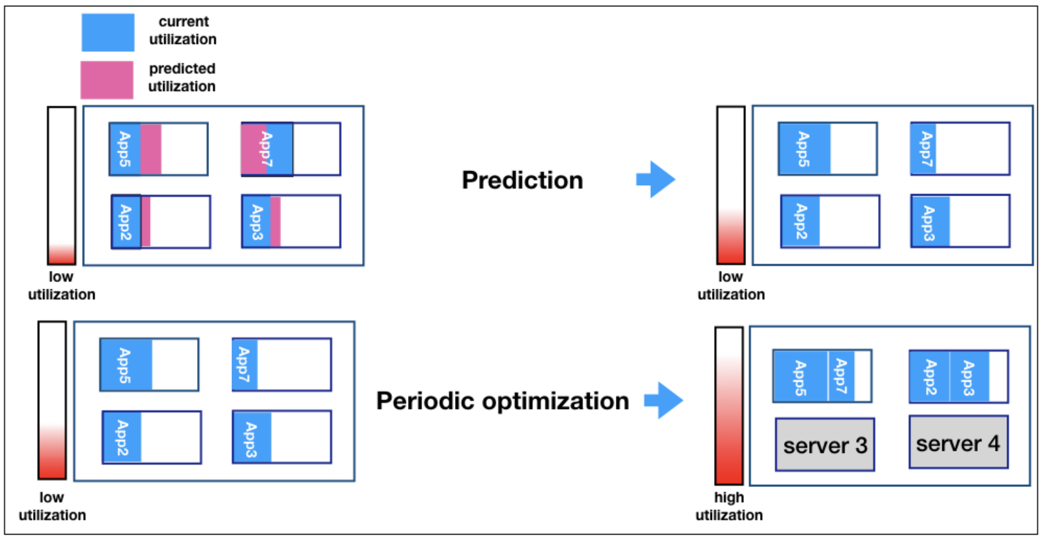
In summary, container-based virtualization has the potential to further improve the energy efficiency than VM-based virtualization. No matter which virtualization technology is used, cloud often deals with three placement decision scenarios.

2.2.2 Placement Decision Scenarios

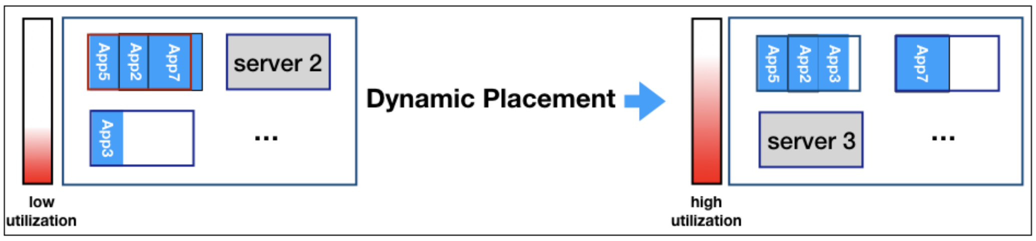
Three placement decision scenarios [83, 110]: initial placement of applications, periodic placement of applications, and dynamic placement of applications (see Figure 2.6).



(a) Initial placement of applications



(b) Periodic placement of applications



(c) Dynamic placement of applications

Figure 2.6: Three scenarios of placement decision

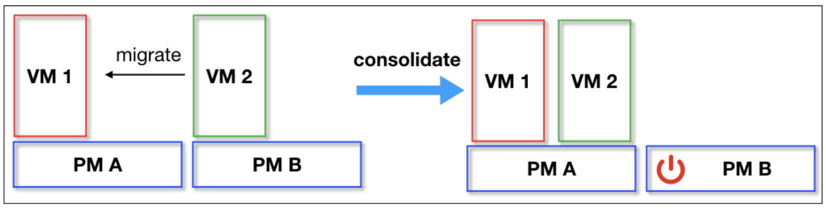


Figure 2.7: A Server Consolidation example: Initially, each PM runs an application wrapped with a VM in a low resource utilization state. After the consolidation, both VMs are running on PM A, so that PM B is turned off to save energy [8].

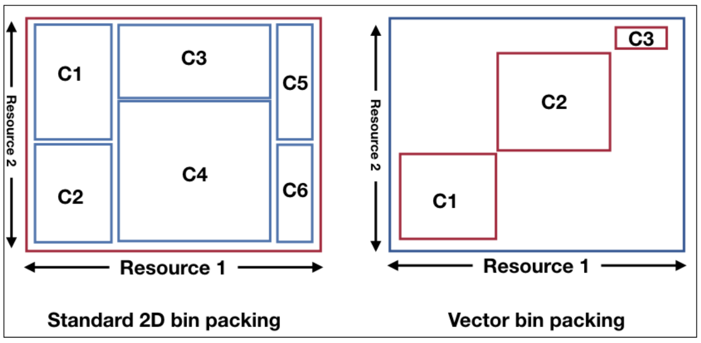


Figure 2.8: A comparison between standard bin packing and vector bin packing

Initial Placement of Applications is applied when new applications arrived. The task is to place applications into a set of PMs [83] so that PMs satisfy all applications' resource demands and minimize the energy consumption.

Periodic Placement of Applications is applied periodically to adjust the current placement of applications. The task is to re-place the current applications so that resource management minimizes the energy consumption and minimizes the cost of migration.

Dynamic Placement of Applications is applied in two scenarios [83]: Overloading and underloading. Overloading is a scenario where the total demands of applications in a PM are higher than the PM's resources. Therefore, the PM causes a degradation in one or more applications' performance. Underloading is a scenario where the PM is running in low utilization. In both scenarios, resource management moves the applications from one PM to another PMs in an on-line fashion [18].

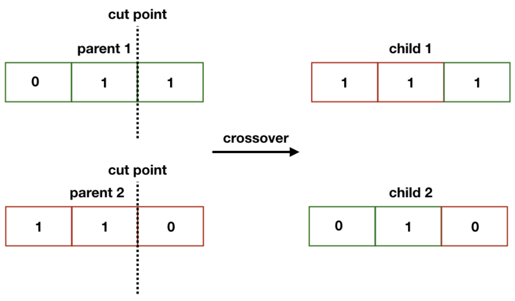
Next section will discuss general server consolidation strategies.

2.2.3 Server Consolidation Strategy

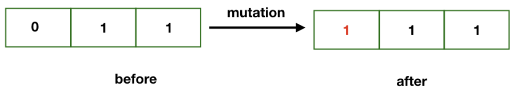
Server consolidation is a resource management strategy that aims to improve the utilization of PMs and reduce the energy consumption. We use VM-based virtualization as an example: a general step of server consolidation is shown in Figure 2.7, a number of VMs is migrated to fewer number of PMs. Resource management applies Server consolidation to solve the low utilization of PMs called PM sprawl [62].

Types of Server Consolidation

Server consolidation can be done in two ways: Static and dynamic [119,130] based on different scenarios. Initial placement and periodic placement involve large number of variables,



(a) Crossover



(b) Mutation

therefore, they are very time-consuming job and often conducted in an off-line fashion. Dynamic placement requires a fast decision-making to place one application to PMs. Thus, resource management uses a dynamic server consolidation strategy to handle the scenario.

2.3 Evolutionary Computation

Evolutionary Computation (EC) algorithms are artificial intelligent algorithms that are inspired by biological mechanisms of evolution, social interactions and swarm intelligence. They are famous for strong search ability because they have several distinguished characteristics such as the use of a population-based search and the ability of avoiding local optima. EC algorithms have been applied successfully to solve a variety of real-world problems [132], especially, combinatorial optimization problems.

Researchers have proposed a variety of EC algorithms [135] to solve optimization problems. The majority of current EC algorithms descends from three major approaches: *genetic algorithms (GAs)*, *particle swarm optimization (PSO)* and *genetic programming (GP)*.

This section will first introduces these three important EC algorithms that we would like to use in our future work. Then, we will review a number of EC algorithms, which are used in combinatorial optimization problems. From these problems, we will conclude some of the advantages of using EC algorithms in solving resource allocation problem in clouds.

2.3.1 Genetic Algorithms (GAs)

GA is a population-based search algorithm [121]. In GA, the solution of a problem is coded as a vector of numbers called chromosome or individual. A GA searches for the best solution by iteratively changing the values in the chromosome.

We describe a general GA procedure as follows. In the beginning, a population of chromosomes which represents the solutions of the problem are often represented as a fixed length string. Each entry or bit can be continuous, discrete or even alphabets depending on the problem. The evolution often starts from randomly generated population followed by an iterative process called generation. In each generation, the fitness value of chromosomes is evaluated according to a defined fitness function. Then, genetic operators such as mutation and crossover are applied on the solutions so that they are modified. As a search mechanism, these operators move solutions to explore the search space. New generations of solutions are then evaluated by the fitness function. This evolutionary procedure ends when a predefined generation number or a satisfactory level of fitness has been reached.

GA-based algorithms are suitable for off-line combinatorial optimization problems. Different from traditional algorithm such as Integer linear programming (ILP) technique, a GA-based algorithm bypasses the complicated procedure of tunning parameters by searching through the solution space of a problem. Although a GA-based algorithm does not guarantee a global optima solution, a GA-based algorithm normally can provide a near-optima solution within feasible amount of time. Although a GA-based algorithm spends less time than an ILP, it still takes too long time than an on-line problem requires. Therefore, we will consider use a GA-based algorithm in the first and second objectives: initial placement of applications and periodic placement of applications.

GA-based algorithms have been applied to a variety of combinatorial optimization problems such as the assembly line balancing problem [3], scheduling in Grid computing [143] and resource allocation problem in clouds [126]. Specifically, we will discuss the literature that applied GA-based algorithms in the server consolidation strategies in Section 2.4.

Next section will introduce another popular algorithm: particle swarm optimization (PSO).

2.3.2 Particle Swarm Optimization (PSO)

PSO is a meta-heuristic algorithm inspired by the social behavior of birds [60]. In PSO, each individual – a particle – searches the solution space. The underlying phenomenon of PSO is to use the collective knowledge to find the optimal solution.

We describe the procedure of PSO as follows. At the initial state, each particle has a random initial position in the search space which is represented by a vector $\vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D is the dimension of the search space. A particle has a velocity as $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The velocity is limited by a threshold $v_{\max}$ so that for any i and d, $v_{id} \in [-v_{\max}, v_{\max}]$. During the search process, each particle maintains a record of its best position so far, called the *personal best* (*pbest*). The best position among all the personal best positions of its neighbors is the *global best* (*gbest*). The position and velocity of each particle are updated according to the following equations:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1}, \tag{2.1}$$

$$v_{id}^{t+1} = w \cdot v_{id}^{t} + c_1 \cdot r_{1i} \cdot (p_{id} - x_{id}^{t}) + c_2 \cdot r_{2i} \cdot (p_{pg} - x_{id}^{i}). \tag{2.2}$$

Here, t is the index of iteration. d is the index of dimension. The inertia weight w is used to balance the local search and global search abilities. The parameters c_1 and c_2 are the acceleration constants. r_{1i} and r_{2i} are random constants following the uniform distribution in the interval $[0, 1]$. p_{id} and p_{gd} denote the values of *pbest* and *gbest* in the d^{th} dimension of the i^{th} particle.

PSO-based algorithms are also suitable for combinatorial optimization problems. Although PSO was originally developed for optimizing problems with continuous representations (e.g. real-value), the recent development of PSO support the discrete problem as well. Specifically, the binary PSO [61] is designed for binary problems; the discrete PSO [74] is designed for scheduling problems; and a Combinatorial PSO [55] is developed for problems with integer representation. In the literature, PSO-based algorithms have also been applied for resource allocation problem [133]. We will discuss these approaches in Section 2.4.

Compared with GA-based algorithms, PSO-based algorithms have a faster convergence speed. However, PSO-based algorithms are more easier to be stuck at local optima for large dimensional problems than GA-based algorithms. Therefore, sometimes PSO-based algorithms can be used in dynamic problems if the number of variables is small. More often, PSO-based algorithms are applied in off-line problems like GA. Therefore, we will also consider using PSO in the first and second objectives.

Next section will introduce the genetic programming-based hyper-heuristic (GP-HH), which is suitable for dynamic problems.

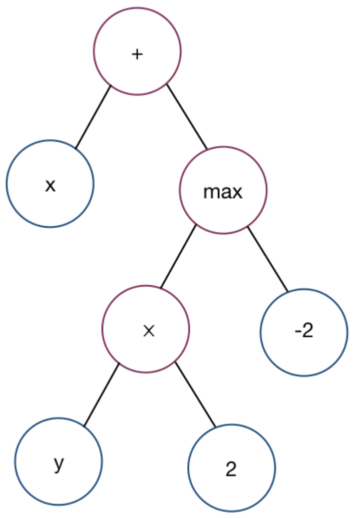


Figure 2.10: GP program that represents x + max(y × 2, -2)

2.3.3 Genetic Programming-based Hyper-heuristic (GP-HH)

Genetic programming [66] 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 shown in Figure 2.10 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 a terminal set and a 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-based hyper-heuristics (GP-HH) has been applied in many applications such as Job shop scheduling to evolve dispatching rules [86]. The term hyper-heuristics [29] means "heuristics to choose heuristics". *Dispatching rule* is essentially a heuristics [89] 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 [22,95,103]. These research studies 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 [77]. 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 that can be used to automatically generate heuristics under multiple constraints.

2.3.4 EC Algorithms in Combinatorial Optimization

This section will review a number of EC algorithms in two typical combinatorial optimization fields: cloud computing resource allocation and job shop scheduling.

Cloud Computing Resource Allocation Traditional cloud computing resource allocation includes six categories [49]: cloud brokering, VM placement, service placement and workflow scheduling are typically static problems; server load balancing and cloud capacity planning are dynamic problems. EC algorithms have been applied in problems in each of these categories. This section will briefly review approaches used in these problems except VM placement. We will explicitly discuss VM placement in Section 2.4.

In cloud brokering problem, a cloud broker is an intermediary between cloud users and cloud providers. Cloud brokers estimate the resource requirements and choose resources for cloud users. The objective of cloud brokering is to minimize the cost for cloud users as well as guarantee the quality of service (QoS) of cloud users' applications.

Frey et al [44] propose a GA approach (CDOXploer) for finding near-optimal cloud deployment architectures and runtime reconfiguration rules for softwares. Their approach takes into account VM types, the number of VMs, and the scaling policies of VMs. CDOXplorer minimizes response times, costs, and SLA violations in order to satisfy the requirement of cloud users. In comparison with Frey's centralized approach, Iturriaga et al [54] propose an Evolutionary Algorithm (EA) with distributed sub-populations. In each subpopulation, they applied a Simulated Annealing (SA) method to find the local optimal solution. The proposed algorithm is shown outperform than the reference list scheduling algorithm in both computation time and performance (maximizing the profit of a broker).

In service placement problem, cloud users would like to optimize the costs and performance (e.g QoS) of their services. Compared with cloud brokering problem, service placement less focuses on resource allocation but more focuses on the latencies between locations. Cloud users need to decide the locations of services and the number of services [49].

Tan et al [111] propose an aggregation approach with binary PSO to solve the service placement problem with two objectives: minimizing cost and minimizing the response time. They find that the single-objective algorithm can only provide one solution for each run, hence, single-objective algorithm cannot provide alternatives when cloud users do not have preferences. Therefore, they develop an NSGA-II based approach [113] that uses Pareto front approach to find a set of non-dominated solutions. They conclude that multi-objective evolutionary algorithms are suitable for the service placement problem.

The workflow scheduling problem allocates a set of dependent services, organized as a directed acyclic graph (DAG). The objectives of workflow scheduling are minimizing makespan and cost. The major difference between workflow scheduling and service placement is the extra constraint on the order of services. Another difference is that workflow scheduling rarely considers the latency issue. Compared with cloud brokering problem, workflow scheduling problems have additional dependencies between applications.

Tsai et al [115] develop an Improved Differential EA (IDEA) to solve the multi-objective scheduling problem. They represent the solution as a permutation of subtasks. They also design a crossover operator that only changes the resource allocation of subtasks and a mutation operator that changes the task order. They compare IDEA with NSGA-II, SPEA2, and DEA in two test scenarios. The results show their approach outperforms than other EC algorithms.

Capacity planning problems estimate the future load of VMs and PMs before allocating resources. The major objectives is to optimize the QoS while minimizing the cost of users. Different from above problems, the capacity planning problem is often treated as a dynamic problem.

Kousiouris et al [65] propose an artificial neural (ANN) network-based framework to predict the load of a GNU Octave system. They use a GA to create the structure of ANN and the ANN is encoded using a bit-string representation. The algorithm can be trained off-line and test on-lien. Therefore, it is well-suit for a dynamic problem.

Similar as the capacity planning problem, server Farm Load Balancing problem is also a dynamic problem. The task is to dispatch incoming requests to a set of machines. The major objective is to satisfy the QoS (e.g. makespan) of cloud users.

Laredo et al [69] propose an on-line and decentralized scheduler for distributing Bag-of-task kind of workloads to heterogeneous infrastructures. Their approach mimics the behavior of sandpile. They manage cloud resources (PMs) by segmenting resources and assign each segmentation with a monitor called agent. These agents interact with their neighbors (predefined) and share their information about resources. Tasks arrive to PMs and accumulate as grains of sand. When an agent detects an unbalance between a segment of resources and its neighbors, it triggers an avalanche. They test the system with very large scale problems: 2048 PMs and 2048 tasks and achieve at least five times better in terms of makespan and throughput than a round robin algorithm.

Server Farm load balancing is very similar to another classic problem: job shop scheduling (JSS). They both dispatch a set of jobs to a set of machines except load balancing problem focus more on computing resources such as CPUs and memories. In the next section, we will introduce JSS and the EC methods applying on JSS.

Job Shop Scheduling Job shop scheduling (JSS) problems [96] dispatch a set of jobs to machines. A job goes through a predetermined sequence of operations in order to finish its tasks. Each machine can only process a specific operation. Therefore, we need to make intelligent decisions to schedule these jobs in order to finish processing jobs before their due dates.

JSS problems can be divided into two categories: static and dynamic. In static JSS problems, all properties of jobs and machines are known in advance. In dynamic JSS problems, properties of jobs are unknown.

The state-of-the-art approaches for static JSS problems use meta-heuristics. Examples of meta-heuristics approaches to JSS include Tabu search [2], GA [138] and etc. These meta-heuristics can handle large JSS problem instances effectively.

Dynamic JSS problems often apply small heuristics such as dispatching rules [19] because dispatching rules have short reaction times and can quickly deal with unforeseen changes in an on-line event. For example, a SPT (shortest processing time) selects the job with the shortest processing time waiting at the available machine. Apart from simple dispatching rules, composite dispatching rules (CDRs) [56] combine several simple dispatching rules to achieve higher performance.

However, it is difficult to design dispatching rules for dynamic JSS problems. This is because, first, no single dispatching rule is more effective than others for all JSS problems. Second, a real-world dynamic JSS problem is changing over time, e.g. machines are added and removed. Therefore, designing dispatching rules often requires domain knowledge.

To design dispatching rules more effectively, researchers use a hyper-heuristic technique to automatically generate dispatching rules. Specifically, a great number of hyper-heuristic approaches to JSS problem uses Genetic Programming (GP) technique called GP hyper-heuristic (GP-HH). GP-HH represents dispatching rules with a tree-based representation and these dispatching rules can be interpreted as priority-based dispatching rules. GP-HH approaches generally outperform manually designed dispatching rules for both static and dynamic JSS problems [19].

In comparison between the dynamic JSS problem and the dynamic placement of applications, they share many similarities. First of all, they are both on-line problems that require a fast reaction. Second, they both dispatch jobs to machines. The differences are the dynamic JSS problem normally does not care the amount of resources that a job consumes. The dynamic placement not only considers the resource requirement of jobs but also the remaining of resources in Physical machines (PMs). Another difference is that the jobs in dynamic JSS go through a route of machines while the jobs in dynamic placement stay at one machine.

The similarities inspire us to apply GP-HH approach on the dynamic placement of applications problem.

In conclusion, EC-based algorithms have been widely used in resource allocation problems in cloud for both static problems and dynamic problems. Specifically, meta-heuristics such as GA and PSO are more suitable for static problems and hyper-heuristics such as GP are suitable for dynamic problems. We conclude several advantages of EC approaches. First, EC-based algorithms rely on searching in the solution space, which bypass the complexity of finding the exact mathematical approach. Second, EC-based algorithms are population-based algorithms, which can find near-optimal solutions within a feasible amount of time. Next section will provide the literature of traditional approaches and EC-based approaches on server consolidation strategies in terms of two virtualization: container and VM.

2.4 Related Work

Related work discusses the server consolidation strategy and the studies of consolidation strategies on three placement decision scenarios: initial placement, periodic placement, and dynamic placement.

2.4.1 Initial Placement of Applications

This section first discusses server consolidation strategies on container-based virtualization and VM-based virtualization.

Container-based Initial Placement of Applications

This section will discuss existing approaches for container-based initial placement. Then, we will describe a potentially way of modeling energy consumption in container-based virtualization: a bilevel model. Last, we will illustrate a number of algorithms to solve bilevel optimization problems.

Existing Approaches We discuss two research works on container-based virtualization: Piraghaj et al [92] and Mann [78] on their similarities and differences. Two papers both consider two resources: CPU and memory. The constraints on these resources are the containers cannot exceed the resources in their located VMs. Both research do not include the balance of CPUs and memories. In contrast, in most VM-based approaches (discussed in next section) consider the balance.

The first difference between two papers is that Mann considers the overheads of VM. Mann models the overhead as a constant value of CPU utilization but he mentioned more sophisticated models can be more realistic. On the other hand, Piraghaj et al [92] adopt a widely used VM-based linear energy model [129] and does not consider the overheads of VM.

The second difference is their resource management architectures. The architecture in Piraghaj's research allows adjusting the size of VMs when allocating new applications, while in Mann's work, containers runs on top of a traditional IaaS where containers must choose to allocate to a type of VM.

The third difference is their distinct ways to achieve initial placement based on their different architectures of resource management. Piraghaj considers the initial placement a two-steps procedure: containers to VMs and VMs to PM. Because their resource management allows cloud providers to customize the size of VMs, in the first step, they must first determine the size of VM. Piraghaj performs clustering technique on historical workload data from Google Cluster Data. In this way, Piraghaj suggests that the applications with similar workload pattern can be categorized into the same group. In the placement step, Piraghaj designs a simple heuristic: in both VM and PM levels, they apply First Fit algorithm. Then, the containers are allocated to certain size of VMs. Piraghaj claims that their main contribution is not the placement strategy but an architecture for container-based resource management. On the other hand, Mann realizes this two-levels of placement are interact with each other, therefore, container-VM and VM-PM must be considered collaboratively. Specifically, the initial placement of applications becomes three parts:

- VM size selection for containers
- Container placement
- VM placement

In order to prove interaction of two-levels of placement, Mann fixed a VM placement algorithm and tested a series of VM selection algorithms such as simple selection [45], Multiple selection, Maxsize, Consolidation-friendly. Mann discovers that the final energy consumption varies with the selection algorithms. Mann claims that the performance is better when VM selection has more knowledge of the PMs' capacity. However, Mann's study only focuses on the partial placement with fixed VM placement algorithm. The answer of "How these two-levels of placement interact ?" is still undiscovered.

We have two reasons to propose a distinct approach from Piraghaj [92] to solve the container-based placement problem. First, Piraghaj's architecture can create arbitrary size of VM when requests arrive. In contrast, our assumption is that the container-based architecture is based on traditional IaaS, where fixed-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, Piraghaj's approach cannot guarantee a near optimal energy consumption. This inspires us to simultaneously allocate containers and VMs.

In order to solve the problem, we believe a promising way is to model the energy consumption in container-based virtualization as a bilevel optimization [26] (described in the next section). Bilevel optimization represents the interaction so that a bilevel optimization algorithm can solve the energy consumption problem. Next section will introduce the basic structure of a bilevel model or a bilevel optimization.

Bilevel Optimization A bilevel optimization [26] is a kind of optimization where one problem is embedded within another. The general formulation of a bilevel optimization problem can be defined as:

$$
\begin{aligned}
min_{x \in X, y} \quad & F(x,y) && \text{(2.3a)}\\
s.t \quad & G(x,y) \leq 0, && \text{(2.3b)}\\
min_{y} \quad & f(x,y) && \text{(2.3c)}\\
s.t \quad & g(x,y) && \text{(2.3d)}
\end{aligned}
$$

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 $F\{x,y\}$ 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. This structure may introduce difficulties such as non-convexity and disconnectedness even for simple cases such as bilevel linear programming problems is strongly NP-hard [104].

In practice, there are a number of problems that are bilevel in nature. For example, transportation related: work design, optimal pricing [20, 28], management: network facility location [53], and engineering related: optimal design [63].

Evolutionary Computation Approaches for Bilevel Optimization Traditional approximation algorithms such as Karush-kuhn-Tucker approach [14, 51], branch-and-bound [6] are often applied to solve bilevel problems. 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 [80] 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 [87] 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 [124] 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 [123] that shows better performance than the previous version.

Particle Swarm Optimization [73] was also used in solving bilevel problems. A recent work is from Sinha et al [104], 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 [124] and WLD [123]. 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.

Table 2.1: A Comparison of different models and approaches

Research	Resources	Algorithm	Power model	Wastage model	Objective
Xu et al [134]	CPU and RAM	GGA and Fuzzy multi-objective	Linear	balance resources	three
Gao et al [46]	CPU and RAM	Ant Colony Optimization	Linear	balance resources	Two
Ferdaus et al [41]	CPU, RAM, and IO	Ant Colony Optimization	Linear	Sum of resources	Single
Wang and Xia [125]	CPU and RAM	MIP	Cubical	No	Single
Wilcox et al [126]	CPU and RAM	GGA	Linear	Sum of resources	Single
Xiong and Xu [133]	CPU,RAM,Bandwidth,Disk	PSO	Non-linear	Sum of resources	Single

VM-based Initial Placement of Applications

This section first describes a commonly used energy model for VM-based virtualization: Bin packing model. Then, we review a number of traditional approaches and EC-based approaches for VM-based initial placement . We mainly study the following five aspects: resources, power model, wastage model (balance between resources), objective, and algorithm.

Energy Model in VM-based Clouds: 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 2.8). Vector bin packing is also referred as multi-capacity [72] or multi-dimensional bin packing problem [133]. 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 [88]. 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 [58]. 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.

Traditional Approaches Wang and Xia [125] 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 power consumption is a cubical power function of frequency and use a *dynamic voltage and frequency scaling (DVFS)* technique to adjust CPU frequencies. 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.

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) [127], Best Fit, Best Fit Decreasing [10] 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 [84] further analyzes and discusses the drawbacks of these approaches. They found that, instead of standard bin packing, only vector bin packing is suitable for modeling resource allocation (see Section 2.4.1). 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.

EC-based Approaches Based on the insight of five principles, Gao et al [46] and Ferdaus et al [41] both propose an Ant Colony Optimization based metaheuristic using a vector algebra complementary resource utilization model proposed by Mishra [84]. 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 [37]. 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 [134] 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 [75].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 [126] also propose a reordering GGA approach because GGA can effectively avoid redundancy [36]. They use a indirect representation [98] 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.

Xiong and Xu [133] 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 2.4) 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.

$$\delta = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{d} (u_j^i - ubest_i)^2} \tag{2.4}$$

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 [85]). 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.

2.4.2 Periodic Placement of Applications

Periodic placement (see Section 2.2.2) is an process that optimizes the current allocation of resources in a periodic fashion [83]. This is because the cloud data center 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 initial placement of applications (see Section 2.4.1), the similarity is that they are both static approaches which consider a batch of applications and PMs. The difference is that periodic placement needs to take the cost of applications migration into account, therefore, it is often considered as a multi-objective optimization problem.

Based on our knowledge, periodic placement has not been studied in the context of container-based clouds. Therefore, this section only discusses VM-based periodic placement.

VM-based Periodic Placement of Applications

This section will first discuss migration models. Secondly, we will discuss the approaches in periodic placement in the VM-context, specifically, in terms of the prediction of workload, these gaps existed in both VM and container context.

Migration Model Murtazaev and Oh [85], Beloglazov et al [9] and Ferreto et al [42] 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. Using the number of migration simplified the optimization process because the optimization only considers one variable. This simplification is suitable for an environment where the sizes of VMs are invariant so that we can ignore the size of VM. However, in container-based virtualization, the size of container can vary in a wide range. Then using the number of migration will be unsuitable.

Another research direction of bandwidth optimization technique considers the network bandwidth and the size of VM memory [34, 47]. However, bandwidth optimization mainly focus on minimizing the transfer of memory pages called deduplication. Therefore, bandwidth optimization technique does not consider the interaction between migration and consolidation while we believe the consolidation should also consider the size of memory and network bandwidth.

Traditional Approaches Murtazaev's approach minimizes the number of migration by developing an algorithm that always chooses a VM from the least loaded PM and attempts to migrate these VMs to the most loaded PMs. Based on this idea, Murtazaev develops 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 [39, 120] 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. Bobroff [15] analyzed a large number of traces from real world data center. 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 [82] 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.

EC-based Approaches In a similar problem to periodic optimization of applications, Dhinesh and Venkata propose a honey bee behavior for load balancing of applications [48]. They represent workloads on overloaded PMs as bees and VMs with low load levels as bees' destinations. The load-balancing task has three objectives: decreasing makespan and response time, improving the load balance, and reducing the number of migrated tasks. The honey bee algorithm simulates bee behavior as a search mechanism. Although this research does not focus on the energy consumption, it provides some insights of the migration model and a new EC-based approach.

In summary, current migration models of placement of applications normally do not consider the resources. This model cannot apply in container-based virtualization. Another disadvantage is that most research do not consider the fluctuation of workloads. In addition, no research has focus on container-based periodic placement.

2.4.3 Dynamic Placement of Applications

This section first discusses server consolidation strategies on VM-based virtualization and container-based virtualization.

VM-based Dynamic Placement

Forsman et al [43] propose two distributed migration strategies to balance the load in a system. *The push* strategy is applied on overloaded PMs; it attempts to migrate *One* VM at a time to less loaded PMs. *The pull* strategy is applied on underutilized PMs requesting workloads from heavier loaded PMs. Each of the strategy is executed on each PM as an intelligent agent. These intelligent agents (e.g PMs) share their status with each other through a communication protocol. Forsman's approach has several interesting features. 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 [52] is useful in smoothing out variations in the average load. Third, they applied an entropy to model the load distribution. The entropy method is also applied in some previous approaches [67,97]. In addition, Forsman's system is agent-based, which means large amount of communication may occur between nodes. The heavy communication would certainly cost extra network resources. Therefore, we expect to design a centralized system, where all nodes are controlled by a controller.

Xiao et al [130] propose an algorithm based on evolutionary game theory. Their approach has 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. Second, they propose an algorithm based on Multiplayer random evolutionary game theory to solve the dynamic placement problem. In their approach, VMs are mapped into players that take part in the evolutionary game. In each iteration, all players choose their best feasible action, i.e, players migrate to the PMs, which can minimize the energy consumption. Some players randomly choose PMs in order to avoid being stuck at a local optimal. Xiao compared their approach with give simple bin-packing heuristics: 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.

Container-based Clouds

Existing Approaches Piraghaj et al [93] propose a framework for container-based resource management including three steps, analyzing resources to trigger migration, deciding which containers to migrate, and placing the container to a VM. In the third step, Piraghaj applies three heuristics: First-Fit, Random, and Least Full. However, this work only reports that their approach can reduce the number of VMs but does not mention how to reduce the number of PMs by migrating VMs. Therefore, this work does not consider the interactions between two levels: VM and PM.

In summary, current approaches for dynamic placement are mostly based on manual designed heuristics, which require domain knowledge. In addition, for container-based dynamic placement, the only research does not consider the interaction between two-level of placement and solves the bilevel problems layer-by-layer.

2.5 Summary

This chapter reviewed the main concepts of cloud resource management, server consolidation strategies, and Evolutionary Computation (EC). It also reviews both VM-based and container-based consolidate strategies in terms of traditional approaches and EC-based approaches. We illustrate the limitations of existing work on three placement decision scenarios in both container-based and VM-based clouds.

- Current research lacks appropriate model to capture the relationship between containers, VMs and PMs. Hence, most research on container-based initial placement of applications 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 placement of applications 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 placement of applications 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 applications, 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 of applications 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.

EC-based approaches have been widely applied in various combinatorial optimization problems. Specifically, meta-heuristics such as GA and PSO are more suitable for static problems and hyper-heuristics such as GP are suitable for dynamic problems. In particular, for bilevel problems, EC-based approaches have shown promising good performance. Therefore, this research aims to address the above-mentioned issues with EC-based approaches. The next chapter will focus on the initial work conducted in investigating NSGA-II for bilevel initial placement of applications.

Chapter 3

Preliminary Work

This chapter presents the initial work for investigating the first sub-objective in objective one – a bilevel model including power model, variables, and constraints. Furthermore, we also consider a multi-objective bilevel model with two optimization objectives: minimizing the energy consumption and minimizing the total price of the used virtual machines (VMs). In addition, this work investigates an NSGA-II algorithm for solving the initial placement of deploying web services. The term "web service" is used in the content instead of "container". The result covers the evaluation of the proposed algorithm along with analysis, and concluding remarks and discussion of the future work (Section 3.5).

We first introduce the related model and design an NSGA-II based algorithm to solve the problem. Then we evaluate and analyze the results. In the end, we summarize the findings and the plan for future work.

3.1 Related Models

We develop a bilevel model for allocating web services to VMs and VMs to physical machines (PMs). We consider two models: the workload model for describing the resource demand of web services and the power model for describing the relationship between resource utilization of web services and energy consumption of PMs.

3.1.1 Workload model

A workload model of web services defines the relationship between the resource demand and requests of a web service. Xavier el al [136] develop 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 VM reaches its capacity. In this work, we adopt the RAT model for web service and model the resource requirement of web services as the number of request times resources per request.

3.1.2 Power Model

A power model of PM defines the relationship between the resource utilization and the energy consumption of a PM. Shekhar's research [27] 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 [16] 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 initial placement problem, for the lack of data.

Beloglazov et al. [9] propose a comprehensive energy model for energy-aware resource allocation problem (Eq 3.2). P_{max} is the maximum power consumption when a VM 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 [68, 99].

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

In this work, we adopt the power model proposed by Beloglazov.

In summary, this preliminary work considers three factors: location of container, location of VM, and types of VM. The model does not include the overheads of VM and the balance between memory and CPU at the moment. Because we would like to investigate the how the types of VM interact with the energy model. Therefore, the first objective has not been completed in this work. Next section will formulate the problem.

3.2 Problem Description

We consider the initial placement of applications 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 initial placement 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 $A = \{A_1, A_i, \dots, A_t\}$ and memory consumption $M = \{M_1, M_i, \dots, M_t\}$, for each request consumes a A_i amount of CPU time and M_i amount of memory. The coming rate is denoted as $R = \{R_1, R_i, \dots, R_t\}$. 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 $PA = \{PA_1, PA_j, \dots, PA_p\}$ and memory $PM = \{PM_1, PM_j, \dots, PM_p\}$. 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 VMs; each VM has its CPU time $VA = \{VA_1, VA_n, \dots, VA_v\}$ and memory $VM = \{VM_1, VM_n, \dots, VM_v\}$.

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

In this work, we consider homogeneous physical machine which means physical machines have the same size of CPU time and memory. The utilization of a CPU of a VM is denoted as $U = \{U_1, U_n, \dots, U_v\}$. The utilization can be calculated by Eq.3.3.

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

The cost of a type of VM is denoted as $C = \{C_1, C_n \dots, C_v\}$.

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 [90]. 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 VMs:

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

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

Hard Constraints

We define hard constraints as the mandatory constraints.

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

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

Soft Constraint

We define the soft constraint as the constraint that can be relaxed.

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

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

This section formulates the initial placement of applications and defines the constraints. Next section will introduce our design of the NSGA-II based algorithm.

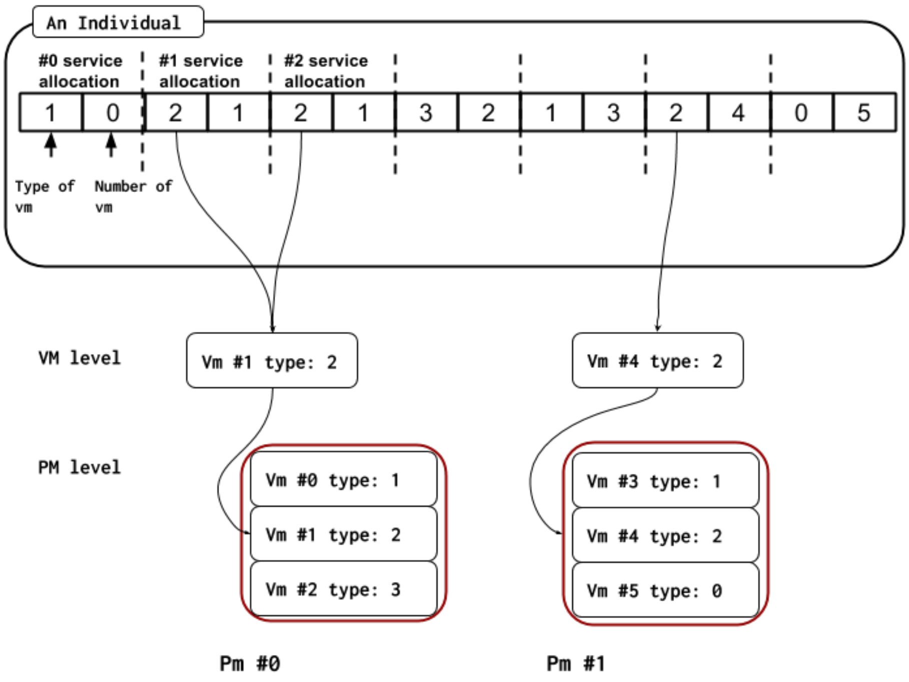


Figure 3.1: An example of chromosome representation

3.3 Methods

This section discusses the proposed algorithm including the framework of NSGA-II, the detailed problem representation, genetic operators including initialization, mutation and selection. Lastly, we give the pseudo code in Section 3.3.7.

Multi-objective Evolutionary Algorithms (MOEAs) are good at solving multi-objective problems. NSGA-II [32] is a well-known MOEA that has been widely used in many real-world optimization problems. We also adopt NSGA-II to solve the initial placement problem. We first propose a representation and then present a NSGA-II based algorithm with novel genetic operators.

3.3.1 Chromosome Representation

Initial placement is a bilevel bin-packing problem. In the first level, bins represent physical machines and items represent VMs. Whereas, in the second level, a VM acts like a bin and web services are items. Therefore, we design the representation in two hierarchies, VM 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 VM that the service is allocated in. The second number denotes the number of VM. For example, in Figure 3.1, service #1 and service #2 are both allocated in the VM #1 while service #1 and service #5 are allocated to different VMs sharing the same type. The first hierarchy shows the VM in which a service is allocated by defining type of VM and number. Note that, the type of VM 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 VMs, which are implicit. The physical machine is dynamically determined according to the VMs allocated on it. For example, in Figure 3.1, the VMs are sequentially packed into physical machines. The boundaries of PMs are calculated by adding up the resources of VMs until one of the resources researches the capacity of a PM. At the moment, no more VMs can be packed into the PM, then the boundary is determined. The reason we designed this heuristic is because a physical machine is always fully used before launching another. Therefore, VM consolidation is inherently achieved.

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

3.3.2 Initialization

Algorithm 1 Initialization

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

The initialization (see Alg 1) is designed to generate a diverse population. In the first step, for each service, it is able to find the most suitable type of VM which is just capable of running the service based on its resource requirements. In the second step, based on the suitable type of VM, 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 and therefore controls the utilization of VMs.

3.3.3 Mutation

The design principle for mutation operator is to enable individuals to explore 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 type of VM 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.

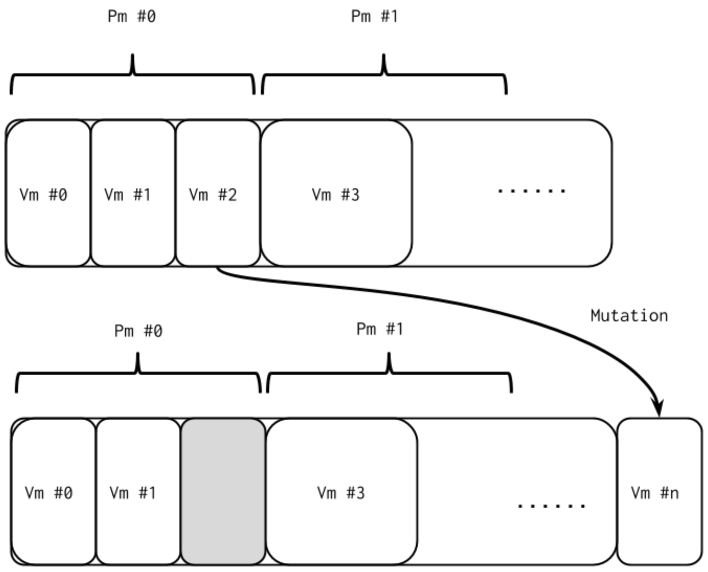


Figure 3.2: An example mutation without insertion that causes a lower resource utilization

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.

Algorithm 2 Mutation

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

3.3.4 Violation Control Method

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

3.3.5 Selection

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

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

An individual with no or less violation is always selected. The ranking method has been proved effectiveness of producing feasible solutions in the original NSGA-II paper [32].

3.3.6 Fitness Function

The cost fitness (Eq 3.5) is determined by the type of VMs at which web services 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

In previous sections (from Section 3.3.2 to Section 3.3.6), we introduce genetic operators and the fitness function. They are the components used in Alg 3.

The main difference between our approach and the original NSGA-II is that our approach has no crossover operator because it is difficult to adapt the crossover operator with our designed representation. 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 [70].

Next, we will validate the proposed algorithm through experiment.

3.4 Experiment

This section first introduces the design of test instances and experiment settings. Then, Section 3.4.2 will illustrate and analyze the results.

Algorithm 3 NSGA-II for initial placement

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

Table 3.1: Instance Settings

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

3.4.1 Dataset and Instance Design

This project is based on both real-world datasets *WS-Dream* [137] and simulated datasets [17]. 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 VMs 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 [17].

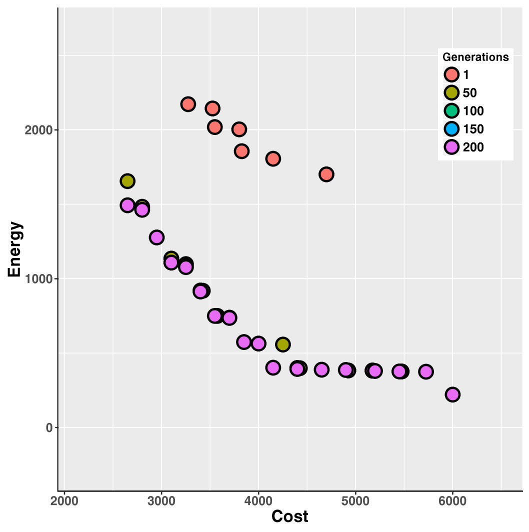
We designed six instances shown in Table 3.1, listed with increasing size and difficulty, which are used as representative samples of initial placement problem.

Selection Method with violation Control vs. without violation control

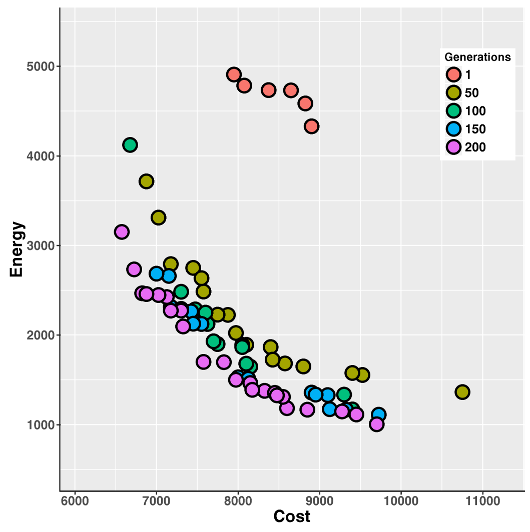
We conducted two comparison experiments. For the first experiment, we make a com-

Table 3.2: VM configurations

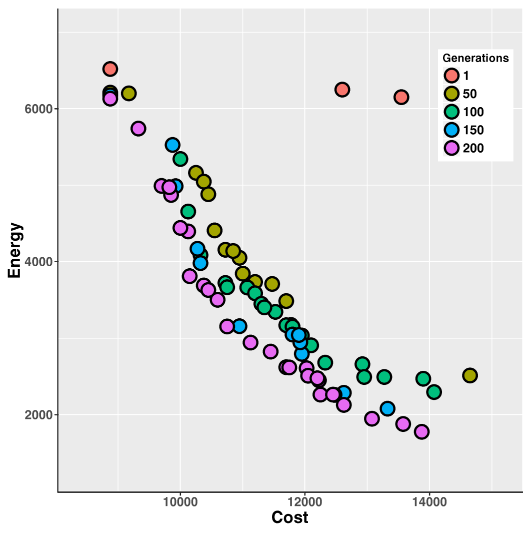
Type of VM	CPU Time	Memory	Cost
1	250	500	25
2	500	1000	50
3	1500	2500	150
4	3000	4000	300



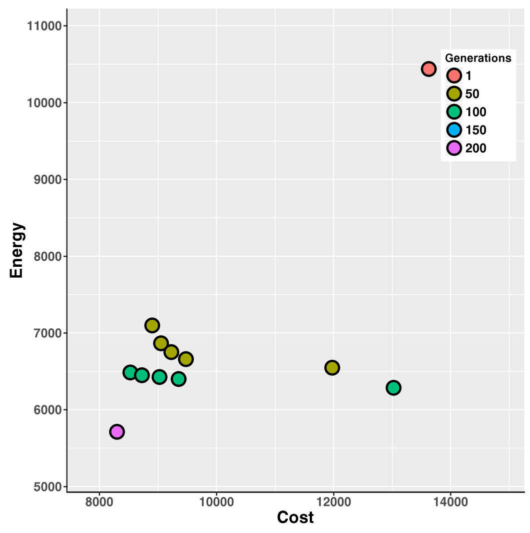
(a) Instance 1



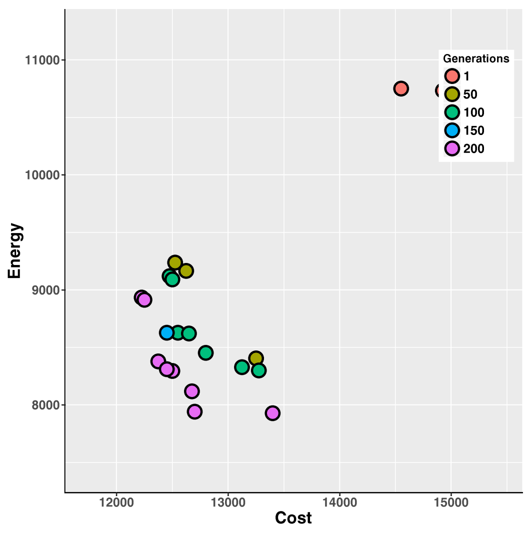
(b) Instance 2



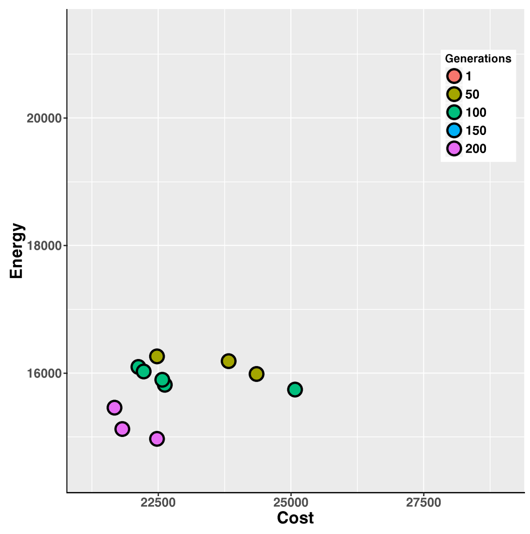
(c) Instance 3



(d) Instance 4



(e) Instance 5



(f) Instance 6

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

parison 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

Instance	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

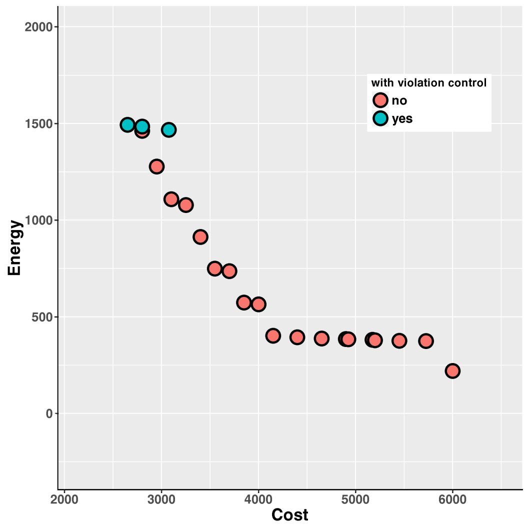
We conducted the experiment for 30 runs with different random seeds. We first obtained an average non-dominated set over 30 runs by collecting the results from a specific generation from all 30 runs. We then applied a non-dominated sorting over them.

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

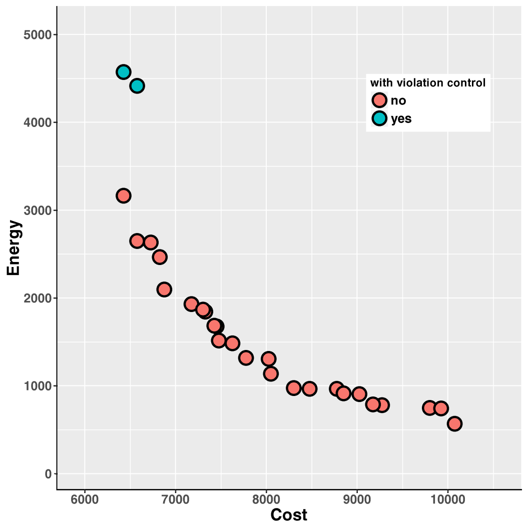
Then, the non-dominated sets of the last generation from two selection methods are compared in Figure 3.4. There are much fewer results are obtained from the violation control method throughout all cases. For the first three instances, the non-dominated set from the violation control method has similar quality as the no violation control method. From instance 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: with and without violation control. 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.

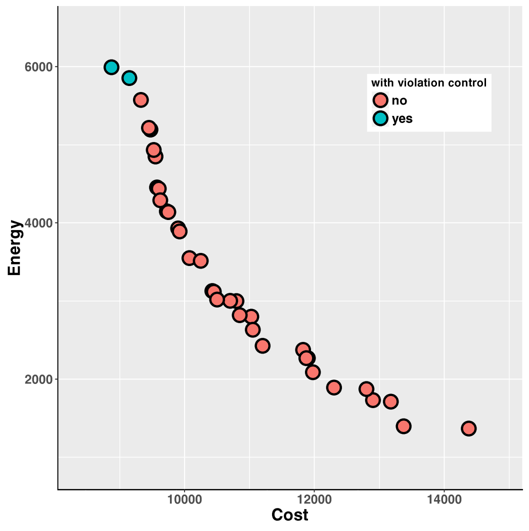
As we mentioned in previous section, the mutation rate and consolidation factor are set



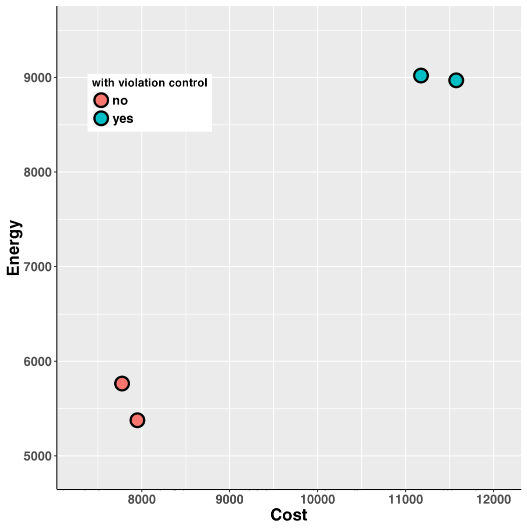
(a) Instance 1



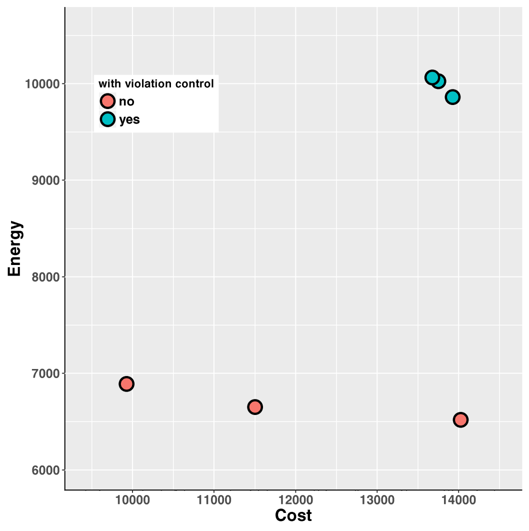
(b) Instance 2



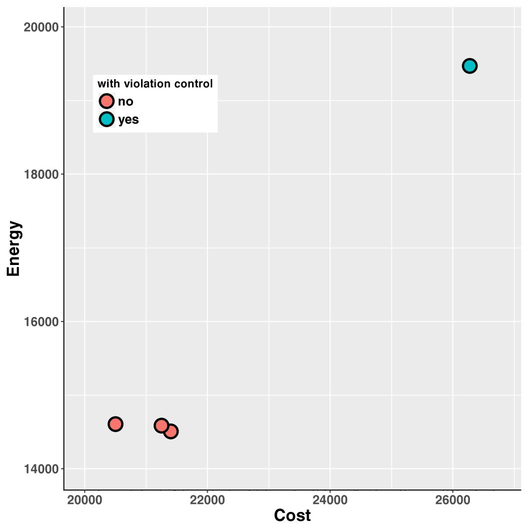
(c) Instance 3



(d) Instance 4



(e) Instance 5



(f) Instance 6

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

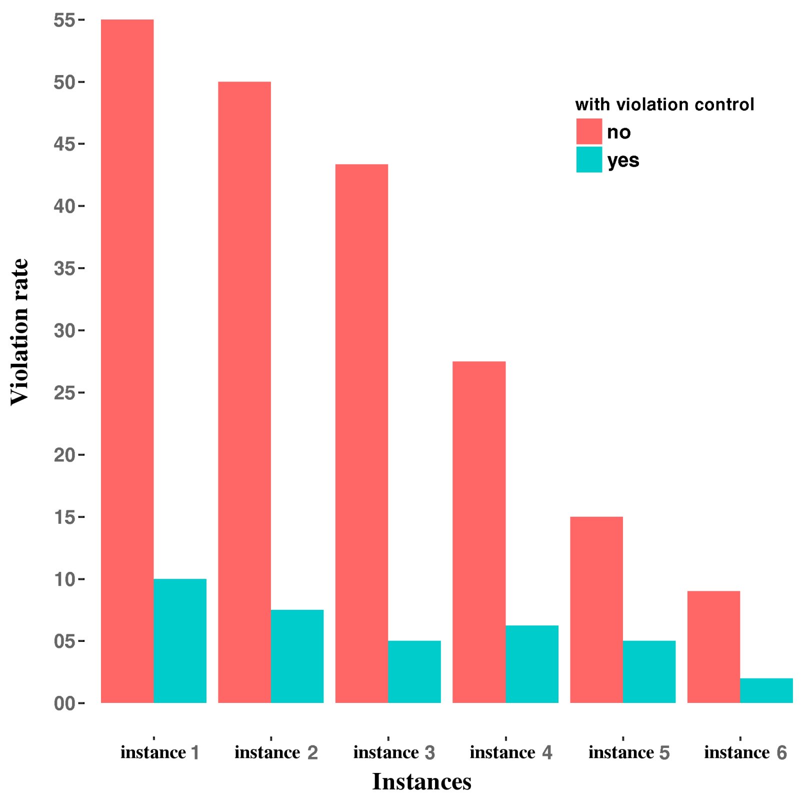


Figure 3.5: violation rate comparison between selection with violation control and without violation control

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 stuck in the local optima, a large mutation rate can help escaping from local optima. For the factor c, a larger percentage would easily lead the algorithm to infeasible regions. Therefore, the factor c 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 Findings and Future Work

This work investigated the first sub-objective of objective 1 in Chapter one: a bilevel energy model for the initial placement of containers and VMs. First, we discussed two sub models workload model and power model. Second, we established a multi-objective formulation of the bilevel problem with two objectives: minimizing the cost of used VMs and minimizing the energy consumption. In order to optimize the problem, we propose a NSGA-II based algorithm with specific designed representation. The representation is embedded with a heuristic to quickly locate feasible solutions. This work designed genetic operators such as population initialization, mutation, selection for generating valid solutions and handling the constraints. We compared the results with different variances of the algorithm.

The results show that our proposed energy model can be used in container-based server consolidation for the first sub objective in objective one. Furthermore, our NSGA-II based approach and proposed representation can quickly find feasible solutions. This preliminary work addresses the second sub objective in objective one partially. However, current work does not consider the balance between CPU and memory and the overheads of VM. Therefore, in the next step, we will continue investigating these two factors and add them into the bilevel energy model. In addition, we will propose a new EC-based approach to solve bilevel optimization problem.

Chapter 4

Proposed Contributions and Project Plan

This thesis will contribute to the field of Cloud Computing by proposing novel solutions for improving energy efficiency in container-based clouds. It will also contribute to the field of Evolutionary Computation (EC) by proposing new representations and genetic operators for solving bilevel optimization problems. The proposed contributions of this project are listed below:

1. Two new bilevel models for predicting energy consumption in container-based clouds

   First, a new bilevel energy model for initial placement of applications will be developed; **second**, we will also propose a new bilevel energy and migration model for periodic placement of applications with the consideration of three types of workload. The above two models will address the relationship between energy consumption and five factors: locations of container, types of VM, locations of VM, overheads of VM, and the balance between memory and CPU. These two bilevel models can be used in optimizing energy consumption and minimizing migration cost in container-based clouds.

2. A new EC-based bilevel single-objective optimization algorithm for solving the initial placement of applications problem based on previously proposed bilevel energy model

   The new algorithm will contribute to both Cloud computing and EC. From the perspective of cloud computing, the algorithm produces a resource allocation with better energy efficiency in container-based clouds. From the perspective of EC, the algorithm will develop new representations, search mechanisms for bilevel optimization problems. In addition, the work also will develop a hybrid approach that combines EC with clustering techniques.

3. A new EC-based bilevel multi-objective algorithm with Pareto front approach for solving the periodic placement of applications

   This algorithm will consider three types of predictable workload: static, periodic, and linear continuously changing From the perspective of cloud computing, the algorithm is expected to achieve better energy efficiency and migration cost than existing VM-

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 placement of applications with considering various types of workload	7
4	Develop a cooperative Genetic programming based hyper-heuristic approach for dynamic placement.	7
5	Writing the thesis	6

based approaches. From the perspective of EC, the algorithm will propose specific representations and genetic operators for three types of workload.

4. A new genetic programming hyper-heuristic (GP-HH) approach for solving single-objective dynamic placement of applications with three types of predictable workload and two types of unpredictable workload in **VM-based clouds**

   The proposed GP-HH focuses on solving the single-level of placement: VM to PM. From the perspective of cloud computing, the GP-HH can be used in VM-based clouds and GP-HH is expected to allocate VMs to PMs quickly and achieve a near-optimal solution in energy consumption. From the perspective of EC, the GP-HH is the first to solve resource allocation problems in container-based clouds. The algorithm will propose new primitive set, search mechanisms.

5. A new cooperative GP-HH approach for solving single-objective dynamic placement of applications with five types of workload in **container-based clouds**

   This work is based on the previously proposed GP-HH approach. From the perspective of cloud computing, the cooperative GP-HH can quickly allocate containers and VMs to achieve near optimal solutions in terms of energy consumption. From the perspective of EC, the new cooperative GP-HH method will solve bilevel dynamic problems.

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, investigating both VM-based and container-based server consolidation algorithms, and producing the proposal, has been completed. The second phase, which corresponds to the first objective of the thesis, is currently in progress and is expected to be finished on time, thus allowing the remaining phases to also be carried out as planned.

4.2 Project Timeline

The phases included in the plan above are estimated to be completed following the timeline shown in Table 4.2. The timeline 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 full completion.

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 bilevel energy model	x											
Propose a new EC based bilevel optimization approach to solve the initial placement of applications	x	x										
Improve the scalability of the proposed EC-based approach up to one thousand applications		x	x									
Propose a multi-objective bilevel model for periodic placement with consideration of static workload.				x								
Propose a multi-objective bilevel EC-based algorithm for periodic placement of applications with Pareto front approach.					x	x						
Extend the bilevel multi-objective model for adapting three types of predictable workload.						x						
Extend the previous EC-based multi-objective algorithm to adapt to three types of workload.							x					
Develop a GP-based hyper-heuristic (GP-HH) algorithm for automatically generating dispatching rules for the single-level placement.								x	x			
Conduct feature extraction on the three predictable workloads and two unpredictable workloads to construct a new primitive set.									x			
Develop a cooperative GP-HH approach for automatically generating dispatching rules for placing both containers and VMs.									x	x		
Writing the first draft of the thesis										x	x	
Editing the final draft										x	x	x

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 and discuss concepts in this field in order to provide readers with the necessary background. Multiple sections will consider issues such as initial placement of applications, periodic placement, and dynamic placement of applications. 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 initial placement of applications.*
  This chapter will establish a new bilevel energy model for the joint placement of container and VM. Based on this model, this chapter will introduce a new EC-based bilevel algorithm combined with clustering technique and heuristics to solve the initial placement of applications.

- *Chapter 4: Develop multi-objective EC-based approaches for periodic placement of applications*
  This chapter proposes a new bilevel energy and migration model based on a previously proposed energy model with three types of workload. This chapter will also propose new EC-based approaches for bilevel multi-objective periodic placement, considering three types of workload. It is then followed by algorithm performance evaluation that contains an experiment design, setting, results and analysis.

- *Chpater 5: Develop a single-objective cooperative Genetic Programming hyper-heuristic (GP-HH) approach for automatically generating dispatching rules for dynamic placement of applications*
  This chapter focuses on providing a Genetic Programming-based hybrid heuristic approach to automatically generate dispatching rules to a dynamic consolidation problem. This chapter will propose two algorithms – a GP-HH for single-level of placement: VM-PM and a cooperative GP-HH for bilevel placement: container-VM and VM-PM.

- *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 phase 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] 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. *Proceedings - IEEE INFOCOM* (2012), 1620–1628.

[2] AMARAL ARMENTANO, V., AND RIGÃO SCRICH, C. Tabu search for minimizing total tardiness in a job shop. *International Journal of Production Economics 63*, 2 (jan 2000), 131–140.

[3] ANDERSON, E., AND FERRIS, M. Genetic algorithms for combinatorial optimization: The assemble line balancing problem. *ORSA Journal on Computing 6*, 2 (may 1994), 1–23.

[4] ANGELO, J. S., KREMPSER, E., AND BARBOSA, H. J. C. Differential Evolution for Bilevel Programming. In *Proceedings of IEEE Congress on Evolutionary Computation* (2013), IEEE, pp. 470–477.

[5] BANZHAF, W., NORDIN, P., KELLER, R. E., AND FRANCONE, F. D. Genetic programming: an introduction, 1998.

[6] BARDS, F., AND SCIENCE, A. an Explicit Solution To the Multi-Level. *Computers & OR 9*, I (1982), 77–100.

[7] BARHAM, P., DRAGOVIC, B., FRASER, K., HAND, S., HARRIS, T., HO, A., NEUGEBAUER, R., PRATT, I., AND WARFIELD, A. Xen and the art of virtualization. *Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03* (2003), 164.

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

[9] BELOGLAZOV, A., ABAWAJY, J., AND BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems 28*, 5 (2011), 755–768.

[10] BELOGLAZOV, A., AND BUYYA, R. Adaptive Threshold-Based Approach for Energy-Efficient Consolidation of Virtual Machines in Cloud Data Centers. *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, December 2010 (2011), 6.

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

[12] 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.

[13] BERNSTEIN, D. Containers and cloud: From LXC to docker to kubernetes. *IEEE Cloud Computing 1*, 3 (2014), 81–84.

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

[15] BOBROFF, N., KOCHUT, A., AND BEATY, K. Dynamic placement of virtual machines for managing SLA violations. *10th IFIP/IEEE International Symposium on Integrated Network Management 2007, IM '07* (2007), 119–128.

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

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

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

[19] BRANKE, J., NGUYEN, S., PICKARDT, C., AND ZHANG, M. Automated Design of Production Scheduling Heuristics: A Review. *IEEE Transactions on Evolutionary Computation X*, X (2015), 1–1.

[20] BROTCORNE, L., LABB, 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.

[21] BURKE, E. K., HYDE, M. R., AND KENDALL, G. Evolving Bin Packing Heuristics with Genetic Programming. *Parallel Problem Solving from Nature - PPSN IX 4193*, Chapter 87 (2006), 860–869.

[22] 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.

[23] BUYYA, R., RANJAN, R., AND CALHEIROS, R. N. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *Proceedings of the 2009 International Conference on High Performance Computing and Simulation, HPCS 2009 cs.DC*, Chapter 4 (2009), 1–11.

[24] CLARK, C., FRASER, K., HAND, S., HANSEN, J., JUL, E., LIMPACH, C., PRATT, I., AND WARFIELD, A. Live migration of virtual machines. *Usenix.Org 43*, 3 (Jul 2005), 14–26.

[25] CLARK, C., FRASER, K., HAND, S., HANSEN, J. G., JUL, E., LIMPACH, C., PRATT, I., AND WARFIELD, A. Live Migration of Virtual Machines. *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation* (May 2005), 273–286.

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

[27] COMPUTING, C., AND MICROSOFT, A. K. Energy-Aware Consolidation for Cloud Computing Energy Aware Consolidation for Cloud Computing. In *Proceedings of the 2008 conference on Power aware computing and systems* (2016), no. November 2008, San Diego, California, pp. 1–5.

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

[29] COWLING, P., KENDALL, G., AND SOUBEIGA, E. A Hyper heuristic Approach to Scheduling a Sales Summit. In *:Selected Papers of theThird International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000*. Springer, Berlin, Heidelberg, Berlin, Heidelberg, Aug 2000, pp. 176–190.

[30] D. SREENIVASAN, P. GAYATHRI, R. ANITHA, AND P. DHIVYA. Optimization of resource provisioning in cloud. *IEEE transactions on service computing 5*, 2 (2012), 14–16.

[31] DAYARATHNA, M., WEN, Y., AND FAN, R. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials 18*, 1 (2016), 732–794.

[32] DEB, K., PRATAB, S., AGARWAL, S., AND MEYARIVAN, T. A Fast and Elitist Multiobjective Genetic Algorithm: NGSA-II. *IEEE Transactions on Evolutionary Computing 6*, 2 (Apr 2002), 182–197.

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

[34] DESHPANDE, U., KULKARNI, U., AND GOPALAN, K. Inter-rack Live Migration of Multiple Virtual Machines. *Vtdc '12* (2012), 19–26.

[35] DUA, R., RAJA, A. R., AND KAKADIA, D. Virtualization vs containerization to support PaaS. In *Proceedings - 2014 IEEE International Conference on Cloud Engineering, IC2E 2014* (2014), IEEE, pp. 610–614.

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

[37] FAN, X., WEBER, W.-D., AND BARROSO, L. A. Power provisioning for a warehouse-sized computer. *ACM SIGARCH Computer Architecture News 35*, June (2007), 13.

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

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

[40] FELTER, W., FERREIRA, A., RAJAMONY, R., AND RUBIO, J. An updated performance comparison of virtual machines and Linux containers. *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2015), 171–172.

[41] FERDAUS, H., AND MURSHED, M. R. N. C. R. B. Virtual Machine Consolidation in Cloud Data. *Euro-Par 2014 Parallel Processing 8632*, Chapter 26 (2014), 306–317.

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

[43] FORSMAN, M., GLAD, A., LUNDBERG, L., AND ILIE, D. Algorithms for Automated Live Migration of Virtual Machines Mattias. *Elsevier 101* (2014), 110–126.

[44] FREY, S., FITTKAU, F., AND HASSELBRING, W. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. International Conference on Software Engineering (ICSE-13). San Francisco, CA, USA,18–26 May 2013. *ICSE* (2013), 512–521.

[45] GANESAN, R., SARKAR, S., AND NARAYAN, A. Analysis of SaaS business platform workloads for sizing and collocation. *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012* (2012), 868–875.

[46] GAO, Y., GUAN, H., QI, Z., HOU, Y., AND LIU, L. Journal of Computer and System Sciences A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences 1*, 8 (2013), 1–13.

[47] GEROFI, B., VASS, Z., AND ISHIKAWA, Y. Utilizing memory content similarity for improving the performance of highly available virtual machines. *Future Generation Computer Systems 29*, 4 (2013), 1085–1095.

[48] GUPTA, H., AND SAHU, K. Honey Bee Behavior Based Load Balancing of Tasks in Cloud Computing. *Applied Soft Computing 3*, 6 (2014), 842–846.

[49] GUZEK, M., BOUVRY, P., AND TALBI, E. G. A survey of evolutionary computation for resource management of processing in cloud computing [review article]. *IEEE Computational Intelligence Magazine 10*, 2 (2015), 53–67.

[50] GYÖRGY DÓSA, AND SGALL, J. First Fit bin packing: A tight analysis. *... on Theoretical Aspects of Computer Science ... i* (2013), 1–15.

[51] HERSKOVITS, J., LEONTIEV, A., DIAS, G., AND SANTOS, G. Contact shape optimization: a bilevel programming approach. *Struc. Multidiscipl. Optim. 20*, 3 (2000), 214–221.

[52] 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.

[53] HUIJUN, S., AND ZIYOU, G. Bi-level programming model and solution algorithm for the location of logistics distribution centers based on the routing problem. *China Journal of Highway and Transport 16*, 2 (Apr 2003), 115–119.

[54] ITURRIAGA, S., NESMACHNOW, S., DORRONSORO, B., TALBI, E. G., AND BOUVRY, P. A parallel hybrid evolutionary algorithm for the optimization of broker virtual machines subletting in cloud systems. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (Oct 2013), pp. 594–599.

[55] JARBOUI, B., CHEIKH, M., SIARRY, P., AND REBAI, A. Combinatorial particle swarm optimization (CPSO) for partitional clustering problem. *Applied Mathematics and Computation 192*, 2 (2007), 337–345.

[56] JAYAMOHAN, M., AND RAJENDRAN, C. Development and analysis of cost-based dispatching rules for job shop scheduling. *European Journal of Operational Research 157*, 2 (2004), 307–321.

[57] JENNINGS, B., AND STADLER, R. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management 23*, 3 (2014), 567–619.

[58] JOHNSON, D. S. Vector Bin Packing. *Encyclopedia of Algorithms* (2014), 1–6.

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

[60] KENNEDY, J., AND EBERHART, C. *Particle swarm optimiza-tion: proceeding of IEEE International Conference onNeural Networks*, vol. 1948. Perth, 1942.

[61] KENNEDY, J., AND EBERHART, R. A discrete binary version of the particle swarm algorithm. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation* (1997), vol. 5, IEEE, pp. 4–8.

[62] KHANNA, G., BEATY, K., KAR, G., AND KOCHUT, A. Application Performance Management in Virtualized Server Environments. *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006* (2006), 373–381.

[63] KIRJNER-NETO, C., POLAK, E., AND DER KIUREGHIAN, A. An outer approximation approach to reliability-based optimal design of structures. *J. Optim. Theory Appl. 98*, I (1998), 1–16.

[64] KIVITY, A., LUBLIN, U., LIGUORI, A., KAMAY, Y., AND LAOR, D. kvm: the Linux virtual machine monitor. *Proceedings of the Linux Symposium 1* (2007), 225–230.

[65] KOUSIOURIS, G., MENYCHTAS, A., KYRIAZIS, D., KONSTANTELI, K., GOGOUVITIS, S. V., KATSAROS, G., AND VARVARIGOU, T. A. Parametric design and performance analysis of a decoupled service-oriented prediction framework based on embedded numerical software. *IEEE Transactions on Services Computing 6*, 4 (2013), 511–524.

[66] KOZA, J. Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems). *Complex adaptive systems* (1992).

[67] KUNKLE, D., AND SCHINDLER, J. A load balancing framework for clustered storage systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5374 LNCS*, 1 (2008), 57–72.

[68] 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.

[69] LAREDO, J. L. J., BOUVRY, P., GUINAND, F., DORRONSORO, B., AND FERNANDES, C. The sandpile scheduler:How self-organized criticalitymay lead to dynamic load-balancing. *Cluster Computing 17*, 2 (2014), 191–204.

[70] 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.

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

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

[73] LI, X., TIAN, P., AND MIN, X. A Hierarchical Particle Swarm Optimization for Solving Bilevel Programming Problems. In *Artificial Intelligence and Soft Computing – ICAISC 2006: 8th International Conference, Zakopane, Poland, June 25-29, 2006. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1169–1178.

[74] LIAO, C.-J., TSENG, C.-T., AND LUARN, P. A discrete version of particle swarm optimization for flowshop scheduling problems. *Computers & Operations Research 34*, 10 (2007), 3099–3111.

[75] LIEN, C. H., BAI, Y. W., AND LIN, M. B. Estimation by software for the power consumption of streaming-media servers. *IEEE Transactions on Instrumentation and Measurement 56*, 5 (2007), 1859–1870.

[76] LIU, H., XU, C.-Z., JIN, H., GONG, J., AND LIAO, X. Performance and energy modeling for live migration of virtual machines. *Proceedings of the 20th international symposium on High performance distributed computing - HPDC '11 16*, 2 (2011), 171.

[77] MANN, Z. A. Approximability of VM allocation : Much harder than bin packing. *Proceedings of the 9th Hungarian -Japanese Symposium on Discrete Mathematics and Its Applications* (2015), 21–30.

[78] MANN, Z. Á. Interplay of virtual machine selection and virtual machine placement. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9846. Springer International Publishing, Cham, Aug 2016, pp. 137–151.

[79] MANVI, S. S., AND KRISHNA SHYAM, G. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications 41*, 1 (2014), 424–440.

[80] MATHIEU, R., PITTARD, L., AND ANANDALINGAM, G. Genetic algorithm based approach to bi- level linear programming. *RAIRO-Operations Research 28*, 1 (Mar 1994), 1–21.

[81] 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.

[82] MENG, X., ISCI, C., KEPHART, J. O., ZHANG, L., BOUILLET, E., AND PENDARAKIS, D. Efficient resource provisioning in compute clouds via VM multiplexing. *Proceeding of the 7th international conference on Autonomic computing - ICAC'10* (2010), 11.

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

[84] 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.

[85] MURTAZAEV, A., AND OH, S. Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing. *IETE Technical Review 28*, 3 (Sep 2011), 212.

[86] 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.

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

[88] PANIGRAHY, R., TALWAR, K., UYEDA, L., AND WIEDER, U. Heuristics for Vector Bin Packing. *Research.Microsoft.Com* (2011).

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

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

[91] PINHEIRO, R. L., LANDA-SILVA, D., AND ATKIN, J. Analysis of Objectives Relationships in Multiobjective Problems Using Trade-Off Region Maps. *2015 Genetic and Evolutionary Computation Conference (GECCO 2015)* (2015), 735–742.

[92] PIRAGHAJ, S. F., CALHEIROS, R. N., CHAN, J., DASTJERDI, A. V., AND BUYYA, R. Virtual machine customization and task mapping model for efficient allocation of cloud data center resources. *The Computer Journal 59*, 2 (2016), 208–224.

[93] PIRAGHAJ, S. F., DASTJERDI, A. V., CALHEIROS, R. N., AND BUYYA, R. A Framework and Algorithm for Energy Efficient Container Consolidation in Cloud Data Centers. *Proceedings - 2015 IEEE International Conference on Data Science and Data Intensive Systems; 8th IEEE International Conference Cyber, Physical and Social Computing; 11th IEEE International Conference on Green Computing and Communications and 8th IEEE International Conference on Internet of Things, DSDIS/CPSCom/GreenCom/iThings 2015* (2015), 368–375.

[94] PIRAGHAJ, S. F., DASTJERDI, A. V., CALHEIROS, R. N., AND BUYYA, R. A Survey and Taxonomy of Energy Efficient Resource Management Techniques in Platform as a Service Cloud. In *IGI Global*. IGI Global, 2016, pp. 410–454.

[95] 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.

[96] POTTS, C. N., AND STRUSEVICH, V. A. Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society 60*, S1 (2009), S41–S68.

[97] QIN, X., ZHANG, W., WANG, W., WEI, J., ZHAO, X., AND HUANG, T. Towards a cost-aware data migration approach for key-value stores. *Proceedings - 2012 IEEE International Conference on Cluster Computing, CLUSTER 2012* (2012), 551–556.

[98] RADCLIFFE, N. Formal Analysis and Random Respectful Recombination. *Proceedings of the 4th International Conference on Genetic Algorithms, San Mateo, California* (1991), 222–229.

[99] RAGHAVENDRA, R., RANGANATHAN, P., TALWAR, V., WANG, Z., AND ZHU, X. No Power Struggles : Coordinated Multi-level Power Management for the Data Center. In *Solutions* (2008), vol. 36, ACM, pp. 48–59.

[100] 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.

[101] ROSEN, R. Resource management : Linux kernel Namespaces and cgroups. *Haifux*, May (2013), 1–120.

[102] SHEN, S., VAN BEEK, V., AND IOSUP, A. Statistical characterization of business-critical workloads hosted in cloud datacenters. *Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015* (2015), 465–474.

[103] 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: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference* (2013), 1549–1556.

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

[105] 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.

[106] SOLTESZ, S., PÖTZL, H., FIUCZYNSKI, M. E., BAVIER, A., AND PETERSON, L. Container-based operating system virtualization. *ACM SIGOPS Operating Systems Review 41*, 3 (2007), 275.

[107] SOMANI, G., AND CHAUDHARY, S. Application performance isolation in virtualization. *CLOUD 2009 - 2009 IEEE International Conference on Cloud Computing* (2009), 41–48.

[108] SPROTT, D., AND WILKES, L. Understanding Service-Oriented Architecture. *The Architecture Journal 1*, 1 (2004), 10–17.

[109] SUN, Y., WHITE, J., LI, B., WALKER, M., AND TURNER, H. Automated QoS-oriented cloud resource optimization using containers. *Automated Software Engineering 24*, 1 (2017), 101–137.

[110] SVÄRD, P., LI, W., WADBRO, E., TORDSSON, J., AND ELMROTH, E. Continuous datacenter consolidation. In *Proceedings - IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015* (2016), IEEE, pp. 387–396.

[111] TAN, B., HUANG, H., MA, H., AND ZHANG, M. *Binary PSO for Web Service Location-Allocation*, vol. 10142 LNAI. 2017.

[112] TAN, B., MA, H., AND MEI, Y. A NSGA-II-based Approach for Service Resource Allocation in Cloud. *CEC* (2017), 2574–2581.

[113] TAN, B., MA, H., AND ZHANG, M. *Optimization of Location Allocation of Web Services Using a Modified Non-dominated Sorting Genetic Algorithm*, vol. 9592. 2016.

[114] TOMÁS, L., AND TORDSSON, J. Improving cloud infrastructure utilization through overbooking. *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference on - CAC '13* (2013), 1.

[115] TSAI, J.-T., FANG, J.-C., AND CHOU, J.-H. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research 40*, 12 (dec 2013), 3045–3055.

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

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

[118] VARASTEH, A., AND GOUDARZI, M. Server Consolidation Techniques in Virtualized Data Centers: A Survey. *IEEE Systems Journal 11*, 2 (2017), 772–783.

[119] VERMA, A., DASGUPTA, G., NAYAK, T. K., DE, P., AND KOTHARI, R. Server Workload Analysis for Power Minimization using Consolidation. *Proceedings of the 2009 conference on USENIX Annual technical conference 1505* (2009), 28.

[120] VISWANATHAN, B., VERMA, A., AND DUTTA, S. CloudMap: Workload-aware placement in private heterogeneous clouds. *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012* (2012), 9–16.

[121] VOSE, M. D., AND HALL, A. Modeling Simple Genetic Algorithms. In *Evolutionary Computation*, vol. 3. Elsevier, 1996, pp. 453–472.

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

[123] WANG, Y. 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.

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

[125] WANG, Y., AND XIA, Y. Energy optimal VM placement in the cloud. *IEEE International Conference on Cloud Computing, CLOUD* (2017), 84–91.

[126] WILCOX, D., MCNABB, A., AND SEPPI, K. Solving virtual machine packing with a Reordering Grouping Genetic Algorithm. *2011 IEEE Congress of Evolutionary Computation, CEC 2011* (2011), 362–369.

[127] 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.

[128] 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 *21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)* (2013), IEEE, pp. 233–240.

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

[130] Xiao, Z., Jiang, J., Zhu, Y., Ming, Z., Zhong, S., and Cai, S. A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory. *Journal of Systems & Software 101* (2015), 260–272.

[131] Xie, J. An Efficient Global K-means Clustering Algorithm. *JCP 6*, 2 (2011), 271–279.

[132] Xing, B., and Gao, W.-J. *Introduction to Computational Intelligence*. John Wiley & Sons, Ltd, 2014.

[133] 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.

[134] Xu, J., and Fortes, J. A. Multi-objective virtual machine placement in virtualized data center environments. *Proceedings - 2010 IEEE/ACM International Conference on Green Computing and Communications, GreenCom 2010, 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, CPSCom 2010* (2010), 179–188.

[135] Yao, X., and Xu, Y. Recent advances in evolutionary computation. *Journal of Computer Science and Technology 21*, 1 (Jan 2006), 1–18.

[136] 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, pp. 1–7.

[137] Zhang, Y., Zheng, Z., and Lyu, M. R. Exploring latent features for memory-based QoS prediction in cloud computing. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems* (2011), pp. 1–10.

[138] Zhou, H., Cheung, W., and Leung, L. C. Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. *European Journal of Operational Research 194*, 3 (may 2009), 637–649.

[139] Zhu, X., Yu, Q., and Wang, X. A Hybrid Differential Evolution Algorithm for Solving Nonlinear Bilevel Programming with Linear Constraints. In *Cognitive Informatics, 2006. ICCI 2006. 5th IEEE Intl. Conf. on* (2006), vol. 1, IEEE, pp. 126–131.

[140] Zio, E., and Bazzo, R. A clustering procedure for reducing the number of representative solutions in the Pareto Front of multiobjective optimization problems. *European Journal of Operational Research 210*, 3 (2011), 624–634.

[141] Zio, E., and Bazzo, R. A Comparison ofMethods For Selecting Preferred Solutions inMultiobjective Decision Making. In *Computational Intelligence Systems in Industrial Engineering*, vol. 6. Atlantis Press, Paris, Nov 2012, pp. 203–230.

[142] Zomaya, A. H. K. R. P. J. K. B. Z. M. M. T. V. U. K. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing NA*, 7 (2014), 751–774.

[143] Zomaya, A. Y., and Crnomarkovic, K. Genetic Algorithms for Scheduling in Grid Computing Environments. *Handbook of Bioinspired Algorithms and Applications 20053845* (2005), 13–208.