Chapter 1

Introduction

1.1 Problem Statement

Cloud computing is a computing model offering a network of servers to their clients in a on-demand fashion. From NIST's definition [33], *"cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."* To illustrate how it works, considering a case: a Cloud provider builds a data center which contains thousands of servers connected with network. These servers are virtualized which means they are partitioned into a smaller unit of resources called *Virtual Machines (VMs)*. A web-based application provider can access and deploy their applications (e.g Endnote, Google Drive and etc.) in these VMs from anywhere in the world. Once the applications start serving, application users can use them without installing on their local computers.

Generally, Cloud computing involves three stakeholders: Cloud providers, Cloud users, and End (application) users [23] (see Figure 1.1).

Each stakeholder has their responsibility, goal, and objectives.

- *Cloud providers* build data centers, provide maintenance and resource management on the hardware infrastructure. Their goal is to increase the profit by boosting the income and reducing the expense. Their income comes from Cloud users' rental of servers or *Physical Machines (PMs)* in terms of resource quality (e.g 3.5GHz dual-core CPU), quantity (e.g 3 PMs), and time (e.g 1 year). Therefore, Cloud providers objective is to maximize utilization of computing resources. A high utilization brings two benefits, firstly, it increases income by accommodating more applications in limited resources. Secondly, it cuts the expense of energy consumption by packing applications in a minimum number of PMs so that idle PMs can be turned off.

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

- *End Users* are the final customers in this chain. They consume services directly from Cloud users and indirectly from Cloud provider. Their goal is to obtain a satisfactory

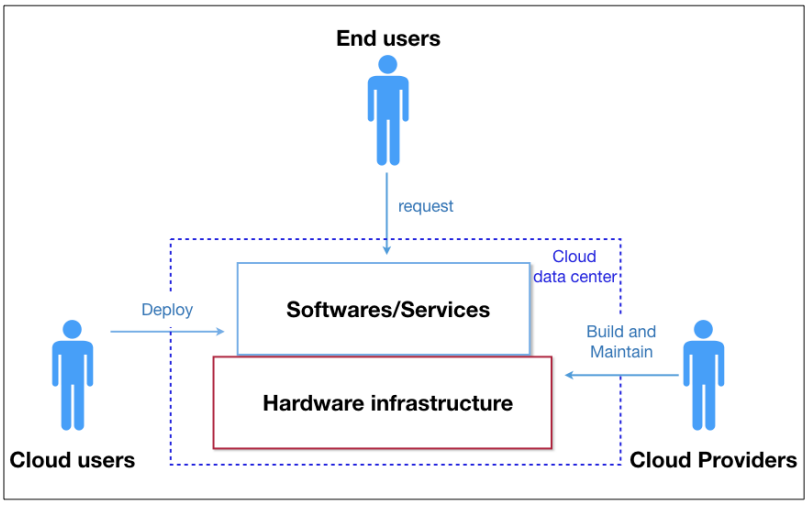


Figure 1.1: Stakeholders of Cloud computing

service. It is achieved by signing a Service Level Agreement (SLA) with Cloud users which constrains the performance of the services.

In this thesis, we focus on an core issue of helping Cloud providers to increase their profits by optimizing the resource allocation in data centers. Specifically, the profit can be improved by reducing the energy consumption. A direct way to reduce energy consumption is to always use a minimum number of Physical machines (PMs) hosting applications. This is because the more PMs are running, the more energy they are consumed.

The problem can be described as, a data center has a number of PMs where each of them can be represented as a set of resources such as CPU cores, RAM and etc. The data center received a list of requests for applications which is also represented as resources. The task is to allocate these requested resources to a minimum number of PMs. The decision variable in this task is the location of each requested resource. The assumptions and constraints are distinct in service models of Cloud computing.

Service models of Cloud computing are critical in solving energy consumption problem because their distinct ways of managing resources have sever effect on the problem. These distinct ways of resource management mainly result from the responsibilities among stakeholders. There are three traditional service models [33]: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The relationship among three service models can be pictured as a pyramid (see 1.2). Their detailed responsibilities among stakeholders are explained as below:

- IaaS, a Cloud provider hosts hardwares such as PMs and cooling infrastructure on behalf of Cloud users. A Cloud provider also provides maintenance, management of hardware resources.

  In IaaS, Cloud provider offers the fundamental resources such as CPU cores, RAMs, and network bandwidth. These resources are often encapsulated in virtualized computing units called virtual machines (VMs). Cloud providers establish a number of types of VM for simplifying the management. The 'type' means a VM contains a certain quantity of resources such as 2-cores and 1 GB RAM. *Traditional* IaaS and PaaS use VM as the fundamental unit of resources.

  A typical procedure of a Cloud user deploying their applications in an IaaS cloud includes several steps. Initially, Cloud users estimate the resources that their appli-

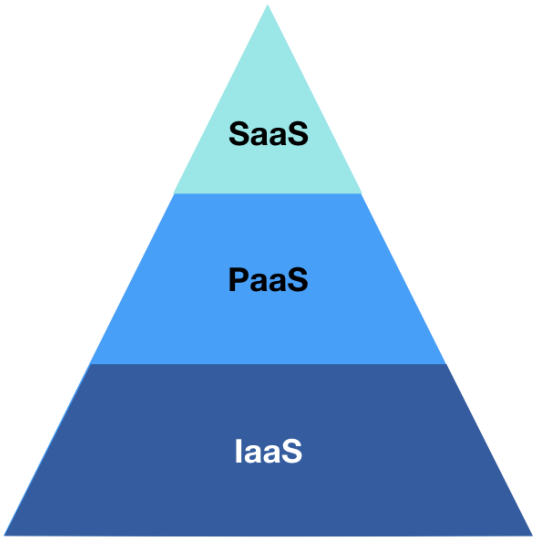


Figure 1.2: Pyramid of service models in Cloud computing. IaaS provides the fundamental resources such as CPU cores, RAM. The resources are usually wrapped with various types of virtual machine. PaaS provides a software platform sitting on top of IaaS for Cloud users to develop applications. SaaS is the application that serves End Users.

cations might consume and select a type of VM which can satisfy the requirement. After Cloud users have made the decisions, they send requests to Cloud providers for a number of VMs. Finally, Cloud providers received the request, provisioned and allocated these VMs to PMs.

From the resource management perspective, the constraint in allocation of VMs is that the aggregated VMs' resources in a PM cannot exceed the capacity of the PM. After these VMs have been allocated, their types cannot be changed. During the life cycle of an application, Cloud providers can dynamically adjust the locations of VMs, provision new VMs (same type) for the replicas of an application, as well as turning on/off PMs.

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

  From resource management perspective, as shown in Figure 1.2, PaaS is sitting above the IaaS which means the underlying resource is still based on IaaS VM types. Different from IaaS, PaaS takes the responsibility of selecting VMs and allows Cloud users to focus on software development.

  A typical procedure of a Cloud user deploying their applications in an PaaS cloud includes several steps. In the first step, Cloud users need to provide the initial estimation of the quantity of resources instead of types of VM. Then, Cloud providers determine the types of VM for applications according to the estimated resources. After this step, resource management system conducts the provisioning and allocating as the same steps in IaaS. During the life cycle of applications, similar to IaaS, Cloud providers can also adjust the location of VMs, add new VMs, and control the status of PMs. Different from IaaS, Cloud providers can change the type of VM for an application as long as the application's performance can be guaranteed.

- SaaS describes the relationship between Cloud users and End users. End users create

workloads for applications. Although this service model does not directly related to the resource management, it provides the fundamental reasons for resource management and optimization. Because of the dynamic nature of workloads, the underlying resources must also be dynamic adjusted to meet the requirement.

Our proposed optimization approaches are based on a new service model: Container as a Service (CaaS) [39] which is a variant of PaaS. The reasons for us to establish our approaches on CaaS are listed as followed:

- CaaS uses a new virtualization technology called containers which has shown several important characteristics that overcome the disadvantages of traditional IaaS and PaaS, therefore, CaaS is a promising trend.

- CaaS has a fine granularity level of resource management which has shown opportunities to improve the resource utilization, however, it often proposes new optimization challenges which have not been studied yet.

Firstly, we illustrate the disadvantages of traditional IaaS and PaaS and discuss the reasons of why current approaches cannot completely overcome these problems. From there, we explain why CaaS is a prescription of their problems.

IaaS has three characteristics which naturally lead to a low resource utilization.

- Separated responsibilities of resource selection for applications and resource allocation. As above mentioned, Cloud users have to select the type of resources. This causes a problem. The accurate estimation is almost impossible because of unpredictable workloads; Cloud users tend to reserve more resources for ensuring the QoS at the peak hours [11] than completely rely on auto-scaling, simply because auto-scaling is more expensive than reservation. However, the peak hours only account for a short period, therefore, in most of time, resources are wasted. As the types of VM are a part of the contract, Cloud providers cannot simply change the type of VMs after provisioning.

- Cloud providers offer fixed types of VM. Because of the fixed size of resources and the one-on-one mapping of applications and VMs, specific applications consume unbalanced resources which leads to vast amount of resource wastage [51]. 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.

- Redundant operating systems (OSs) cause vast resource wastage. Since each VM runs a separated operating system, a PM might run many operating systems at a time. However, normal applications do not need specific operating systems commonly used OSs - such as Linux-based: RedHat, or Windows server versions - are well enough for their needs. Therefore, running duplicated OSs in a PM is unnecessary.

For the first two drawbacks, previous researchers and developers have come up with two strategies: server consolidation and overbooking.

Server consolidation [62], as above mentioned, utilized a dynamic migration technique to resolve the low utilization problem by gathering applications into a fewer number of PMs (see Figure 1.3), so that the resource utilization of PMs are maintained at a high level. In the meanwhile, idle PMs can be turned off to save energy. Consolidation dramatically improves hardware utilization and lowers PM and cooling energy consumption.

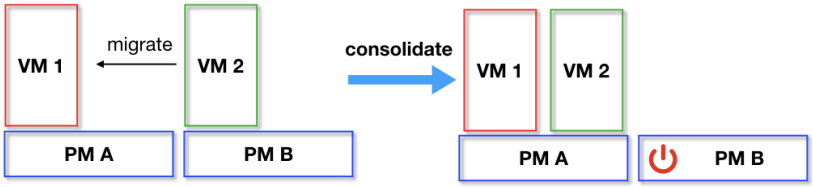


Figure 1.3: A Server Consolidation example: Initially, each PM runs an application wrapped with a VM in a low resource utilization state. After the consolidation, both VMs are running on PM A, so that PM B is turned off to save energy [3].

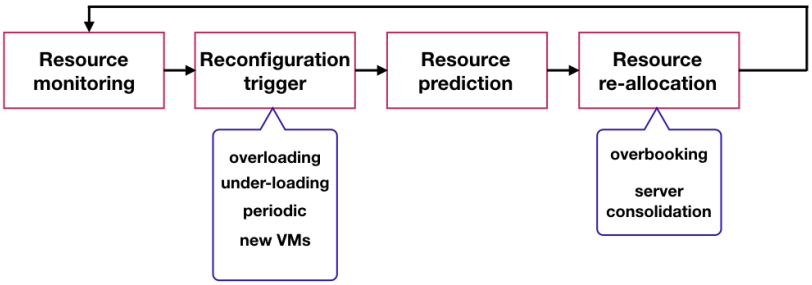


Figure 1.4: A workflow of dynamic resource management [34]

Overbooking strategy [51] is used to overcome the low utilization problem raised by the over-provisioning of VMs. It allocates more VMs in a PM even their aggregated resources have exceeded the PM's capacity. The advantage of this approach is that, indeed, it improves the resource utilization. The disadvantages are also obvious: if one or more applications are experiencing a heavy load; the PM is likely to run out of resources. At this moment, all the applications are suffer from a QoS degradation. In order to avoid the overloading, Cloud users often carefully predict the utilization of applications based on their previous workloads, when applications start degrading, a dynamic resource management approach is used to adjust the VMs' location.

The work-flow of utilizing these two strategies in resource management is shown in Figure 1.4. It seemingly solves the over-provisioning problem, however, the overbooking strategy brings new challenges. Although the overbooking strategy have been studies for years [], the prediction of workload is very difficult [] or even impossible as referred in many literatures []. The improvement of resource utilization is completely based on an accurate prediction of a large number of applications. Its effectiveness is hard to evaluate and justify. It is also very time consuming to predict the utilization of every application. Therefore, it is urgent to provide a strategy without making a prediction. One possible way is to use finer granularity strategy of resource management.

Furthermore, there is no solution for the third drawback in the context of IaaS.

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

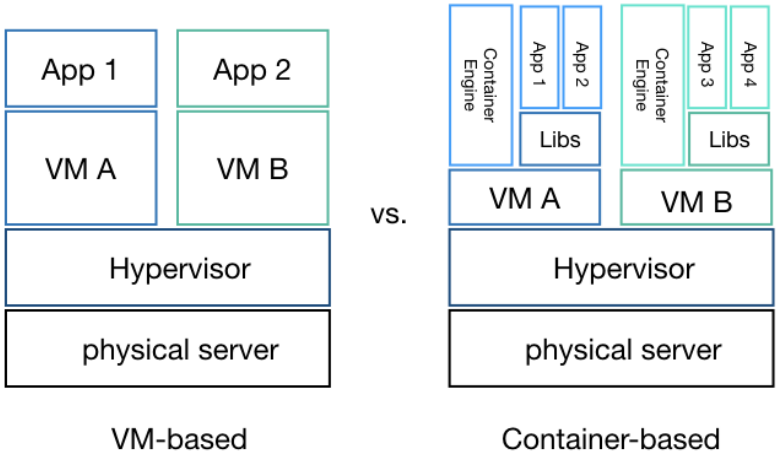


Figure 1.5: A comparison between VM-based and Container-based virtualization

supports automatic resource management as well as an editable programming environment.

The recent development of container technology [45] has driven the attention of both industrial and academia because its potential to solve these problems in both IaaS and PaaS. Container is an operating system level of virtualization which means multiple containers can be installed in a same operating system (see Figure 1.5 right-hand side). Each container provides an isolated environment for an application. In short, a VM is partitioned into smaller manageable units. Container as a Service (CaaS) is a variant of PaaS propose by a leading Lab in this field [40]. Instead of VMs, CaaS uses both container and VM as a hybrid resource management unit.

CaaS solves the problems in IaaS and PaaS because of containers' characteristics. The most important feature of container is that it utilizes the resources inside VMs. Therefore, even CaaS is also built upon IaaS, it solves the fixed types by allocating containers in it. This feature also solves the redundant operating systems problem. The second characteristic is that a container's size is changeable and can be controlled by Cloud provider in real-time. Therefore, the Cloud providers can monitor the resource utilization of applications and adjust it accordingly. This feature provides better flexibility which solves the over-provisioning problem. In addition, one feature that separates CaaS and PaaS is that, containers allow user-defined libraries instead of constrained by the platform.

Although CaaS has these promising characteristics, it also proposes a difficult research problem. In order to understand the difficulty of the problem, it is necessary to illustrate the difficulty of current VM-based server consolidation problem.

Currently, vast amount of server consolidation methods are mostly VM-based and they are mainly modeled as bin-packing problems [31]. This is because VMs and PMs are naturally modeled as items and bins and server consolidation and bin-packing have the same optimization objective: minimize the number of bins/PMs. The complexity of bin-packing problem is NP-hard which means it is extreme time-consuming to find its optimal solution when the number of decision variables are large. Deterministic methods such as Integer Linear Programming [48] and Mixed Integer Programming [54] have been used but they are well-known that they are very time-consuming for a large scale problem. More research proposed heuristic methods to approximate the optimal solution such as First Fit Decreasing (FFD) [37], Best Fit Decreasing (BFD) [5]. Manually designed heuristics are designed to tackle the special requirements such as a bin-item incomplete scenario [20] and multi-tier applications [25, 30]. Although these greedy-based heuristics can quickly approximate an

answer, as Mann's research [31] showed, server consolidation is a lot more harder than bin-packing problem because of the multi-dimensional, many constraints. Therefore, general bin-packing algorithms do not perform well with many constraints and specific designed heuristics only perform well in very narrow scope.

Although traditional VM-based server consolidation has been studied for year, these methods can not be directly applied on container-based problem because container-based consolidation has two levels of allocation: Containers are allocated to VM as the first level and VMs are allocated to PMs as the second level. These two levels of allocation interact with each other, for the first level of allocation affects the decision in the second level.

1.2 Motivation

Cloud data center has a high dynamic nature where it constrantly receives new requests for resource provisioning and releases old current resources. Therefore, a data center needs different strategies to handle different scenarios.

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

1. The initialization,
   At this stage, a set of applications or containers is allocated to a set of empty VMs and these VMs are allocated to a set of PMs. Initalization problem is also refered as placement problem [23] which is fundamental for server consolidation problem. This problem is inherently more difficult than previous VM-based consolidation problem, since container-based consolidation naturely has a two-level structure. Only a few research focus on this problem, Piraghaj [39] designs a dynamic allocation system. She proposes a two-step procedure; it first allocates containers to VMs and then allocate containers to VMs. Since, these two-level structure interact each other, separate solution certainly leads to a local optima. Therefore, in this thesis, we will solve the problem simutaneously.

   In this objective, we will establish the fundamental concepts in studying this joint allocation of containers and VMs including new problem models: price and power model, new problem constraints, and optimization objectives. The major challenges for this objective is to design representations and several EC approaches to solve this problem. More specifically, in designing the EC approach, new search mechanisms, operators will be designed and new representations will be proposed to fit the problem.

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

   The challenges are three folds, firstly, similar with initialization problem, the problem has two level of allocations and they interact with each other. Secondly, like VM-based consolidation, Container-based consolidation is considered as a multi-objective problem with minimization of migration cost as well as keeping a good energy efficiency. Thirdly, consolidation is a time-dependent process which means the previous solution

affects the current decision. Previous research only consider each consolidation as an independent process. As a consequence, although in one consolidation, the migration is minimized. It may lead to more migration in the future consolidation. We will consider the robustness of consolidation and propose a novel time-aware server consolidation which takes the previous immediate consolidation and the future consolidation into consideration.

3. Dynamic consolidation,
   It takes one container and allocates it to VMs. Since the size of container can be dynamically adjusted, when the an application is under-provision or over-provision, the orginal container is halted, resized and re-allocated. Hence, there is a need to allocate this new container in real time. It is also refered as a dynamic placement problem.

   To solve a dynamic placement, heuristics and dispatching rules are often used [5,18,43, 44]. In this scenario, a dispatching rule is considered as a function that determines the priorities of PMs that a container can be placed. However, dynamic placement is much complex than bin-packing problem [31]. Because of its dynamic nature, human designed heuristics are ill-equipped in approximating solutions when the environment has changed [47]. Multi-objective genetic algorithm (GA) [57] has been applied. However, GA is too slow for dynamic problem.

   We intend to develop a hyper-heuristic method - Genetic Programming (GP) technique [2] or artificial immune system [22]- to learn from the best previous allocation and automatic evolves dispatching rules to solve this problem. GP has been applied in generating dispatching rules for bin-packing problem [9, 47] and other scheduling problems [36]. The results have shown promising results.

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

4. Large-scale of static server consolidation problem,
   In this case, initialization and global consolidation are belonged to this category, since they are usually conducted in an off-line fashion. Since Cloud data center typically has hundreds of thousands PMs and more, static server consolidation is always very challenging. Many approaches have been proposed in the literature to resolve the problem. There are mainly two ways, both rely on distributed methods, hierarchical-based [24,35] and agent-based management systems [59]. The major problem in agent-based systems is that agents rely on heavy communication to maintain a high-level utilization. Therefore, it causes heavy load in the networking. Hierarchical-based approaches are the predominate methods. In essence, these approaches are centralized methods where all the states of machines within its region are collected and analyzed. The major disadvantage of hierarchical-based approaches is that it only provides local solutions. In fact, it is infeasible and unnecessary to check all the states of machines since the search space is too large and most machines do not need a change. This idea motivates a way to improving the effectiveness is to reduce the number of variables so that the search space is narrowed. In this thesis, we are going to investigate the way to eliminate the redundant information.

1.3 Research Goals

In this thesis, we aims at providing a series of approaches to continuously optimize the a joint allocation of VMs and containers that considers all three stages: Initialization, global consolidation, Dynamic consolidation. In addition, the static allocation normally involves with large amount of variables which is particular difficult to optimize. We also going to propose a method to solve this problem. These approaches combine element of AI planning, to ensure the objectives and constraint fulfillment, and of Evolutionary Computation, to evolve a population of near-optimal solutions. The research aims at determining a flexible way in creation of solutions to solve server consolidation problems. As discussed in the previous section, the research goal can be achieved in the following objectives and sub-objectives.

1. The initialization Problem,
   Currently, most research focus on VM-based server consolidation technique. They often modeled this problem as a vector bin-packing problem [61]. Container adds an extra layer of abstraction on top of VM. The placement problem has become a two-step procedure, in the first step, containers are packed into VMs and then VMs are consolidated into physical machines. These two steps are inter-related to each other.

   (a) *Modeling*
   Our first sub objective is to propose a description of model for the initialization problem. In order to achieve this goal, we will first review the related models including VM-based placement models and bi-level optimization models. Furthermore, we are going to consider the differences and design the constraints and other characteristics.

   (b) *Representation*
   Based on this new model, we are going to develop a representation that is suitable for this problem.

   (c) *New operators and searching mechanisms*
   In order to utilize Evolutionary Computation (EC) to solve this problem, we are going to develop searching mechanisms according to the nature of problem. In order to achieve this goal, we will design several new operators. In order to evaluate the quality of these components, we will perform analytical analysis on the result.

2. Global consolidation problem,
   As previous section mentioned, global consolidation is a time-dependent problem. The optimal consolidation in previous operation might lead to more migrations in the current consolidation. The robustness of a data center is particularly important. The robustness measures the stableness of result of consolidation.

   The objective of global consolidation has three sub-objectives. In order to measure the degree of robustness, we need to design a robustness measure. The second objective is to design static consolidation algorithm with considering its previous immediate result. The third objective extends the second objective to a more general case, considering both previous immediate and next allocation. The evaluation of algorithm is based on analytical analysis of fitness functions and robustness measure.

   (a) *Design a robustness measure*
   Previous studies only use simple measurement which counts the migration number between two static consolidation. This measurement aims at minimizing the

number of migration in a static placement process. It may cause more migration in the next consolidation. Therefore, it needs a time-aware measure of the robustness of system. A data center should be both consolidated as well as robustness after consolidate. Therefore, in this objective, the first sub-problem we are going to solve is to propose a robustness measure.

(b) *Design an consolidation method consider previous consolidation result*
Based on the robustness measure, we will first design an allocation method which takes the previous allocation into account. It has two optimization objectives, maximize the robustness and also minimize the energy consolidation.

(c) *Design a time-aware consolidation method*
We will generalize the previous sub-objective to a more general one: design a time-aware allocation method which takes previous and next allocation into consider.

3. Dynamic consolidation with a GP approach,

   (a) Construct Functional Set and Primitive Set for the problem
   As the basic component of a dispatching rule, primitive set contains the states of environment including: status of VMs, features of workloads. The functional set contains the operators which combines low level features.

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

4. Large-scale Static Consolidation Problem

   (a) Propose a preprocessing method to eliminate variables
   Current static consolidation takes all servers into consider which will lead to a scalability problem. In this objective, we will propose a method that categorizes servers so that only a small number of servers are considered. This approach will dramatically reduce the search space. The potential approaches that can be applied in this task are various clustering methods.

1.4 Published Papers

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

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.

1.5 Organisation of Proposal

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

Chapter 2

Literature Review

2.1 Background

Cloud computing has made one critical change in software industry, it separates the role of traditional service provider into service provider and infrastructure provider. As Wei [55] states, "one provides the computing of services, and the other provides the services of computing". Therefore, this separation add one more layer between service provider and users, as: Cloud providers, Cloud users (service providers), and End users.

The separate responsibility between Cloud providers and Cloud users has completely reformed the software industry [10] by providing three major benefits to Cloud users. First, Cloud users do not need upfront investment in hardwares (e.g PMs and networking devices) and pay for hardwares' maintenance. Therefore, it eliminates the risk of initial investment. Second, Cloud users do not need to worry about the limited resources which can obstruct the performance of their services when unexpected high demand occurs. Cloud providers off an elastic nature of Cloud which can dynamic allocates and releases resources for a software. Cloud users only need to pay the resources that they have used under a *pay-as-you-go* policy. Third, Cloud users can publish and update their applications at any location as long as there is an Internet connection. These advantages allow anyone or organization to deploy their softwares on Cloud in a reasonable price.

As previous section mentioned, our goal in this thesis is to help Cloud providers to increase their profit from data centers. Specifically, we focus on how to save money by cutting expenses of Cloud data centers.

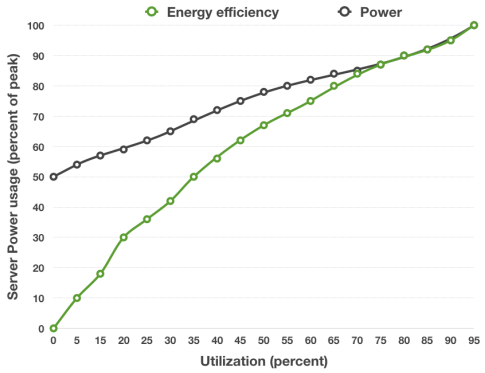


Figure 2.2: Disproportionate between utilization and energy consumption [3]

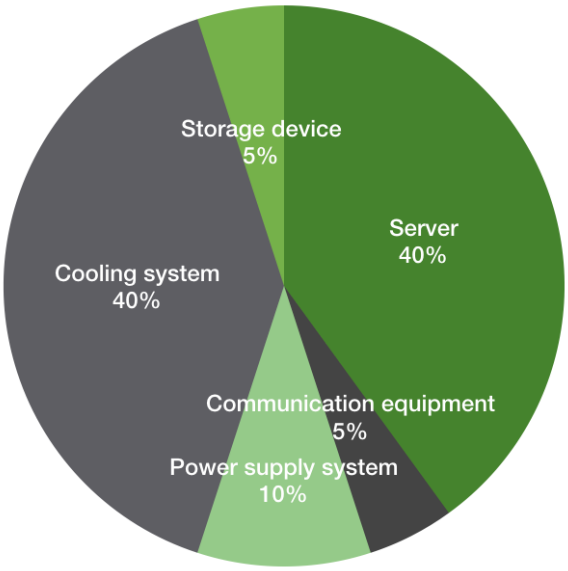


Figure 2.1: Energy consumption distribution of data centers [42]

Apart from upfront investment, energy consumption [27] is the major expense of data centers. Therefore, it is also the top concern of Cloud providers. Energy consumption is derived from several parts as illustrated in Figure 2.1. Cooling system and servers or PMs account for a majority of the consumption. A recent survey [12] shows that the recent development of cooling techniques have reduced its energy consumption and now server consumption has become the dominate energy consumption component.

According to Hameed et al [21], PMs are far from energy-efficient. The main reason for the wastage is that the energy consumption of PMs remains high even when the utilization are low (see Figure 2.2). Therefore, a concept of *energy proportional computing* [3] raised to address the disproportionate between utilization and energy consumption.

Virtualization [52] is the fundamental technology that enables Cloud computing. It partitions a physical machine's resources (e.g. CPU, memory and disk) into several isolated units called virtual machines (VMs) where each VM allows an operating system running on it. This technology rooted back in the 1960s' and was originally invented to enable isolated software testing, because VMs can provide good isolation which means applications running in co-located VMs within the same PM without interfering each other [46]. Soon,

people realized that it can be a way to improve the utilization of hardware resources: With each application deployed in a VM, a PM can run multiple applications. Later, a dynamic migration of VM was invented, which compresses and transfers a VM from one PM to another. This technique allows resource management in real time which inspires the strategy of server consolidation.

2.1.1 Resource Management in IaaS

The resource management in IaaS can be roughly separated into three [34, 50] which are applied in different scenarios: Application initialization, Prediction and Global consolidation, and Dynamic resource management (see Figure 2.3).

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

   In this problem, a set of applications or VMs are waiting in a queue. The resource capacity of the PM and usage by applications are characterized by a vector of resource utilizations including CPU, memory and etc. Then, the allocation system must select a minimum number of PMs to accommodate them so that after the allocation, the resource utilizations remain high. The problem is to consider the different combinations of applications so that the overall resource utilization is high. This problem is naturally modeled as a static bin-packing problem [13] which is a NP-hard problem meaning it is unlikely to find an optimal solution of a large problem.

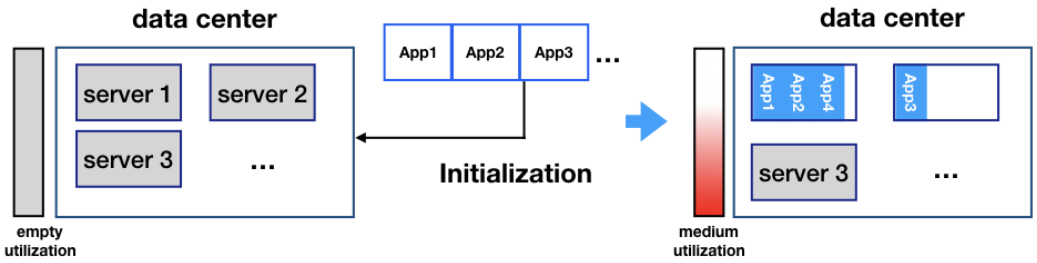
2. *Prediction and Global consolidation* is conducted periodically to adjust the current allocation of applications so that the overall utilization is improved.

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

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

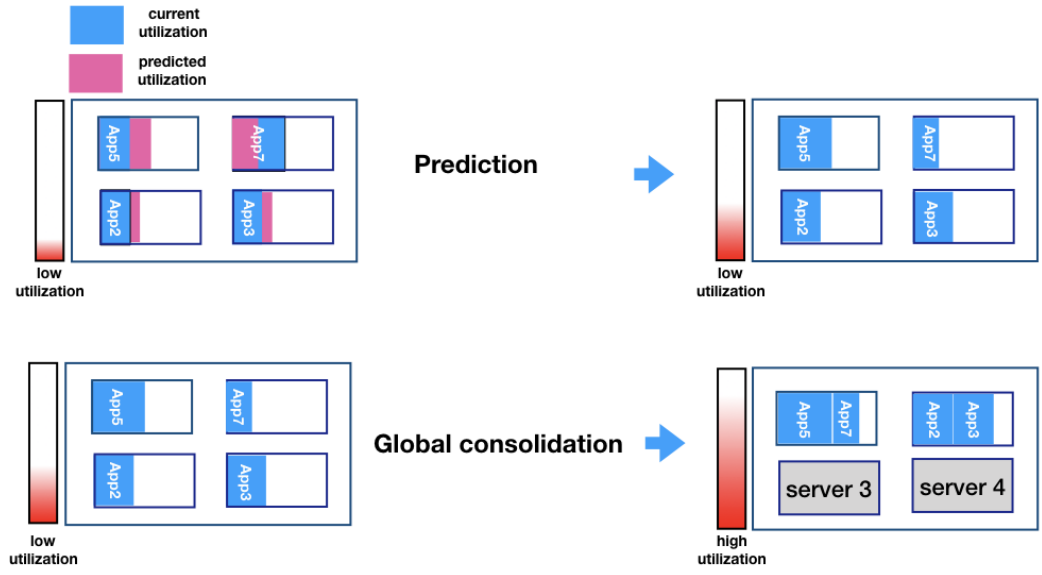
   In comparison with initialization, instead of new arrivals, the global consolidation considers the previous allocation. Another major difference is that global consolidation needs to minimize the differences of allocation before and after the optimization. This is because the adjustment of allocation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

3. *Dynamic resource management* Dynamic resource management is applied in three scenarios. **First**, it is applied when a PM is overloading. In order to prevent the QoS from dropping, an application is migrated to another PM. This is called hot-spot mitigation [34]. **Second**, it is applied when a PM is under-loading. Under-loading is when a PM is in a low utilization state normally defined by a threshold. At this moment, all the applications in the under-loading PM are migrated to other active PMs, so the PM



(a) Initialization

(b) Dynamic resource management



(c) Prediction and Consolidation

Figure 2.3: Three stages of resource management in IaaS

becomes empty and can be turned off. This is called dynamic consolidation. **Third**, it is applied when a PM having very high level of utilization while others having low. An adjustment is to migrate one or more application from high utilized PMs to low ones. This is called load balancing.

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

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

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

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

Finally, a consolidation plan includes four major items:

1. A list of existing PMs after consolidation
2. A list of new virtual machines created after consolidation
3. A list of old PMs to be turned off after consolidation
4. The exact placement of applications and services

By the nature of Cloud resource management, server consolidation techniques can also be categories into static and dynamic methods [53, 56]. Static method is a time consuming process which is often conducted off-line in a periodical fashion; initialization and global consolidation belong to this category. It provides a global optimization to the data center. Dynamic method adjusts PMs in real time. It often allocates one application at a time. Therefore, it can be executed quickly and often provides a local optimization to the data center.

2.2 Static Consolidation Methods

Static initialization, is also frequently referred to initial placement problem [23]. Whenever a request for provisioining of applications by one or more Cloud users. The resource management system schedules the applications into a set of PMs. Currently, most state-of-the-art research focus on VM-based placement, in this case, applications are installed in VMs. Therefore, "application placement" and "VM placement" are used interchangable in the literature.

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

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

Because of its NP-hard nature, several researchers propose well-known bin-packing heuristics First Fit (FF), First Fit Decreasing (FFD), Best Fit (BF) and etc. These algorithms have constant-factor approximation to bin-packing problems. However, VM placement is more complex than bin-packing problem [31],

Panigraph et al [37] study variants of First Fit Decreasing (FFD) algorithms and inspired by bad isntances for FFD-type algorithms, they propose a geometric heuristics which outperform FFD-based heuristics in most of cases.

2.3 Dyanmic Consolidation Methods

2.4 Evolutionary Computation Approaches on Server Consolidation Problem

Chapter 3

Preliminary Work

Service Oriented Architecture (SOA) and Cloud computing have significantly reformed the software industry. SOA provides a decentralized application architecture which allows software composition and reuse in a large, global scale. Meanwhile, Cloud computing provides a scalable, reliable, and flexible infrastructure to web services.

As the dramatic increase of web services and cloud facilities, the management of resources has become a critical issue. In recent years, as the power bill has become the largest fraction of the operating cost of Cloud facility [8], to reduce power consumption has become a paramount concern for Cloud service providers. In order to achieve that, a common approach is to re-allocate web services to a minimum number of physical machines (PMs) [15]. Therefore, idle computing servers are turned down or put into save mode. This optimization process, often called *consolidation* involves with two levels of delivery mode, Software as a service (SaaS) and Infrastructure as a service (IaaS). Because of the complexity, consolidation tasks for IaaS and SaaS are often considered as separated tasks with different objectives. For SaaS, the challenges concentrate on satisfying the Service Level Agreements (SLAs) with unpredictable requests using a minimum amount of resources. Whereas, for IaaS, the challenges are the VM migrations and energy conservation.

There are extensive algorithms proposed for SaaS and IaaS levels of resource allocation [17, 32]. Ref. [1] proposes a heuristic algorithm for service consolidation in a set of servers with minimizing costs while avoiding the overload of server and satisfying end-to-end response time constraints.

Ref. [19] proposes two algorithms for energy efficient scheduling of VMs in Cloud, including an exact VM allocation algorithm which is an extended Bin-Packing approach, and a migration algorithm based on integer linear programming.

However, as the two levels of resource allocation are interact with each other, we believe they cannot be separated. They should be considered as one global optimization with multi-objectives from the perspectives of both service providers and cloud providers. Therefore, in this paper, we first propose a model for solving *service resource allocation in Cloud (SRAC)*. Secondly, we propose a NSGA-II-based multi-objective algorithm with specifically designed operators to solve the problem. The two objectives are:

1. propose a model for solving IaaS and SaaS resource allocation together
2. propose a NSGA-II-based algorithm to solve SRAC.

The rest of the paper is organized as follows. Section 3.1 discusses the traditional approaches for IaaS and SaaS and the power model for VM allocation. It will also introduce related works of evolutionary multi-objective optimization techniques. Section 3.2 describes the definition of the SRAC problem. Section 3.3 introduces the representation and genetic

operators for SRAC problem. Section 3.4 illustrates the experiment design, results and discussions. Section 3.5 draws a conclusion and discusses the future work.

3.1 Background

3.1.1 Traditional approaches

Ref. [60] proposes a single-objective genetic algorithm to solve placement of service (SaaS) on physical machines. Their major contributions are three-fold. Firstly, they consider web services as a workflow and optimize the makespan of a workflow. Secondly, they design a representation to the problem. Thirdly, they do not only consider computing nodes, but storage nodes as well.

Ref. [58] develops a *Resource-Allocation-Throughput (RAT)* model for web service allocation. The *RAT model* mainly defines several important variables for an atomic service which represents a software component. Based on this model, firstly, an atomic service's throughput equals its coming rate if the resources of the allocated VM are not exhausted. Secondly, increasing the coming rate will also increase an atomic service's throughput until the allocated resource is exhausted. Thirdly, when the resource is exhausted, the throughput will not increase as request increasing. At this time, the virtual machine reaches its capacity.

Anton Beloglazov et al. [4] propose two algorithms for VM allocation. The first one is a bin-packing algorithm, called Modified Best Fit decreasing (MBFD) which is used when a new VM allocation request arrives. The second algorithm, named Minimization of Migration, is used to adjust the current VMs allocation according to the CPU utilization of a physical machine. Their experiments have shown that these methods lead to a substantial reduction of energy consumption in Cloud data centers.

3.1.2 Power Model

Shekhar's research [49] 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 [6] develops an energy model to profile the power of a VM. They monitor the subcomponents of a VM which includes: CPU, cache, disk, and DRAM and propose a linear model (Eq 3.1). Total power consumption is a linear combination of the power consumption of CPU, cache, DRAM and disk. The parameters α and β are determined based on the observations of machine running CPU and IO intensive jobs.

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

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

Anton Beloglazov et al. [4] propose a comprehensive energy model for energy-aware resource allocation problem (Eq 3.2). P_{max} is the maximum power consumption when a virtual machine is fully utilized; k is the fraction of power consumed by the idle server (i.e. 70%); and u is the CPU utilization. This linear relationship between power consumption and CPU utilization is also observed by [28, 41].

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

3.1.3 Multi-objective Evolutionary Optimization

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

$$\min \ \vec{f}(\vec{x}) = (f_1(\vec{x}), \ldots, f_m(\vec{x})), \tag{3.3}$$

$$s.t. \ \vec{x} \in \Omega. \tag{3.4}$$

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

Multi-objective Evolutionary Optimization Algorithm (MOEA) are ideal for solving multi-objective optimization problems [16], because MOEAs work with a population of solutions. With an emphasis on moving towards the true Pareto-optimal region, a MOEA algorithm can be used to find multiple Pareto-optimal solutions in one single simulation run [26]. Therefore, this project would employ MOEA approaches. This is also the first time to employ MOEAs technique for SRAC problem.

3.2 Problem Description

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

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

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

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

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

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

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

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

In order to satisfy the performance requirement, Service providers often define Service Level Agreements (SLAs) to ensure the service quality. In this work, we define throughput

as a SLA measurement [38]. Throughput denotes the number of requests that a service could successfully process in a period of time. According to *RAT* model, the throughput is equal to the number of requests when the allocated resource is sufficient. Therefore, if a VM reaches its utilization limitation, it means that the services have been allocated exceedingly. Therefore, all services in that VM suffer from performance degradation.

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

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

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

Hard constraint

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

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

Soft constraint

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

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

3.3 Methods

As we have discussed, Multi-objective Evolutionary Algorithms are good at solving multi-objective problems and NSGA-II [14] has shown his effective and efficiency. NSGA-II is a well-known MOEA that has been widely used in many real-world optimization problems. In this paper we also adopt NSGA-II to solve the SRAC problem. We first propose a representation and then present a NSGA-II based algorithm with novel genetic operators.

3.3.1 Chromosome Representation

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

Figure 3.1 shows an example individual which contains seven service allocations. Each allocation of a service is represented as a pair where the index of each pair represents the

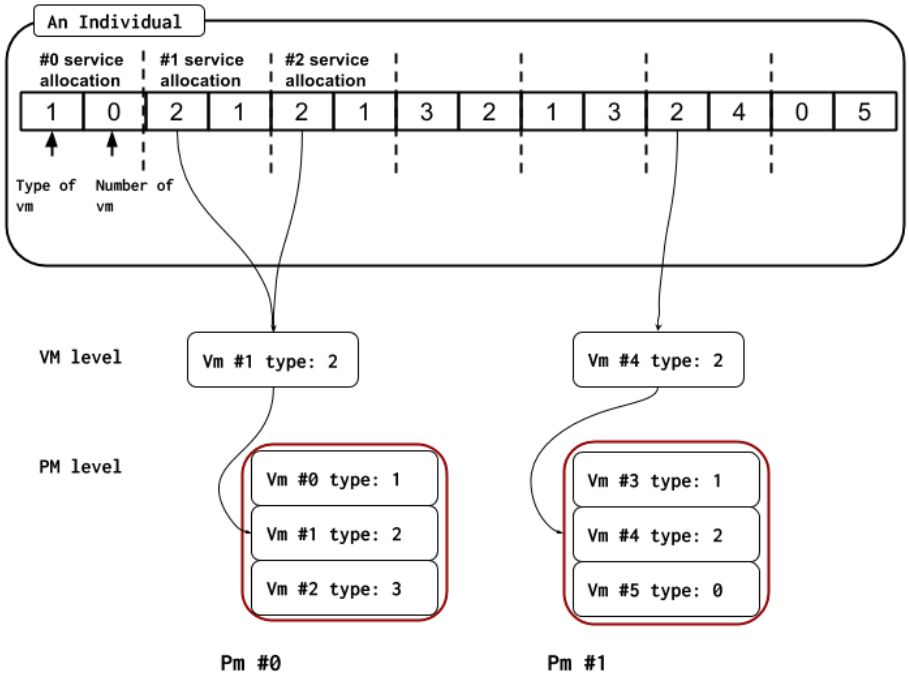


Figure 3.1: An example chromosome representation

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

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

3.3.2 Initialization

The initialization (see Alg 1) is designed to generate a diverse population. In the first step, for each service, it is able to find the most suitable VM type which is just capable of running the service based on its resource requirements. In the second step, based on the suitable VM type, a stronger type is randomly generated. If there exists a VM with that type, the service is either deployed in the existing VM or launch a new VM. We design a

Algorithm 1 Initialization

Inputs:
VM CPU Time VA and memory VM,
Service CPU Time A and memory M
consolidation factor c
Outputs: A population of allocation of services

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

consolidation factor c which is a real number manually selected from 0 to 1 to control this selection. If a random number u is smaller than c, the service is consolidated in an existing VM.

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

3.3.3 Mutation

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

The consolidation is conducted with a roulette wheel method which assigns fitness value to each VM according to the reciprocal of its current utilization. The higher the utilization, the lower the fitness value it is assigned. Therefore, a lower utilization VM has a greater probability to be chosen. At last, if a new VM is launched, it will not be placed at the end of VM lists. Instead, it will be placed at a random position among the VMs. The reason is illustrated in Figure 3.2. In the example, VM #2 is mutated into a new type and be placed at the end of the VM list. However, because of the size of VM #3 is too large for PM #0, the hollow in PM #0 will never be filled. This problem can be solved with the random insertion method.

3.3.4 Violation control method

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

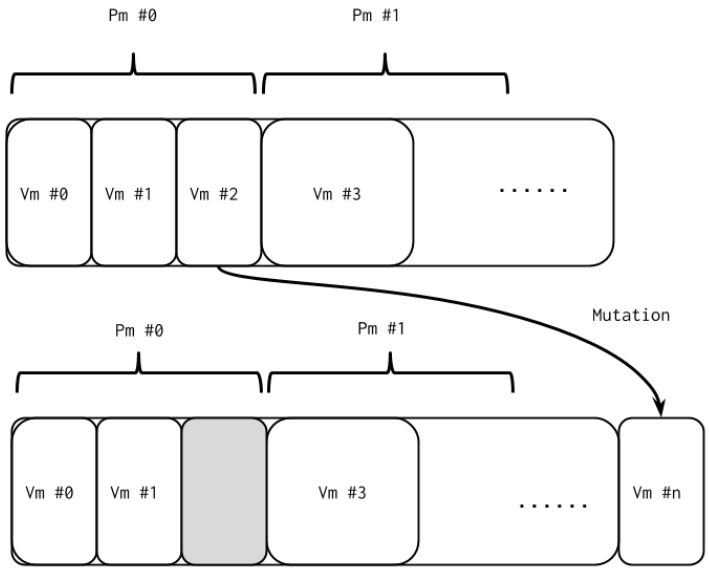


Figure 3.2: An example mutation without insertion that causes a lower resource utilization

Algorithm 2 Mutation

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

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

3.3.5 Selection

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

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

An individual with no or less violation is always selected. This method has been proved effective in the original NSGA-II paper [14].

3.3.6 Fitness Function

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

3.3.7 Algorithm

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

That is, a random switch of chromosome would completely destroy the order of VMs, hence, no useful information will be preserved. Therefore, we only apply mutation as the exploration method. Then, the algorithm becomes a parallel optimization without much interaction between its offspring, which is often addressed as Evolutionary Strategy [29].

Algorithm 3 NSGA-II for SRAC

Inputs:
VM CPU Time VA and memory VM,
PM CPU Time PA and memory PM,
Service CPU Time A and memory M
consolidation factor c
Outputs: A Non-dominated Set of solutions

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

3.4 Experiment

3.4.1 Dataset and Problem Design

This project is based on both real-world datasets *WS-Dream* [63] and simulated datasets [7]. The *WS-Dream* contains web service related datasets including network latency and service frequency (request coming rate). In this project, we mainly use the service frequency matrix. For the cost model, we only consider the rental of virtual machines with fixed fees (monthly rent). The configurations of VMs are shown in Table 3.2, the CPU time and memory were selected manually and cost were selected proportional to their CPU capacity. The maximum

Table 3.1: Problem Settings

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

Table 3.2: VM configurations

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

PM's CPU and memory are set to 3000 and 8000 respectively. The energy consumption is set to 220W according to [7].

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

Selection Method with violation Control vs. without violation control

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

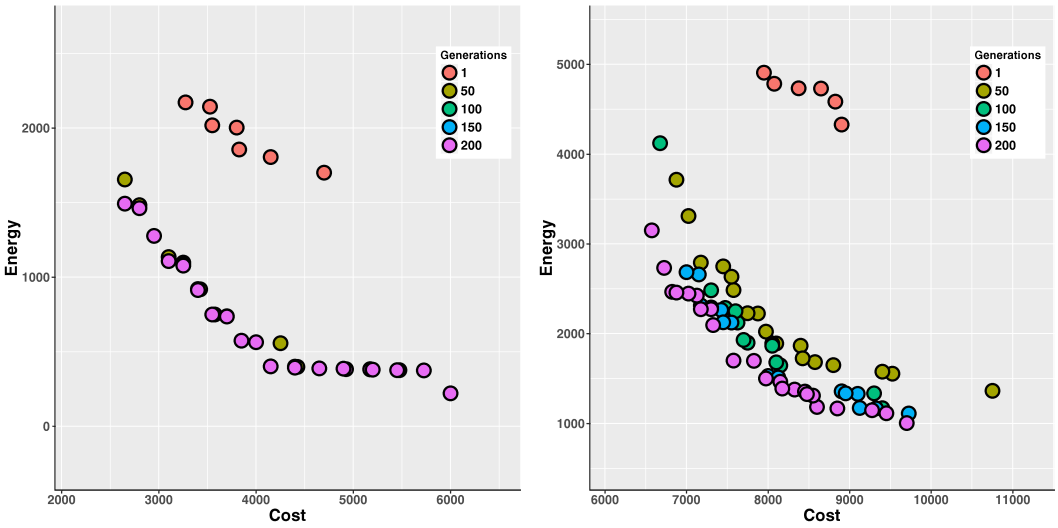
The experiments were conducted on a personal laptop with 2.3GHz CPU and 8.0 GB RAM. For each approach, 30 independent runs are performed for each problem with constant population size 100. The maximum number of iteration is 200. k equals 0.7. We set mutation rate and consolidation factor to 0.9 and 0.01.

3.4.2 Results

Table 3.3: Comparison between two Mutation methods

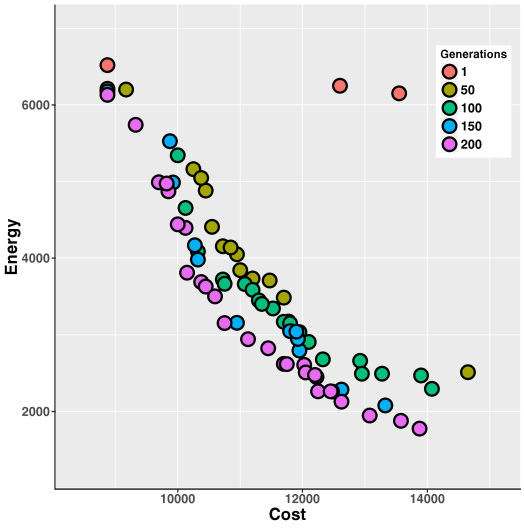
Problem	roulette wheel mutation		Greedy mutation	
	cost fitness	energy fitness	cost fitness	energy fitness
1	2664.6 ± 66.4	1652.42 ± 18.2	2661.7 ± 56.9	1653.2 ± 18.2
2	6501.1 ± 130.2	4614.0 ± 110.7	6495.37 ± 110.7	4132.5 ± 80.4
3	8939.2 ± 118.5	6140.7 ± 204.0	9020.5 ± 204.0	5739.6 ± 148.6
4	11633.7 ±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

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

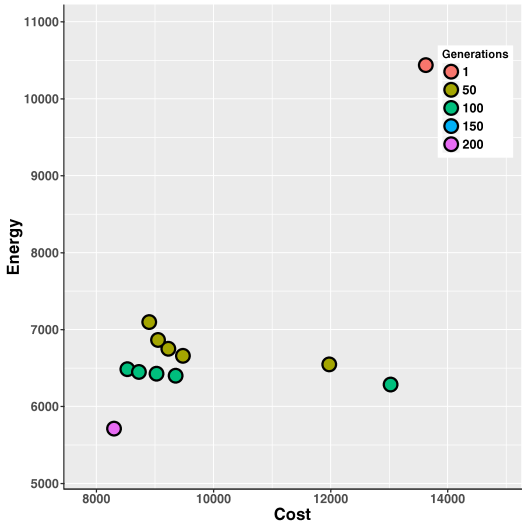


(a) Problem 1

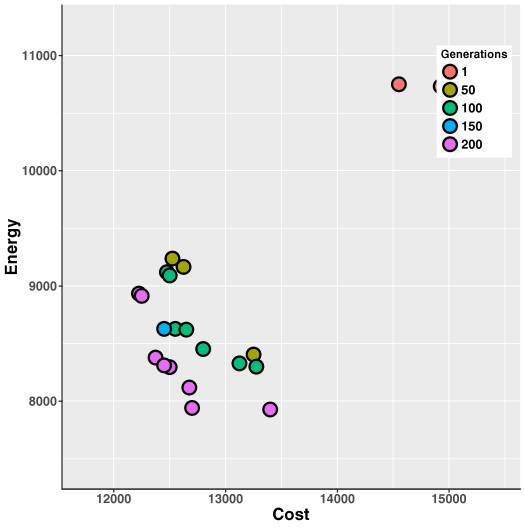
(b) Problem 2



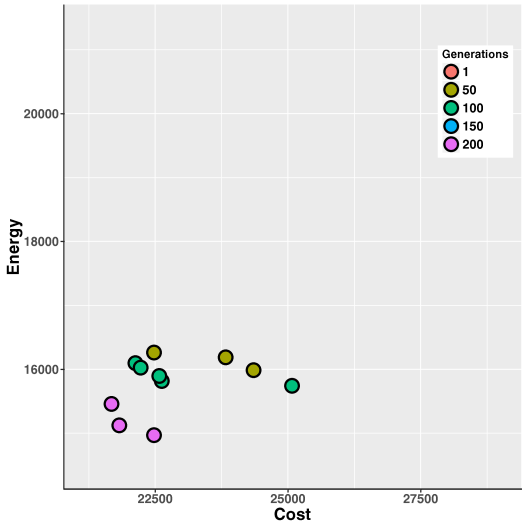
(c) Problem 3



(d) Problem 4

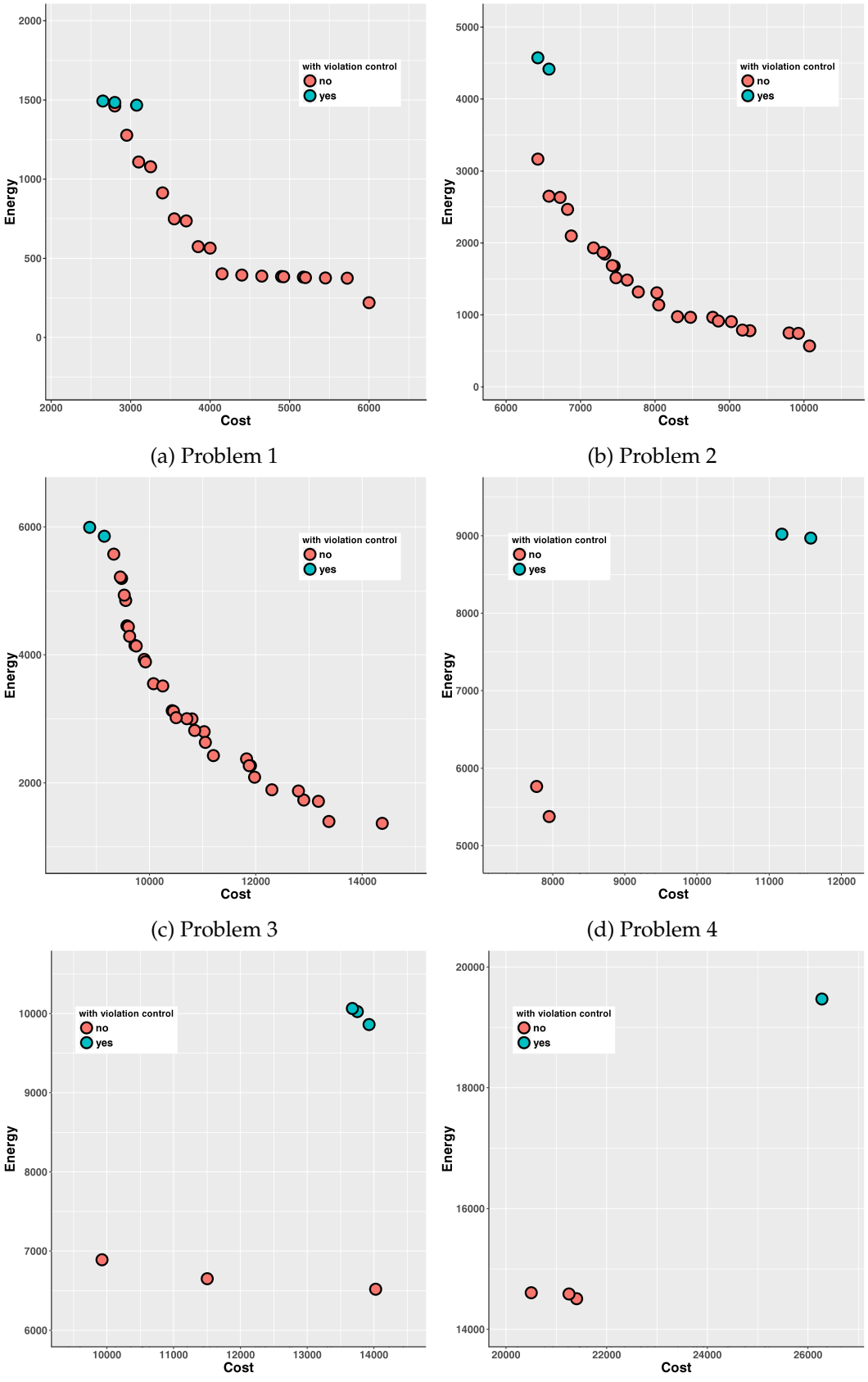


(e) Problem 5



(f) Problem 6

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



(a) Problem 1

(b) Problem 2

(c) Problem 3

(d) Problem 4

(e) Problem 5

(f) Problem 6

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

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

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

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

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

Mutation with roulette wheel vs. Mutation with greedy algorithm

Table 3.3 shows the fitness value comparison between mutation methods. According to statistics significant test, there is little difference between methods. The possible reason is the consolidation factor is set to 0.01. In each mutation iteration, there is only 1% probability that a service will be consolidated in an existed VM, therefore, the influence between different consolidation strategies is trivial.

3.5 Conclusion

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

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

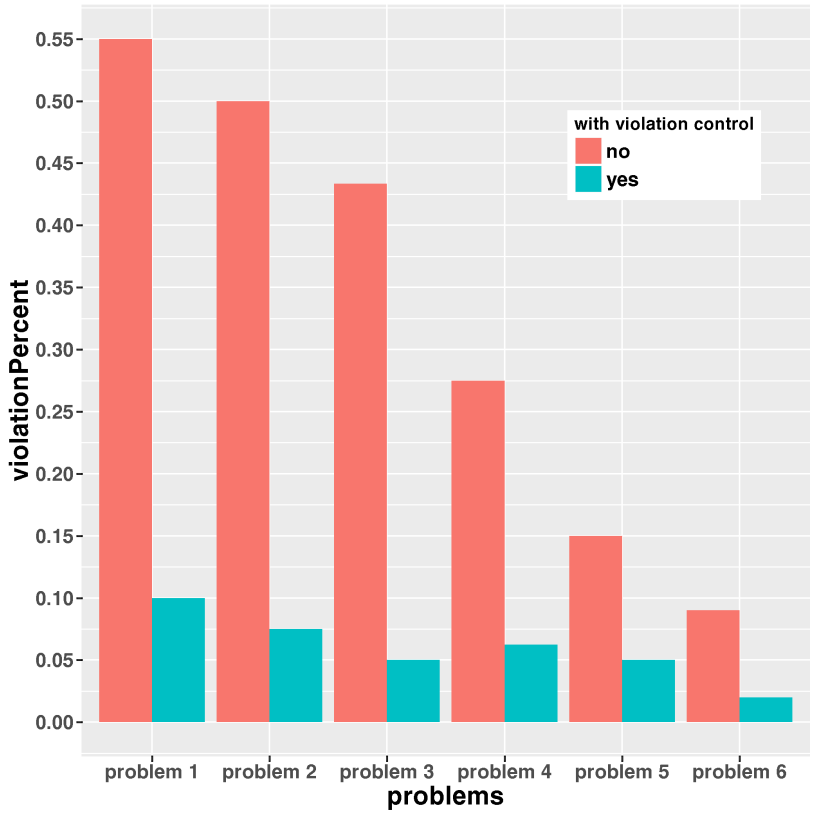


Figure 3.5: Violation Percentage comparison between selection with violation control and without violation control

Chapter 4

Proposed Contributions and Project Plan

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

1. Propose a new model for the joint allocation of container and VM problem. New representations will be proposed to problem. The first EC-approach to solve the problem. Reduce the energy consumption of data centers.
2. Propose a new robustness measure for time-dependent server consolidation problem. New optimization algorithms not only consider the two-level of packing problem but also take the previous and next consolidation into considered.
3. Develp a GP-based hyper-heuristic approach to automatically generate dispatching rules for dynamic server consolidation problem. New functional and primitive set for constructing useful dispatching rules will be studied. New genetic operations and representations will be proposed.
4. Develop a preprocessing algorithm to reduce the number of variables in a large scale static consolidation problem without scacrificing too much performance. Useful features of server consolidation problem will be studied.

4.1 Overview of Project Plan

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

4.2 Project Timeline

The phases included in the plan above are estimated to be completed following the timeline shown in Figure **??**, which will serve as a guide throughout this project. Note that part

of the first phase has already been done, therefore the timeline only shows the estimated remaining time for its full completion.

4.3 Thesis Outline

The completed thesis will be organised into the following chapters:

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

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

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

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

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

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

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

4.4 Resources Required

4.4.1 Computing Resources

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

4.4.2 Library Resources

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

4.4.3 Conference Travel Grants

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

Bibliography

[1] ANSELMI, J., AMALDI, E., AND CREMONESI, P. Service consolidation with end-to-end response time constraints. In *2008 34th Euromicro Conference Software Engineering and Advanced Applications* (Sept 2008), pp. 345–352.

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

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

[4] BELOGLAZOV, A., ABAWAJY, J., AND BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems 28*, 5 (2012), 755–768.

[5] BELOGLAZOV, A., ABAWAJY, J. H., AND BUYYA, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Comp. Syst. 28*, 5 (2012), 755–768.

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

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

[8] BROWN, R. E., MASANET, E. R., NORDMAN, B., TSCHUDI, W. F., SHEHABI, A., STANLEY, J., KOOMEY, J. G., SARTOR, D. A., AND CHAN, P. T. Report to congress on server and data center energy efficiency: Public law 109-431.

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

[10] BUYYA, R., YEO, C. S., VENUGOPAL, S., BROBERG, J., AND BRANDIC, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems 25*, 6 (June 2009), 599–616.

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

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

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

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

[15] DESAI, S., BAHADURE, S., KAZI, F., AND SINGH, N. Article: Multi-objective constrained optimization using discrete mechanics and NSGA-II approach. *International Journal of Computer Applications 57*, 20 (2012), 14–20.

[16] DESAI, S., BAHADURE, S., KAZI, F., AND SINGH, N. Article: Multi-objective constrained optimization using discrete mechanics and NSGA-II approach. *International Journal of Computer Applications 57*, 20 (2012), 14–20.

[17] DUBOIS, D. J., AND CASALE, G. Autonomic provisioning and application mapping on spot cloud resources. In *Cloud and Autonomic Computing (ICCAC), 2015 International Conference on* (2015), IEEE, pp. 57–68.

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

[19] GHRIBI, C., HADJI, M., AND ZEGHLACHE, D. Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (2013), IEEE, pp. 671–678.

[20] GUPTA, R., BOSE, S. K., SUNDARRAJAN, S., CHEBIYAM, M., AND CHAKRABARTI, A. A Two Stage Heuristic Algorithm for Solving the Server Consolidation Problem with Item-Item and Bin-Item Incompatibility Constraints. *IEEE SCC* (2008).

[21] HAMEED, A., KHOSHKBARFOROUSHHA, A., RANJAN, R., JAYARAMAN, P. P., KOLODZIEJ, J., BALAJI, P., ZEADALLY, S., MALLUHI, Q. M., TZIRITAS, N., VISHNU, A., KHAN, S. U., AND ZOMAYA, A. Y. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing 98*, 7 (2016), 751–774.

[22] HOFMEYR, S. A., AND FORREST, S. Architecture for an Artificial Immune System. *Evolutionary Computation 8*, 4 (2000), 443–473.

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

[24] JUNG, G., HILTUNEN, M. A., JOSHI, K. R., SCHLICHTING, R. D., AND PU, C. Mistral - Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. *ICDCS* (2010), 62–73.

[25] JUNG, G., JOSHI, K. R., HILTUNEN, M. A., SCHLICHTING, R. D., AND PU, C. Generating Adaptation Policies for Multi-tier Applications in Consolidated Server Environments. *ICAC* (2008).

[26] KANAGARAJAN, D., KARTHIKEYAN, R., PALANIKUMAR, K., AND DAVIM, J. Optimization of electrical discharge machining characteristics of wc/co composites using non-dominated sorting genetic algorithm (NSGA-II). *The International Journal of Advanced Manufacturing Technology 36*, 11-12 (2008), 1124–1132.

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

[28] 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.

[29] 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.

[30] LI, B., AND JIANXIN, L. *An Energy-saving Application Live Placement Approach for Cloud Computing Environment*. ACM International Conference on Cloud Computing, 2009.

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

[32] MAZUMDAR, S., AND PRANZO, M. Power efficient server consolidation for cloud data center. *Future Generation Computer Systems 70* (2017), 4 – 16.

[33] 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.

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

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

[36] 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.

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

[38] PASCHKE, A., AND SCHNAPPINGER-GERULL, E. A categorization scheme for sla metrics. *Service Oriented Electronic Commerce 80*, 25-40 (2006), 14.

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

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

[41] RAGHAVENDRA, R., RANGANATHAN, P., TALWAR, V., WANG, Z., AND ZHU, X. No power struggles: Coordinated multi-level power management for the data center. In *ACM SIGARCH Computer Architecture News* (2008), vol. 36, ACM, pp. 48–59.

[42] 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.

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

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

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

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

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

[48] SPEITKAMP, B., AND BICHLER, M. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on services ...* (2010).

[49] SRIKANTAIAH, S., KANSAL, A., AND ZHAO, F. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems* (2008), vol. 10, San Diego, California, pp. 1–5.

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

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

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

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

[54] WANG, Y., AND XIA, Y. Energy Optimal VM Placement in the Cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD* (2016), IEEE, pp. 84–91.

[55] WEI, Y., AND BLAKE, M. B. Service-Oriented Computing and Cloud Computing - Challenges and Opportunities. *IEEE Internet Computing 14*, 6 (2010), 72–75.

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

[57] XU, J., AND FORTES, J. A. B. Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments. *GreenCom/CPSCom* (2010).

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

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

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

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

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

[63] ZHANG, Y., ZHENG, Z., AND LYU, M. Exploring latent features for memory-based QoS prediction in cloud computing. In *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on* (2011), pp. 1–10.