Profile-based Application Management For Green Data Centres

A THESIS SUBMITTED TO

THE SCIENCE AND ENGINEERING FACULTY

OF QUEENSLAND UNIVERSITY OF TECHNOLOGY

IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Meera Vasudevan

School of Electrical Engineering and Computer Science

Science and Engineering Faculty

Queensland University of Technology

2016

Profile-based Application Management For Green Data Centres

A THESIS SUBMITTED TO
THE SCIENCE AND ENGINEERING FACULTY
OF QUEENSLAND UNIVERSITY OF TECHNOLOGY
IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Meera Vasudevan

Supervisors: Prof. Yu-Chu Tian, Dr Maolin Tang and Prof. Erhan Kozan

School of Electrical Engineering and Computer Science
Science and Engineering Faculty
Queensland University of Technology

2016

Copyright in Relation to This Thesis

© Copyright 2016 by Meera Vasudevan. All rights reserved.

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

QUT Verified Signature

Signature:

Date: 11/08/2016

To my family

Abstract

Data Centres are essential to modern society, making the technologies we use everyday possible. This has led to the proliferation of data centres around the world. The increasing number of people and organizations using these data centres has resulted in rising energy consumption and consequent increase in electricity bills and carbon footprint. As a result, research on "green" initiatives that reduce energy consumption whilst maintaining performance levels of data centres is pressing. Application management on virtualized data centres have become an important technique for researchers due to its flexibility, ease of implementation and compatibility with other data centre energy measures. However, inefficient application assignment to virtual machines (VMs) adversely affects the Quality of Service (QoS) such as resource utilization and workload balance of data centres. Therefore, this research explores application management methods for reducing energy consumption whilst maintaining performance-efficiency.

This thesis presents a profile-based application management framework for energy-efficient ("green") data centres. Profiles provide prior knowledge of the run-time characteristics of applications, VMs and servers. This is used to inform static and dynamic application assignment, and consolidation of applications. Static application assignment to VMs assume perfect future knowledge of workload, whereas dynamic application assignment to VMs consider real-time variable workload. Consolidation strengthens profile-based application management framework by assigning applications onto reduced number of active VMs.

The thesis first explores the concept and building of profiles such that future workload can be predicted and assignments can be planned in advance. Next, a Repairing Genetic Algorithm using profiles is developed for the implementation of static application assignment. Then, a profile-based dynamic application assignment that handles real-time applications and workload is developed. Finally, application consolidation is implemented by a Local Search Optimization heuristic. The profile-based application assignment framework is integrated into

a three-layer energy management system incorporating VM placement to derive actual energy savings. Experiments are conducted to demonstrate the effectiveness of the proposed profile-based application management framework.

Keywords

Data Centre, Application Assignment, Virtual Machine, Energy Efficiency, Resource Scheduling, Optimization, Evolutionary Algorithm, Genetic Algorithm, Static Assignment, Dynamic Assignment, Consolidation

Acknowledgments

PhD is and always will be the first amazing opportunity that I have embarked on towards a brighter future. The journey has been tough at times and exciting at others. Through it all, I have learned many things about my work and more importantly about myself. I am truly happy to have had a chance to experience it. It would not have been possible without the people who helped me along the way and will always be a part of my achievements. Firstly, I would like to express my sincere thanks to my principal supervisor, Professor Glen Tian who gave me the incredible opportunity to undertake my PhD and constantly provided invaluable guidance and advice throughout my candidature. I would like to express my gratitude to associate supervisor, Dr Maolin Tang for his valuable support and expert advice, especially on genetic algorithms. I would also like to thank associate supervisor, Prof. Erhan Kozan for his patience, kindness and guidance on mathematical optimizations and operations research. I would also like to extend my thanks to my fellow graduate students in the Data Science department for their friendship and support.

My heartfelt thanks to my beautiful family for the immeasurable love, immovable support and patience shown to me. Dad, thank you for teaching me, supporting me in my endeavours and being patient with me. Mom, words cannot express how much you mean to me. Thank you for being the beautiful soul that you are. My beautiful sister, thank you for being my best friend and bringing me brownies when I spent days writing this thesis. I would also like to thank my grandmother and grandfather for their love and support. Grams, your funny anecdotes brightened up even my darkest days. I love you all dearly and thank you for making my life filled with joy and happiness.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data centres are critical computing infrastructures that have become essential to modern society through services like cloud computing, big data and the Internet of Things. However, the proliferation of data centres with inefficient application management results in high energy consumption and carbon footprint [Dayarathna et al., 2016]. The cost associated with energy consumption claims a large portion of the total operational costs of a data centre [Markets and Markets, 2015]. However, implementation of green strategies is capable of reducing energy consumption and cost by 40% [Whitney and Delforge, 2014]. This thesis develops a profile-based application management framework for energy-efficient ("green") data centres.

To achieve this objective, the methods of application assignment and Virtual Machines (VM) placement in data centres play a significant role. Many research activities on energy-efficient data centres have been focused more on VM placement and less on application assignment. Existing technologies implemented in data centres do not consider the dynamic characteristics of applications and VMs. Without such knowledge, assigning applications to VMs with energy-efficiency degrades performance such as resource utilization. However, this can be improved if we have prior knowledge of run-time characteristics of applications and VMs. Profiles will serve this purpose. They capture the main characteristics of applications, VMs and servers over a period of time and gives sufficient information to estimate and allocate resources. With such knowledge, application assignment to VMs can be made more energy-efficient.

Therefore, this thesis explores a concept of using Profiles for energy- and performance-efficient application assignment to Virtual Machines (VMs). First, building of profiles using real data centre workload logs is outlined. Then, using profiles in a three-layer data centre

architecture to implement static and dynamic application assignment, and application consolidation strategies are discussed. Profile-based static application assignment assumes perfect future knowledge and is used to design a Repairing Genetic Algorithm (RGA). Profile-based dynamic application assignment considers real-time applications and variable workload. Application consolidation considers VM workload fluctuations and re-assigns applications to reduce the number of active VMs. The three developed strategies form the complete solution to profile-based application management and are evaluated to demonstrate actual energy savings. The complete three-tiered profile-based assignment solution results in greener data centres with high Quality of Service (QoS).

This Chapter is organized as follows. Section 1.1 discusses the research background. Section 1.2 presents the statement of the research problem. The research significance and motivation is discussed in Section 1.3. Section 1.4 states the contributions made during the course of this PhD. The thesis organization and list of research articles derived from this research are presented in Sections 1.4 and 1.6.

1.1 Research Background

The cloud computing era caters to a rapidly growing online population and the subsequent explosion of data. This has revolutionized the Information and Communication Technology (ICT) industry. As a result, modern economy relies heavily on data centre and cloud systems for computing, storage and network services to name a few. Data centres are facing an escalation of services related to high-powered technologies such as artificial intelligence, IPv6, Remote Direct Memory Access (RDMA), virtualizations and cloud solutions. The proliferation of data-intensive applications and high performance computing (HPC) has resulted in large-scale deployment of data centres around the world with rapidly increasing number of connected file servers, database servers, storage servers, network components, and power and cooling systems.

The everyday usage of data centres by the growing number of Internet users leads to massive amounts of energy consumption [Delforge, 2014]. Consequently, exorbitant operation costs and high carbon footprint in the form of carbon dioxide (CO_2) emissions are inevitable. The electricity consumed by data centres is predicted to rise from 7% to 12% of the global electricity consumption by 2017 according to Corcoran and Andrae [2013]. Overall, data centres consume

1.1% to 1.5% of the world's total electricity consumption [Koomey, 2011]. More than 35% of the current data centre operational expenses are accounted for by energy consumption. This figure is projected to double in a few years [Vaid, 2010]. Buyya et al. [2013] confirm that energy costs of powering a data centre doubles every five years. The Natural Resources Defense Council (NRDC) released a report disclosing that 91 billion kWh of electrical energy was consumed by data centres in 2013. This statistic is projected to increase by 53% by year 2020 [Whitney and Delforge, 2014].

The necessity for green measures has become very real and emerging according to Rafiei and Bakhshai [2012]. Up to 40% of energy savings can be realized on deployment of energy-efficient measures [Whitney and Delforge, 2014]. Le et al. [2009] concluded that deploying green initiatives at data centres decreases the carbon footprint by 35% at only 3% cost increase. According to the Greenpeace report by Cook and Pomerantz [2015]; hyper-scale companies like Apple, Google and Facebook lead the charge in deploying green solutions to reduce energy consumption and carbon footprint. Many small- and medium-scale data centres deploy virtualization, however most are wary about initiating energy-efficient strategies [Nunez, 2014]. This is due to risks involving data reliability and overhead of re-booting switched-off servers. The Green Grid, a global consortium by means of multiple alliances with international organizations have been established for the express purpose of addressing the rising energy consumption of data centres. One of its alliances is with the Japan Data Centre Council (JDCC), established in 2008, with the main objective of identifying and undertaking measures to combat the ever increasing energy consumption of data centres. The Green Grid Japan Data Centre Award [The Green Grid, 2013] is effective in encouraging numerous businesses and organizations to implement energy efficient measures to their data centres. The Carbon Reduction Commitment (CRC) energy efficiency scheme introduced in the UK aims at spreading awareness of the harmful effects of carbon emissions and induces British based companies to adopt green management strategies. The CRC energy efficiency scheme aims at reducing carbon emissions by 60% over the next four decades [CarbonZone, 2010].

Data centre energy consumption is composed of fixed and variable parts. The fixed energy consumption part depends on physical resources such as servers, communication network elements, cooling systems and power supply. The variable energy consumption part depends on the IT side of data centres such as application, VM and server resource usage [Huang and Masanet, 2015, Orgerie et al., 2014]. Green measures to reduce fixed energy involves

hardware implementations such as deployment of fuel cells and renewable energy to power data centres. However, such measures require a large proportion of time and money from installation, execution and maintenance budget perspectives [Binkley, 2016]. Moreover, the larger part of the energy supplied is consumed by the IT side of data centres [Outlook, 2014]. Green measures to reduce variable energy involves software implementations such as energy-efficient application management and energy-aware resource management [Beloglazov et al., 2011]. Consequently, software implementations provide a more logical and economical solution in terms of ease of implementation, maintenance and modification, and associated costs. Due to its flexibility and economical nature, these software energy-efficient strategies can be used by a wide spectrum of large, medium and small data centres [Costa-Requena et al., 2014, McFarlane, 2015].

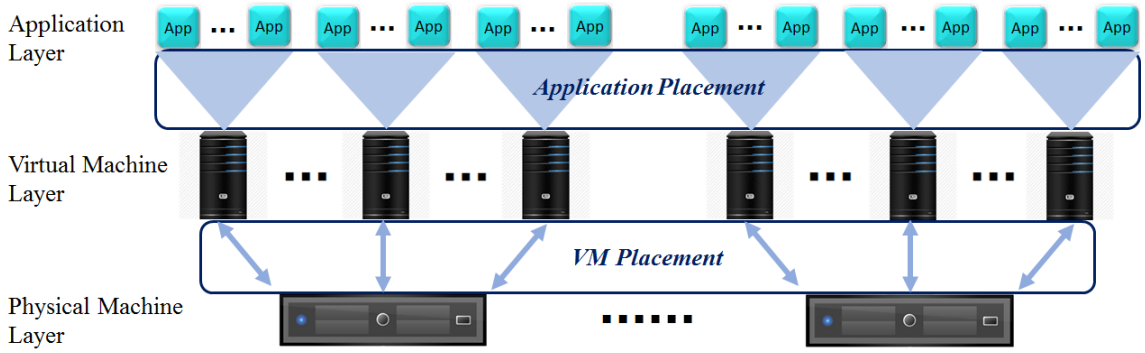


Figure 1.1: Three-layer data centre architecture.

Software energy management in data centres can be implemented at three layers: application, virtual machine (VM) and server or physical machine (PM), as shown in Figure 1.1. Server resource usage, ON/OFF operations, sleep cycles, cooling and dynamic voltage frequency scaling (DVFS) are dealt with in the PM Layer. The VM layer is responsible for VM management, including VM placement, sizing and migration. The application layer assigns incoming applications requested by cloud consumers or data centre users to VMs for execution. Energy-efficient IT solutions such as virtualization, resource scheduling, server consolidation, application management etc. are deployed in one or more of these layers.

This thesis focusses on reducing the variable energy consumption of data centres through implementation of a three-layered profile-based application management framework. Profiles provide prior knowledge of the run-time workload characteristics and can be created with low-computation overhead using readily available data centre workload logs.

1.2 Statement of the Research Problem

The research objective of this thesis is to develop a profile-based application management framework for energy-efficient data centres. Assignment to virtual machine will maintain a trade-off between energy consumption and Quality of Service (QoS). (The term assignment refers to application assignment throughout this thesis, unless otherwise specified.)

To achieve the objective, the following questions have to be answered;

1. **Profiling Problem:** What is the most computationally-efficient method of building profiles?
2. **Static Assignment Problem:** How to make use of profiles for application assignment to VMs?
3. **Dynamic Assignment Problem:** How to implement profile-based assignment for real-time applications with dynamic workload?
4. **Consolidation Problem:** How to consolidate profile-based assignment such that some VMs can be emptied and shut down?

1.3 Research Significance and Motivation

High energy consumption results in substantial operational, maintenance and cooling costs of data centre components. Data centres provide computational, processing and storage services to business, scientific and consumer domains. However, these services continue to increase on a daily basis, overrunning the existing infrastructure capacity and surpassing the current energy requirements [Gates, 2015]. This contributes to excessive heat discharge which leads to the reduced lifetime of IT components thereby adversely affecting the reliability of data centres. Another important factor that has researchers attempting to find energy-efficient solutions for data centres is the increase in carbon emissions with potential impacts on climate change [Whitehead et al., 2014].

This research targets widely deployed small- to medium-density data centres. Hyper-scale large data centres run by Internet giants such as Microsoft, Google, Dell, Facebook etc. make up for only 5% global data centre energy usage as seen from Figure 1.2. The remaining 95%

is attributed to small- and medium-scale data centres operated by thousands of businesses, universities and government agencies. These data centres distinctly lack energy-efficient initiatives when compared to well-managed hyper-scale large data centres [Whitney and Delforge, 2014]. Nunez [2014] lists the reasons for small- and medium-scale data centre energy inefficiency: 1) companies unaware that data centre claims 30-50% of company electricity bill; 2) lack of resource and expertise and 3) data reliability risk involved in switching-off servers and overhead time of rebooting servers (sleep cycles).

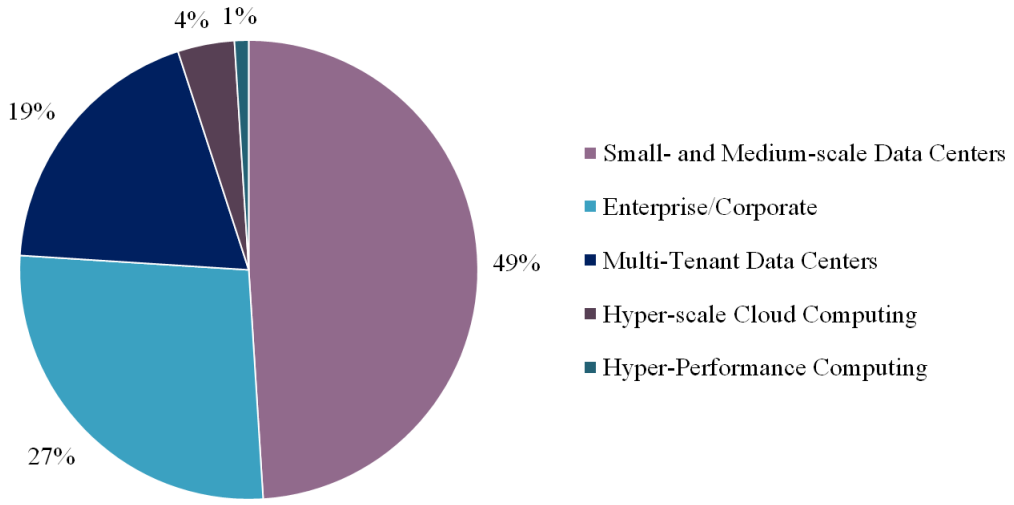


Figure 1.2: Data Centre World Distribution [Delforge, 2014].

Currently, there is a widespread awareness on developing energy-efficient measures with simultaneous maximum performance efficiency and minimum energy consumption [Dayarathna et al., 2016]. Zhao et al. [2016] observed that there is a rapidly increasing number of computing and data-intensive applications. Such applications have variable resource requirements based on user demands and submission times. As such, it is a difficult task to minimize the energy consumption while preserving the Quality of Service (QoS) in resource utilization and workload balance. In most cases, deployment of an energy-optimized solution invariably degrades the performance efficiency in terms of inefficient application assignment, resource provisioning and load balancing.

Our research is significant because it aims to resolve the major issue of energy inefficient application management by using a concept of profiles. Profiles provide prior knowledge of the run-time characteristics of the workload. They are implemented in the decision making stage of application assignment to VMs. The profiling technique makes the best use of the available VM resources and minimizes application waiting times in queues. Application arrival times

and resource requirements can be analysed, recorded and used in the next (future) instance of application arrival to execute prompt assignment decisions.

The research problems (Section 1.2) addressed in this thesis play a significant role in the daily operations of a data centre. As such the challenges arising from developing a profile-based application management scheme motivate the research. The challenges of addressing each of the research problems include:

1. Maintain a trade-off between energy-efficiency and performance;
2. Develop a scalable, energy- and performance-efficient framework for application assignment to VMs;
3. Model a real-time service quantitatively as a dynamic application management scheme using profiles; and
4. Consolidate applications without performance degradation whilst maintaining load balance.

The methodology chosen to implement profile-based assignment is a Genetic Algorithm (GA) based heuristic. For a typical medium-scale data centre, we assume there are 2,000 VMs and 5,000 applications. Exhaustive search for optimal application assignment to VM will take approximately

$$VMs^{applications} = 2000^{5000} = 10^{16500} \text{ compilations}$$

If we assume each compilation takes 1 nanosecond, then time taken to perform exhaustive search is

$$10^{16500} \text{ compilations} = 3.3 \times 10^{16483} \text{ years}$$

This is infeasible and confirms that application assignment to VM is an NP-hard problem with a very large solution space. As such a heuristic-based algorithm is required to solve the problem by reducing the solution space. This thesis uses a GA-based heuristic due to its ability to give a feasible solution on termination of the algorithm at any time. For example there may be a scenario where the assignment solution is required in a short amount of time (on interruption)

or another scenario where assignment solution is obtained on the natural (without interruption) termination of the algorithm. GA is well suited to both these scenarios. The GA parameters also enable control of the solution search space, avoids becoming trapped in a local optima and converges towards a global optima [Bajpai and Kumar, 2010].

This thesis introduces the concept and building of profiles for application assignment to VM. A profile-based application assignment framework that addresses all the challenges discussed above is presented. To the best of our knowledge, an application assignment strategy built on profiles has not been found in the literature. Another factor distinguishing the work presented in this thesis is the three-tiered energy management framework. The framework incorporates profile-based application assignment method to a First-Fit Decreasing (FFD) based VM placement policy. This provides a complete working solution to energy-efficient profile-based application management under QoS constraints. The following chapters of this thesis will discuss the methodology of profile-based application management in more detail.

1.4 Main Contributions of this Research

This thesis has made main contributions in four aspects in response to the four research problems. The contributions are:

Contribution 1 to the Profiling Problem: A systematic approach for profile building. Profiles are used to predict future workload and plan application assignment to virtual machine (VM) in advance of actual application arrival. Application, VM and server profiles are simulated for testing and later employ real data centre workload logs. Experimental evaluation demonstrates that the profiling approach is 22% more energy-efficient than the commonly used (benchmark) general and workload history approach to application management. Profiles have previously been used for pattern recognition, performance and behavioural analysis of scheduling strategies. However, to the best of our knowledge, profiles have yet to be used in the decision making stage of application assignment to VMs.

Contribution 2 to the Static Assignment Problem: A profile-based static application assignment framework. As the problem size of assigning applications to VMs is large, a genetic algorithm (GA) based heuristic is required to solve the NP-hard constrained optimization problem. The GA is modified to improve solutions by designing a Repairing Genetic

Algorithm (RGA) to implement the profile-based static application assignment framework. RGA has two main components: 1) Longest Cloudlet Fastest Processor (LCFP) generated initial population for faster convergence and minimized VM makespan (completion time of all applications on the VM); 2) Infeasible-solution Repairing Procedure (IRP) that convert infeasible solutions that violate resource usage constraints to feasible solutions by re-assigning applications from a violated VM host to other VMs until the violations are null and the solution fitness is satisfactory. The RGA is implemented with the assumption of a known (relatively consistent) workload. It is experimentally evaluated at various stages with other application assignment algorithms and proves to be energy and resource efficient under the investigated scenarios.

Contribution 3 to the Dynamic Assignment Problem: A profile-based dynamic application assignment framework. The RGA is implemented with varying (dynamic) workload. Future workload is predicted using profiles and application assignment to VMs are planned in advance, such that real-time applications are dynamically assigned to VMs. The problem is modelled after a real data centre and problem size is larger. Finishing time of applications are estimated using profiles to satisfy application deadline constraints and minimize waiting time to application assignment. The dynamic approach exhibits robustness by addressing infrequent scenarios such as new/random applications or failed/deactivated VMs. Incorporating First-Fit Decreasing (FFD) VM placement policy with profile-based dynamic application assignment demonstrates actual energy savings.

Contribution 4 to the Consolidation Problem: A profile-based application assignment consolidation approach. Consolidation requires the number of active VMs to be scaled down. This is implemented through a local search optimization (LSO) heuristic that empties out and shuts down under-utilized VMs by re-assigning applications to other VMs. This ensures VM resource-efficiency, further reduces server energy and maintains workload balance. The consolidation approach completes the final solution of energy-efficient profile-based application management in data centres of this thesis. Experimental analysis of the three-tiered energy management system using profiles demonstrates significant energy savings.

1.5 Thesis Organization

The thesis is organized around our four research questions and their solutions as the main contributions. It begins with this introductory Chapter. This is followed by a comprehensive literature review in Chapter 2. Chapters 3 through 6 present our main contributions, each solves a research question. Finally, Chapter 7 concludes the thesis and discusses future work.

The core chapters of this thesis are structured as seen in Figure 1.3. Chapter 3 solves the Profiling Problem and discusses building of profiles as Contribution 1. The Static Assignment Problem discussed in Chapter 4 develops a Repairing Genetic Algorithm as Contribution 2. Chapter 5 solves the Dynamic Assignment Problem for real-time applications to make Contribution 3. Contribution 4, application consolidation presented in Chapter 6 solves the Consolidation Problem to form a final solution.

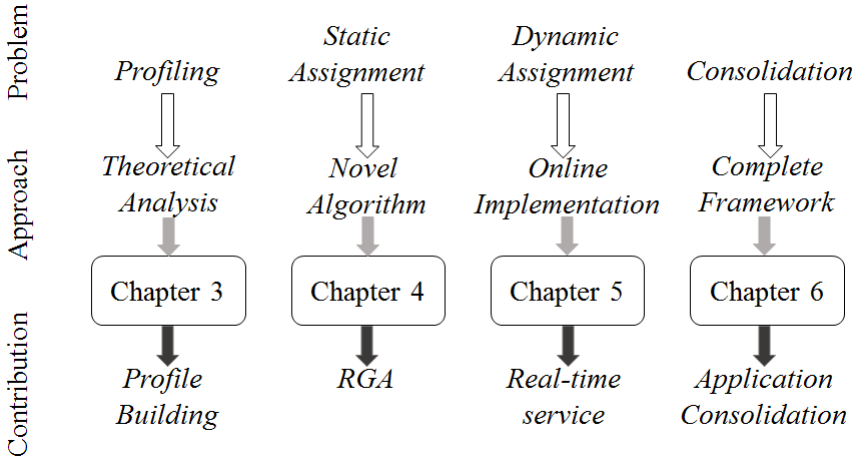


Figure 1.3: Thesis Organization.

1.6 Research Articles from PhD Research

Six research articles have been derived from the PhD research presented in this thesis. Two of them are presented in top-tier conferences. Three are under review in three journals. And a further one is in preparation. These articles are listed below:

1. Vasudevan, M., Tian, Y.-C., Tang, M. and Kozan, E. Profiling an application assignment approach for green data centers. In *Proceedings of the IEEE 40th Annual Conference of the Industrial Electronics Society*, IEEE, Dallas, TX, USA, Oct 29 - Nov 1, 2014, pages

5400-5406. (ERA Rank: A)

Relevant to Chapter 3, address Profiling Problem to make Contribution 1.

2. Vasudevan, M., Tian, Y.-C., Tang, M., Kozan, E., and Gao, J. Using genetic algorithm in profile-based assignment of applications to virtual machines for greener data centers. In *Proceedings of the 22nd International Conference on Neural Information Processing, Part II, Lecture Notes in Computer Science*, Springer International Publishing, Istanbul, Turkey, Nov 9-12, 2015, pages 182-189. (CORE Rank: A)

   Relevant to Chapter 4, address Static Assignment Problem to make Contribution 2.

3. Vasudevan, M., Tian, Y.-C., Tang, M. and Kozan, E. Profile-based application assignment for greener and more energy-efficient data centers. Future Generation Computer Systems. In Press, accepted on 30 June 2016 (ERA Rank: A)

   Relevant to Chapter 3, address Profiling Problem to make Contribution 1.

4. Vasudevan, M., Tian, Y.-C., Tang, M., Kozan, E. and Zhang, X. Repairing genetic algorithm for profile-based application assignment for greener data centers. Submitted to journal, Applied Soft Computing.

   Relevant to Chapter 4, address Static Assignment Problem to make Contribution 2.

5. Vasudevan, M., Tian, Y.-C., Tang, M. and Kozan, E. Profile-based dynamic application assignment for greener data centers. Submitted to The Journal of Supercomputing.

   Relevant to Chapter 5, address Dynamic Assignment Problem to make Contribution 3.

6. Vasudevan, M., Tian, Y.-C., Tang, M. and Kozan, E. Application consolidation of profile-based application assignment for greener data centers. In preparation of submission to journal.

   Relevant to Chapter 6, address Consolidation Problem to make Contribution 4.

Chapter 2

Literature Review

Currently, energy-efficiency is one of the most important issues in data centre management [Maza, 2016]. Traditionally, the evolution of computing systems is marked by enhancement of performance as per consumer demands. However, the Internet Age, has brought forward a more pressing concern in the form of rising energy consumption, electricity costs and CO_2 footprints. The work conducted in this research targets energy optimization of data centres. This Chapter classifies and analyses the energy-efficient data centre approaches designed to date. The Chapter concludes with a discussion on the research motivations following technical gaps in the reviewed literature.

Data centres and cloud systems have become essential to modern society. They cater to the explosive growth of connected users and networked devices. The unstable growth in data is due to the proliferation of smart devices, IoT, big data, social media, high performance computing and data intensive applications [TechNavio, 2015]. According to Evans [2011], there are more internet connected devices than people on the Earth. The Internet of Things (IoT) is currently marked at 22.9 billion connected devices as of year 2016. It is predicted to expand close to 50.1 billion connected devices by year 2020 [Statista, 2016].

Numerous research works have been conducted on developing energy-efficient measures for data centres. According to Beloglazov et al. [2011], significant changes in energy consumption can be effected by hardware efficiency, energy-aware resource management and efficiency of applications. Bashroush et al. [2016] further claims that software architects have been faced with challenges regarding energy-efficient design decisions that compromise user experience, reliability and performance. This issue is more serious when considering large-scale distributed

systems such as data centres.

To address this challenge, this thesis introduces the concept of utilizing profiles, which provide prior knowledge of the run-time characteristics of the workload, as an energy- and performance-efficient application management approach. The primary goal of this research is to design a profile-based application management framework. A Repairing Genetic Algorithm (RGA) is designed towards this objective. The RGA is incorporated with a three-tiered energy management system to implement three strategies: static assignment, dynamic assignment and application consolidation.

This Chapter reviews energy optimization strategies that are directly related to our research. The following literature have been instrumental in developing our research theory and methodology. It is organized as follows. Section 2.1 discusses general energy optimization strategies in data centre. Each of the following sections review literature directly related to the four Research Problems discussed in Section 1.2 of Chapter 1. Section 2.2 addresses the Profiling Problem and presents works using profiles for behavioural or performance analysis. Section 2.3 and Section 2.4 addresses the Static and Dynamic Assignment Problem respectively and reviews static and dynamic energy-efficient application assignment techniques. Section 2.5 addresses the Consolidation Problem and discusses related works. The Chapter is concluded with a discussion on the research motivations in Section 2.6.

2.1 Data Centre Energy Optimization Strategies

A data centre is composed of IT equipment such as servers, storage hardware, routers, switches, racks and cables, and lighting, air movement, power and cooling infrastructure [Data Center Huddle, 2016]. Figure 2.1 presents the approximate energy distribution amongst the data centre components and further energy distribution of the data centre IT equipments [Barroso et al., 2013, Melis, 2013]. The IT equipment draws half the energy (50%) required to power and maintain a data centre. As a result, research and development on the energy optimization of data centres focus on minimizing the energy draw of IT equipment. The servers draw the majority of the IT energy consumption through CPU (42%), memory (12%) and disk storage (14%). Therefore, energy-efficient strategies through server or virtual machine (VM) resource

allocation is a popular research field. Other areas of research into energy-efficient data centres include utilization of renewable energy, virtualization, cloud computing and management strategies. Each of these strategies is briefly discussed below.

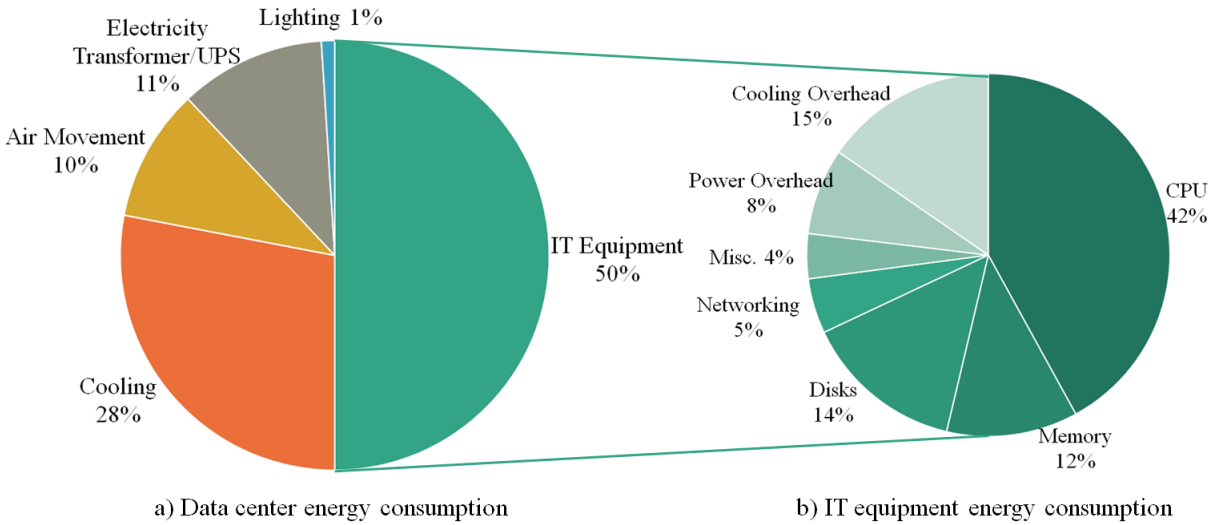


Figure 2.1: Analysis of data centre energy consumption [Melis, 2013] and energy distribution of IT equipment [Barroso et al., 2013].

2.1.1 Classification of Green Strategies

Renewable Energy. The optimum use of renewable energy is an ongoing research problem. The variability of such energy makes it a challenge to be used in place of uninterruptable power supply (UPS). Many multinational companies aim to significantly reduce energy consumption and carbon emissions by using renewable energy as an alternate power source. British Telecommunications Plc. utilize wind farm generated energy to successfully reduce carbon emissions by 80% [RE100, 2016]. All of Apple's data centres are powered by 100% renewable energy such as solar power [Apple, 2016]. Both Google [2016] and Facebook [2016] currently runs on 35% renewable energy individually. They aim to follow Apple's example of a fully renewable energy powered data centre by addressing challenges associated with massive data operations.

Virtualization. One of the most energy-efficient and cost-effective methods for data centre consolidation is virtualization [Bohrer et al., 2002]. This technology is necessitated by applications utilizing only a fraction between 11% and 50% of the server resources. This leaves the remaining resources redundant. The concept of virtualization is to divide a single physical component such as a server into one or more VMs. The VMs share the server resources between

them. Therefore, virtualization does not require additional hardware. It also simplifies server management and augments efficient resource utilization. Virtualization allows for either regular or live migration of VMs in order to satisfy the load balancing policy (workload distributed optimally across servers) and to ensure maximum resource utilization. Even though the virtualization technology has been around for many years, it is only now that its importance to the environment is becoming clear. Pretorius et al. [2010] conducted a critical comparison between physical server data centre infrastructure and virtualized data centre infrastructure. Both the infrastructures are modelled and analysed in terms of carbon emissions and energy savings. The results clearly demonstrate that virtualization can reduce the carbon emissions by 30%.

Cloud Computing. The service of renting out data centre resources to users around the world is known as cloud computing. The services rented out include websites, social media, applications and storage. Major companies like Google, IBM, Amazon and Facebook individually own data centres to support their applications, resources, memory and file storage. A Service Level Agreement (SLA) is created based on user requirements. The cloud providers must adhere to this in order to provide their customers with reliable and efficient services. Cloud computing is one of the most innovative technologies that can efficiently share data centre resources amongst multiple users. It controls the need for more physical server installations at data centres, thereby restraining higher energy consumption.

Management Strategies. Management strategies include assignment and scheduling strategies applied to applications, VMs and resource. (In the reviewed literature the term task can be applied to application.) Assignment strategies involve assigning applications to data centre resources. Some of the factors considered in assignment schemes include application runtime, server workloads, resource requirements or availability, energy consumption and performance efficiency. Resource provisioning amongst VMs must be carried out in such a way that the performance level must be maintained while minimizing the energy consumption [Hwang and Pedram, 2016, Sun et al., 2015]. Dynamic scheduling can be classified as real-time scheduling and batch scheduling [Hao and Liu, 2015, Mathew et al., 2014]. In real-time scheduling, applications are allocated to servers as soon as they arrive. Minimum Completion Time (MCT) [Santos et al., 2014], Minimum Execution Time (MET) [Kim et al., 2014] heuristics, Switching Algorithm (SA) heuristics and k-percent best (KPB) heuristics are some real-time scheduling heuristics. Batch scheduling involves collection of applications which is then assigned to servers at particular intervals of time. Some examples of batch scheduling heuristics are Min-Min

heuristics, Max-Min heuristics and sufferage heuristics.

2.1.2 Green IT Management Strategies

Popular energy-efficient data centre management strategies include task scheduling, workload prediction and application assignment. As such, a brief discussion on the work done in task scheduling and workload prediction methods are given below. The rest of this Chapter delves further into the work on energy-efficient application assignment to date.

Efficient scheduling of tasks or applications in computing systems ensures processing of service requests within deadlines. Li et al. [2012b] consider a cyber physical system made up of multiple wireless sensor networks (WSN). A polynomial three phase task scheduling heuristic called HTPTS is implemented. In phase one the application deadline is divided into sub-deadlines pertaining to the various tasks that make up the application. Phase two is an adaptive task graph partitioning algorithm that allocates tasks to WSNs with the objectives of minimizing energy consumption and maintaining workload balance. The third phase is a scheduling algorithm to support the objectives of phase two. A common constrain in scheduling heuristics such as this, is that application deadlines are unknown prior to execution. In our project, the profiles enable awareness of application deadlines prior to assigning them to VMs.

An optimal application assignment accounts for variations in application resource demands due to unforeseen events. These events include suspended tasks, high workload conditions or change in system demands. Some papers [Campbell et al., 2008, Huang et al., 2011, Kontogiannis, 2005] have developed adaptive application assignment strategies for real-time demands. In [Huang et al., 2011], the authors design an initial task schedule which adapts to new system conditions or task demands through online periodic adjustments. This technique not only contributes to the lifetime reliability improvement but also minimizes energy consumption. Another technique uses Coloured Petrie Nets for adapting the task model to varying task demands [Kontogiannis, 2005]. The simulation tool examines the effect of task demands, workload variations and work organization schemes on system operation and human reliability. This in turn helps in identifying different permutations of task and workload assignment. Campbell et al. [2008] present an adaptive task allocation procedure. This method uses system history to make non-greedy task assignments. In other words, a host node will have the freedom to choose not to host a particular task if it is deemed not suitable by certain standards pertaining

to individual host nodes.

There are many techniques for predicting the workload of a CPU. Some of the popular methods are adaptive filtering [Sinha and Chandrakasan, 2001] and Kalman filtering [Bang et al., 2009]. Hong et al. [2012] presents a linear prediction model to predict the workload of WSN. Initially, the queued tasks are scheduled for deployment at the microprocessor. Once the task arrives at the processor nodes, the CPU workload is computed over a specified observation frame. This CPU workload data for individual tasks are used to build a workload history. The workload history is then used to estimate the future workload. On the basis of this predicted workload estimate, Dynamic Voltage Scaling (DVS) is applied to the microprocessor to adjust its current workload. This cycle is similar to that of a feedback control loop.

Nagothu et al. [2010] claims that up to 80% of the energy reductions in cloud computing can be achieved with the help of adaptive workload prediction and optimal task allocation. Future workload prediction using advanced signal processing analysis enables identification of unloaded microprocessors. These processors are then placed in low power sleep modes to conserve energy. Doulamis et al. [2004] implements workload prediction employing both fuzzy classification and neural network model to increase prediction accuracy whereas, Yang et al. [2005] implements dynamic prediction using a self-adaptive load prediction model and mobile agent technology.

Workload prediction for task allocation can be improved through modification of standard heuristics like Min-Min. Xiao et al. [2009] proposes prediction based Min-Min heuristic (P-MinMin) derived from the Min-Min heuristic and the application of latency prediction. A weighting parameter determines the optimal task allocation solution by maintaining a trade-off between latency and energy consumption. The efficiency of the heuristic is determined with the help of Deadline Missing Ratio (DMR), which is the ratio between the number of simulation runs with latency exceeding their deadlines to the total number of simulation runs. According to the simulation results, the DMR of P-MinMin is lower and therefore more efficient than that of the Min-Min heuristic.

In the papers discussed above, workload prediction is carried out on the basis of historical workload data. This technique can be improved by using application, VM and server profiles for building a more stable and reliable log of resource demands and availability as demonstrated in the following chapters of this thesis. Incoming applications can be anticipated using

prior knowledge of application submission times in profiles. This facilitates faster energy and performance aware assignment decisions.

2.2 Profiles in Energy-Aware Data Centres

Profiles have been used as a means of performance and behavioural analysis in data centre management. Some of the most prominent work in energy-aware data centre strategies using profiles are discussed below. In our research, profiles provide prior knowledge of the run-time characteristics of the data centre workload.

Static profiling technique is discussed in [Mars et al., 2011, 2012] for prediction of performance degradation in relation to multiple application assignment to a single machine. The method of Bubble-Up and Bubble-Flux [Yang et al., 2013] is developed to accurately predict the performance degradation incurred on allocating multiple workloads to servers for maximum utilization. Bubble-Up maintains a trade-off between machine utilization and Quality-of-Service (QoS) degradation by setting a degradation threshold for each application. However, it has limitations such as the requirement of workload knowledge, prediction inflexibility in terms of load changes, and the incapability of predicting more than two co-running applications. Those limitations are overcome with the Bubble-Flux management strategy. The Bubble-Flux accurately manages the QoS to provide maximum utilization of servers. Servers are monitored to observe shared resource fluctuations in real-time to predict the effect on the QoS of latency-sensitive applications.

Profiles are used to study a number of variables such as energy consumption and workload of data centres and cloud systems. Rethinagiri et al. [2015], developed ParaDIME, Parallel Distributed Infrastructure for Minimization of Energy for data centres. This work included power infrastructure and computing measures addressed at the software and hardware levels of a data centre. Static energy profiles are used to indicate applications that can be run in low energy mode. Subsequently, such applications are allocated low-performance VMs and limited parallelization. Chen et al. [2013] profiled energy consumption with respect to computation, data and communication intensive tasks to build an energy consumption model. The work is extended in [Chen et al., 2014a] to present StressCloud for profiling the energy consumption and performance of cloud systems. The tool successfully profiled varying task workloads and

resource allocation. Chen et al. [2014b] uses online profiling to collect workload information of tasks. A workload-aware frequency adjuster tunes the core frequencies using this information. The tasks are allocated to the cores by the preference-based scheduler. This method uses profiles to adjust frequencies prior to allocation. However, in our method, the profiles are directly used to make actual allocation decisions.

Varying resource demands are commonly dealt with by migrating VMs. However, Do et al. [2011] investigate the relationship between resource demands and application performance metrics. They present an application profiling technique using Canonical Correlation Analysis (CCA) method. The CCA analyses the performance efficiency of the applications in term of their resource usage and builds application profiles. These profiles are then used to build a performance prediction model. In [Baxter and Patel, 1992], the authors propose an application migration algorithm based on the profiles created from the dynamic behaviour of the static application allocation. These profiles are then used to determine the migration destination of the applications. Similarly, our profile-based method implements a solution repairing and application consolidation procedure that enables performance-efficient migration of applications.

The above works have made use of profiling for analysis of energy consumption and workload in data centres and cloud systems. However, to the best of our knowledge, profiling as a decision making method for the assignment of applications to VMs has not been found in the literature.

2.2.1 Application Assignment Models

Various mathematical optimization methods have been developed for modelling application assignment. Implementing an energy aware application management framework begins with building an optimization model. Many papers employ mathematical optimization techniques such as linear programming and non-linear programming for application assignment. Some of the most relevant papers to this research is reviewed here.

One of the optimization priorities, commonly approached by researchers is the assignment of deadline-driven applications. To accurately surmise the feasibility region of applications with deadlines involves complex calculations. Zeng and Di Natale [2013] approximate the feasibility region and employs mixed integer linear programming method to identify sub-optimal solutions for application assignment. Oxley et al. [2015] models the execution times of applications using

Gaussian distribution. This is used to calculate the probability of a deadline violation which is used as a constraint for resource allocation. Hatime et al. [2013] uses branch and bound deadline-driven allocation solution through two techniques: best-node and depth-first. The best-node technique executes search level by level from top to bottom. Whereas, the depth-first technique executes search branch by branch moving towards the bottom most node. This thesis uses profiles with prior knowledge of application deadlines, recorded from previous instances, to address deadline constraints. This avoids complex calculations.

Prompt execution of tasks or applications is crucial to the performance of a scheduling algorithm. Liu et al. [2012] proposes an integer programming optimization model for task scheduling with energy consumption and task execution time as problem constraints similar to [Fanti et al., 2012], where a distributed algorithm aimed at minimizing task allocation costs is presented. A trust model based task scheduling algorithm is proposed by Xu and Qu [2011], that not only considers the transmission time of the data files but also the trust level of data file host nodes. The Min-Min task scheduling algorithm is specifically designed for data intensive tasks. Data intensive tasks require more than one data file stored in various nodes, which may need to be accessed simultaneously for the successful execution of the task. The algorithm aims to overcome the high probability of data loss and error usually associated with data intensive tasks. However, it compromises on the task execution time. The work done in this thesis will estimate the execution time of applications to ensure successful execution of tasks.

The application of a statistically based task allocation model in a dynamic computing system environment is a viable approach. Singh and Kumar [2014] propose HABACO, a hybrid optimization algorithm combining ant colony and artificial bee optimization heuristics. The algorithm maintains server workload balance by live migration of VMs and applications. As live migration of VMs acquires overhead for transferring memory/instructions and risks crashing applications, Benbrahim et al. [2014] proposes a dynamic resource allocation model with reduced VM migrations. From reviewing the above papers, adaption of a task model according to real-time parameter variations can be improved by building dynamic profiles which updates periodically. Also, workload balance can be maintained by application consolidation which migrates applications from one VM to another instead of live migration of VMs.

2.3 Static Application Assignment

Application assignment involves assigning computing tasks or applications to data centre resources such as CPU and memory. The factors considered in assignment schemes include application runtime, server workload, resource requirements or availability, energy consumption and performance efficiency. Thus, a key issue is how to formulate and solve the application assignment problem subject to various constraints.

Early attempts to discuss the classical assignment problem include [Ewashko and Dudding, 1971, Machol, 1970, Votaw and Orden, 1952] and [Lo, 1988]. The most common approach used to solving these problems is to model the problem in linear programming and then solve the problem with the Hungarian algorithm [Kuhn, 1955]. However, the Hungarian algorithm does not perform well with large/increasing data sets or problem size i.e. it has low scalability. Thus Hungarian algorithm is not suitable for large-scale application assignment problems as those in modern data centres. Later, the authors of [Caron et al., 1999] discuss the assignment problem of nurses qualified to carry out jobs in different units of a hospital with the side constraints of seniority and job priority. They took into consideration absentee staff and unexpected work overload. Since then, object-oriented assignment solutions has been applied to a number of real-world problems including data centre application management. Some works on applications resource provisioning and the use of evolutionary algorithms in assignment are discussed below.

2.3.1 Static Resource Provisioning

Resource provisioning ensures VM workload balance and allows switching-off machines to reduce energy consumption. The EnergyFarm manager designed by Ricciardi et al. [2011] switches off computational resources at periods of low workloads determined by a service-demand matching algorithm and server job aggregation capabilities to maintain system performance efficiency. This data centre energy manager is developed on the concept of an emergency shutdown software, like PowerFarm developed by Doria et al. [2010] in case of power loss, fire, flood or any natural disasters, which is inherent in every data centre. EnergyFarm data centre manager is developed by modifying the PowerFarm software such that server workloads and energy consumption are monitored by the manager in order to turn off idle servers. The EnergyFarm manager achieves considerable resource allocation efficiency amid 20% to 68%.

A similar application and platform aware resource allocation framework for consolidated server systems is proposed by Tembey et al. [2014]. The framework targeted multicore platforms and scale-up server systems. Song et al. [2014a] used online bin packing for resource provisioning while reducing number of servers and meeting application demands.

A popular method of provisioning resources involve prioritizing applications. High priority applications, whose successful execution is more important than that of the low priority applications, are given first choice of resources. Some papers like [Song et al., 2008, 2009] have proposed the concept of ensuring high performance of critical applications by compromising on the performance of low priority applications when there is a high demand for resources at a given time. Song et al. [2009] proposes a multi-tiered resource scheduling scheme consisting of three schedulers, each at the application, local and global levels. The application level scheduler assigns applications to VMs. The local level scheduler, [Song et al., 2008] monitors the application priority levels and the resource utilization levels in order to control the resource distribution amongst VMs. The global level scheduler acts as a controller that monitors the overall VM resource flow. Although the scheme improves the overall QoS of the system, the main constraint is that each of the applications have a pre-defined priority.

Weights can be used to set priorities for applications prior to resource allocation. Ergu et al. [2013] proposes a rank-based task resource allocation model. The model weights tasks according to a reciprocal pairwise comparison matrix and the analytical hierarchy process. Sheikh and Khan [2005] generates timetables on a daily basis such that the resource allocation of teachers to courses is optimal. This allows the best available teachers to teach their relevant courses in the most optimal time slot. The authors use weighting based on the priority of courses chosen by teachers to determine objective values. The problem that is addressed in this paper is similar to our own research where the applications are assigned to the most suitable VMs. Weighting, which is in the form of an energy cost matrix, derived from analysing both the profiles of applications and VMs to minimize the objective function is used. In other words, the weighting will ensure that only the best possible assignment will take place.

Application runtime is a performance metric in determining the efficiency of a resource provisioning framework. In order to increase resource efficiency, Chiang and Huang [2011] uses Task and Resource Allocation CONtrol (TRACON) framework. The TRACON framework comprises of three important aspects; the interference prediction model, the interference aware

scheduler and the task and resource monitor. Here, the interference term refers to the change in runtime and performance. The authors claim that TRACON achieves upto 50% improvement in application runtime. A different method using a speed scaling model for resource efficiency is proposed by Han and Cai [2013]. The model deploys applications to resources based on cooperative game theory.

Many existing resource provisioning solutions are deadline-driven and considers execution times to increase resource computational efficiency [Calheiros and Buyya, 2014, den Bossche et al., 2013, Moens et al., 2013]. Panda and Jana [2015] presents three task scheduling algorithms, each focused independently on completion time, median execution time and makespan for heterogeneous multi-cloud systems. Bhoi and Ramanuj [2013] develops an enhanced max-min application scheduling algorithm for cloud systems. The algorithm is contingent on the expected execution time rather than the more common runtime as a selection criteria. Fahim et al. [2014] estimates the finish time of tasks prior to allocation to VMs. The allocation objectives include minimizing the degree of imbalance of VMs and load balancing. In this thesis, we use a similar approach to estimate finish time of applications using profiles in order to satisfy deadline constraints whilst minimising energy consumption.

An issue with load-balanced resource allocation is the variation of runtime parameters that easily disrupts the initial optimal allocation. This renders the performance of the system unimpressive. Gertphol et al. [2002] overcome the need for dynamic reallocations in such scenarios by using a performance metric called Maximum Allowable Increase in Load (MAIL). The MAIL metric refers to the effectiveness of a resource allocation. This allows the resource allocation with the highest MAIL value to be carried out. Pahlavan et al. [2012] employs a variation aware chassis consolidation method for application assignment. According to the performance results, this approach minimizes the energy consumption of the data centre by addressing process variation through variability analysis of application assignment and chassis consolidation.

Other popular methods for resource provisioning include machine learning and resource allocation heuristics. Machine learning technology as a resource allocation method is effective in handling varying workload [Berral et al., 2011, Grehant and Demeure, 2009]. Yuan et al. [2012] makes use of the N:1 mapping visualization technology and reinforcement learning to propose an energy efficient scheduling algorithm that has been proven to minimize the energy

consumption of energy unaware data centres by 40% along with maximising the resource utilization. Energy-aware management of data centres for cloud computing is also investigated through heuristic resource allocation [Beloglazov et al., 2012]. León and Navarro [2013] builds a quantitative model to describe the problem of minimizing energy consumption for resource allocation in data centres. All the above studies utilize different methods to achieve energy savings, but none of them use the profiling concept on which our work in this thesis is based.

One of the technological gaps apparent while reviewing these papers is that estimating VM resource availability at different time periods has not been considered. Discerning the resource availability of a VM facilitates best possible assignment decisions. Accordingly, we build VM profiles that contain resource data at different time periods which is then used in assignment decisions.

2.3.2 Evolutionary Algorithm based Assignment

Implementation of evolutionary algorithms for application management is one of the most energy-efficient techniques to maintain a trade-off between energy consumption and performance [Kessaci et al., 2011]. Evolutionary algorithms such as genetic algorithm (GA) have been successfully applied for application scheduling and resource provisioning in data centres and cloud computing [Guzek et al., 2014].

Application scheduling in data centres is generally NP-complete. The main advantage of using GA for NP-complete allocation problems is to allow faster convergence to a global optima by searching the solution space in multiple directions [Bajpai and Kumar, 2010]. Wang et al. [2014] presents a multi-objective bi-level programming model based on MapReduce for application scheduling. The model considers server energy-performance association and network to adjust job locality. Solutions are derived using GA enhanced with newly designed encoding/decoding and local search operation. Pop et al. [2015] propose a reputation guided genetic scheduling algorithm for autonomous tasks inter-clouds environment. Sindhu and Mukherjee [2013] tests a GA-based application and VM scheduler for cloud systems with various initial populations generated from heuristics. Among various methods, the Longest Cloudlet Fastest Processor (LCFP) is shown to be the most efficient when considering a large number of processing nodes. Therefore, LCFP is incorporated into our designed Repairing Genetic Algorithm (RGA) for application assignment to VMs.

Resource provisioning using evolutionary algorithm allows for multiple objectives such as energy consumption, performance and scalability. Tao et al. [2014] uses a Pareto-front and solution-based hybrid GA for energy-efficient resource allocation in cloud systems. The model utilizes the crossover operator for multiple genes and a case library for initializations by identifying case similarities. The concept of case library is similar to that of profiles. While a case library considers only allocation information, profiles are more adaptable by considering individual component information. Sharma and Reddy [2015] combine dynamic voltage frequency scaling, bin packing and genetic algorithm to propose a hybrid energy-efficient approach for resource provisioning. An energy-efficient resource allocation approach using an open source GA framework called jMetal is designed by Portaluri et al. [2014]. The allocation objectives include optimizing task completion times whilst satisfying computational and networking task requirements. The method ensures scalability and performance efficiency for a large number of tasks. Our research uses application and VM profiles to solve a penalty-based GA for large problem sizes without compromising performance efficiency such as resource utilization efficiency.

It is worth mentioning that GA has also been used to solve VM placement problems, which have been proven to be NP-complete. Wu et al. [2012] minimise the energy consumption of servers and the communication network within the data centres using GA-based VM placement. The work is extended in [Tang and Pan, 2015] to significantly improve the energy and performance efficiency with an enhanced hybrid GA. A similar method of using a GA-based repairing heuristic algorithm for workflow scheduling in cloud systems is also discussed by Ghorbannia Delavar and Aryan [2014]. Considering the application management layer, our work in this thesis also develops an infeasible-solution repairing procedure and then incorporates the procedure distinctively into application assignment to VMs.

2.4 Dynamic Application Management

Dynamic application management schemes handle real-time workload and dynamically assign applications to resources or processors. This thesis implements dynamic application management by developing a Repairing Genetic Algorithm (RGA). Many dynamic schemes implement various methods for energy saving. Some works on the methods of dynamic resource provisioning, application scheduling and assignment are discussed below.

2.4.1 Dynamic Resource Provisioning

Prediction is a popular method of handling workload variation. Dynamic workload can be characterized by varying resource demands of incoming applications. Nguyen et al. [2013] addresses this issue with an elastic distributed resource scaling framework called AGILE. It is capable of handling dynamic workloads with minimum penalty in terms of resource constraints incurred. It uses online profiling to model the constraint violation rate and carry out wavelet-based prediction of resource demands. It further employs this prediction to handle variations in workloads. In comparison with online profiling, offline profiling is used in an overdriver framework to analyse the memory overload probability of VMs [Williams et al., 2011]. This thesis initially builds profiles offline and then maintains them online by updating information such as resource utilization, execution times and deadline to handle real-time workload.

Other methods for handling varying workload include stochastic optimization and machine learning. Stochastic optimization techniques are effective in dynamic resource provisioning as it allows for multiple objectives. Arroba et al. [2014] proposes an automatic multi-objective particle swarm optimization method for dynamic cloud energy optimization that considers both power consumption and server temperature to derive accurate server power model that can later be used to predict short-term power variations in data centres. Machine learning can be used to study workload behaviour and implement energy-efficient resource provisioning. Bahrpeyma et al. [2015] develops an adaptive controller for dynamic resource provisioning by using an Ink Drop Spread (IDS) method based on reinforcement learning. The controller nullified application rejection rate and minimized energy wastage. Wang and Su [2015] presents a dynamic hierarchical task resource allocation scheme. The tasks and nodes are classified into levels based on power and storage using fuzzy pattern recognition. Incoming tasks can only be hosted by nodes on the same level. The profiles used in this thesis offers a computationally simpler method of studying workload in preparation of application assignment.

Constrained and data-intensive applications require more computational- and interference-aware resource scheduling. Cress by Li et al. [2014] is a dynamic resource scheduling scheme for constrained applications. Cress attempts to maintain a trade-off between resource utilization and individual application performance. The scheme employs a conversion method to dynamically adjust soft and hard constraints for fluctuating workloads. PIASA by Sampaio et al. [2015] is a dynamic power and interference aware resource management mechanism. It is designed to

handle different types of data-intensive application workloads in dynamic cloud environments.

It is worth mentioning some works on dynamic resource scheduling applied at the VM layer. Buyya et al. [2010] proposes a dynamic resource scheduling algorithm and software technology for the optimum energy management of data centres. Optimum VM-resource allocation is carried out in two stages. The first stage of the VM allocation is deployed using a modified Best Fit Decreasing (BFD) algorithm wherein the VMs are arranged in a decreasing order of their utilization levels and allocated to the host server that will provide only the minimal increase in energy consumption with regard to the allocation. The second stage consists of optimizing the current allocation of VMs wherein the VMs to be migrated are selected and according to the Modified Best Fit Decreasing (MBFD) algorithm are placed on host servers. The selection of the VMs to be migrated are based on the upper and lower utilization levels of the host servers such that if the utilization of the CPU exceeds or falls below the respective thresholds, the related VMs will be live migrated. The performance efficiency results of the cloud computing model, obtained with the help of the CloudSim toolkit, shows a significant gain on the basis of response time and cost saving. CloudSim as described by Calheiros et al. [2011] is a modelling, simulation and experimentation toolkit that can be used to model clouds, data centres and VMs. It also allows for the implementation of resource allocation policies which can then be studied, tested and modified.

2.4.2 Dynamic Application Scheduling and Assignment

Combining application assignment with another power conservation technique provides more energy-efficiency. Generally an assignment heuristic is used in conjunction with frequency scaling. Verma et al. [2008] presents the pMapper placement framework for the dynamic assignment of applications in virtualized systems. The pMapper framework consists of performance, power and migration manager along with an arbitrator. The performance manager deals with QoS and SLA requirements and accordingly re-sizes the VMs. The power manager monitors and controls the energy consumption by implementing Dynamic Voltage and Frequency Scaling (DVFS) as discussed by Sueur and Heiser [2010] or CPU throttling described by Stoess et al. [2007]. The migration manager is in charge of the live migration of VMs. The arbitrator plays the part of a central controller which monitors the overall system and adjustments to performance and power measures when necessary. This energy aware application placement

controller framework minimizes both the energy and migration costs of the resource provisioning and maintains the QoS. Zhang and Guo [2014] presents a static and dynamic sporadic task low power scheduling algorithm for sporadic real-time tasks with the objective of minimizing energy consumption. The algorithms use a combination of dynamic voltage scaling and power management to adjust task delay and processing speed while maintaining deadline constraints. In our research, we pair application assignment with VM placement to gain more energy savings and consolidate the system.

Applications can be classified according to required server resources prior to assignment. Zapater et al. [2012] proposes a technique for energy aware task scheduling and workload distribution for green data centres. The workload entering the data centre goes through a resource manager which characterizes each task for every system resources. Then two optimization approaches: static off-line data centre optimization and dynamic run-time allocation are applied. The static off-line data centre optimization identifies the most suitable combination of resources to tasks to build a heterogeneous data centre which has an acceptable performance and energy efficiency. The dynamic run-time allocation optimization approach will then allocate tasks or workload to the computing resources during run-time. However, this approach is only effective in data centres with periodic workloads.

Dynamic assignment of applications at times leads to resource conflicts which arise due to the increase in the application runtime. Shi et al. [2011] presents an application placement framework (EAPAC) for data-intensive applications. The framework overcomes resource conflicts by ensuring that a mixture of applications with different resource requests are assigned to individual servers. The EAPAC consists of an application level load balancer and an application server manager. The load balancer assigns applications to server hosts while the server manager monitors the resource provisioning amongst servers. The EAPAC is claimed to be able to improve the task waiting time by 4 times as compared to Tang's method presented in [Tang et al., 2007] for dynamic application placement in data centres. However, the EAPAC is intended for deployment in non-virtualized environments. In contrast the coupled application placement framework, CPA by Korupolu et al. [2009] is deployed in virtualized data centres. In our work, resource conflicts are avoided using automatically updating profiles.

Dynamic application scheduling can be energy-efficiently implemented using prediction. Kong et al. [2011] tackles the challenge of optimal task scheduling on two levels. The first

stage involves building a fuzzy prediction model of the workload and the second stage involves implementation of the proposed on-line dynamic task Scheduling Algorithm based on Load-balance and Availability Fuzzy prediction (SALAF). The SALAF algorithm aims to schedule tasks by satisfying the task availability and response time requirements. According to the performance results, the SALAF algorithm efficiently enhances the total availability of virtualized data centres while sustaining good responsiveness. The initial optimal allocation of a task on a parallel VM could prove to be ineffective due to the workload variation in individual server hosts. Therefore, the workload is dynamically redistributed to improve system performance. SGEESS, Smart Green Energy-Efficient Scheduling Strategy is developed by Lei et al. [2015]. SGEESS considers renewable energy supply prediction and dynamic electricity price for real-time scheduling of application.

One of the limitations of the above papers is that whenever an application enters the data centre, it undergoes placement processing regardless of the frequency of its execution. In order to avoid the redundancy in application processing, we employ the profiling technique that consists of data related to applications such as their resource requirements and duration that saves on application processing time.

2.5 Consolidation Strategies

The basic concept of consolidation is to reduce the number of active servers or VMs of a data centre, thereby minimizing energy consumption while maintaining load balance. Consolidation targets underutilized machines (approximately below 20%) that in their idle states consume 50% of energy consumed at maximum utilization (approximately 80%) [Bohrer et al., 2002]. Consolidation can be implemented by various means including VM, resource, workload or application consolidation. The strategy is one of the most efficient methods of saving energy. Wang and Wang [2014] presents a performance controlled power optimization method for multi-tier applications in virtualized data centres. The method reassigns CPU resources to handle variations in workload. This is integrated with DVFS to provide server consolidation.

Artificial Intelligence (AI) is one of the most popular research areas in server consolidation. Swarm intelligence is a form of AI and can represent a population of machines. Pop et al. [2012] discusses a swarm-inspired data centre consolidation framework that maintains energy

and workload performance efficiency. The framework is based on behaviour of birds flying in V-formation. The leading birds represent active fully-loaded servers that are potential candidates to be switched-off after workload execution. The middle of the V-formation is attributed to servers with low workload. These servers are the primary choice for new incoming applications. The end of the V-formation represents servers that are idle. These servers are candidates for active states and incoming applications. Intelligent agents which are a part of servers collaborate to arrive at decisions regarding resource provisioning, dynamic power management (on/off states) and server placement in the V-formation.

Server consolidation in virtualized data centres can be implemented by VM migrations. Hermenier et al. [2013] proposes BtrPlace, a flexible consolidation manager for multi-tier applications. BtrPlace is created using constraint programming. There is provision to further expand placement constraints as per custom demands of consolidation manager. BtrPlace reassigns VMs to satisfy data centre viable configuration, such that the VM consolidation satisfies all specified constraints. Most recently, Ye et al. [2015] presents an energy-efficient server consolidation framework. It reduces the number of active physical servers and VM migrations in data centres whilst maintaining workload performance. Profiles are used to analyse performance losses of workloads during co-location and migration of VMs.

Queueing theory in application assignment can be used to predict waiting times to consolidate resource. Chen et al. [2012] proposes a queueing-theory based tool for multi-core systems. The proposed tool predicts application scalability and resource demands; and provides consolidation suggestions based on performance and resource usage. Desnoyers et al. [2012] presents Modellus, an automated modeling of complex web applications in data centres. It uses queueing theory, data mining and machine learning strategies to predict application resource usage and dependencies between collaborating applications. The presented approach provides high accuracy prediction of application behaviour. Li et al. [2012a] presents a resource consolidation algorithm, Online Coloring First-Fit (OCFF), for heterogeneous applications. The OCFF packs incoming applications into the minimum number of servers.

Workload consolidation can be resource-effectively carried out with machine learning. The authors, Singh and Rao [2012] implement server workload prediction using an online ensemble learning approach in large data centres for workload consolidation. Workload characterization is carried out as a time-series based on the utilization of CPU. Subsequently, ensemble learning

algorithm is developed through four phases. The first phase collects historical workload data to create base workload learners. The second phase applies Weighted Majority (WM) algorithm to form the final prediction. In the third phase, the weights of the base workload learners are updated in terms of the accuracy of their individual predictions. The fourth and final phase involves the application of dynamic weighted algorithm. According to evaluation results, the ensemble learning algorithm achieves high accuracy in terms of server prediction. Workload consolidation can also be carried out by prioritizing processors. Liu et al. [2013] targets parallel applications and proposes a priority-based workload consolidation method. VMs are classified according to high-low CPU priority tiers. Parallel applications are scheduled to a tier as per execution times to increase resource utilization.

Implementation of consolidation often leads to resource contention. Han et al. [2016] studies multiple distributed parallel applications for interference effects on multiple cluster nodes. The authors then present a static profile-based model on interference and heterogeneity. The profiles collect execution times of the distributed applications to conduct a sensitivity analysis. Similarly, Prekas et al. [2015] addresses applications with low tail latency assigned to dedicated servers. The strategy presented detects load changes and re-assigns flow groups of latency critical applications without packet dropping. Pareto frontier static configurations are used for resource provisioning, thereby consolidating workload. In our research, application consolidation using a local optimization heuristic is implemented to reduce the number of active VMs.

2.6 Technological Gaps and Motivation

Our literature review has helped us gain a clearer understanding of the existing strategies and the work done on the energy optimization of data centres. After careful study of the various energy aware measures, we have realized that developing an energy- and performance-efficient application management of a data centre is a significant problem yet to be solved. After study of the numerous energy-efficient measures for application assignment, the following technological gaps are identified, which motivate the research of this thesis:

- On entering the data centre, every application undergoes placement processing regardless of the frequency of its execution. The profile-based assignment strategy presented in this

thesis, collects and reuses data such as resource demands and availability. This reduces the waiting time for processing applications and allows the assignment algorithm to make better energy and performance efficient decisions.

- Profiling has been previously considered for resource consumption pattern identification, behavioural and performance analysis. This thesis presents an approach of using profiles in the decision making stage of assigning applications to VMs for data centres.

- Application finish time is unknown prior to scheduling. In our research, the profiles enable us to estimate the finish times of applications prior to assigning them to VMs.

Application assignment in many existing data centres use generic algorithms such as First-Fit, Best-Fit or random assignment to VMs. Though these approaches maintain assignment performance such as speedy execution times and resource utilization, they fail to deliver on energy-efficiency. The profile-based approach presented in this thesis overcomes this problem by capturing the run-time characteristics of applications, VMs and servers over a period of time. Profiles use this information to assign application to VMs energy-efficiently.

In this thesis, we have chosen an improved genetic algorithm (GA) based heuristic to implement profile-based application assignment. This is motivated by two aspects which are discussed below: 1) Solution time of optimal application assignment methods; and 2) Time Complexity analysis.

The optimal energy-efficient assignment of applications would require exhaustive or brute-force search which enumerates all possible solutions and checks if each solution satisfies problem objectives such as minimum energy consumption [Stephens, 2013]. Another ideal algorithm is the Hungarian Algorithm which is used to find optimal assignment solutions [Luis Bassa and Gil-Lafuente, 2012]. Consider a medium-scale data centre with $m = 2,000$ VMs and $n = 5,000$ applications. If we assume each compilation takes 1 nanosecond. The following are the number of compilations and solution time required for:

1. Hungarian algorithm: $n*m = 10^7$ compilations takes approximately 11 hours to compile;

2. Exhaustive search: $m^n = 10^{16500}$ compilations takes approximately 3.3×10^{16483} years to compile.

This is impractical and confirms that both Hungarian Algorithm and exhaustive search does not perform well with large problem sizes. Figure 2.2 presents an analysis of the very large problem size in terms of the number of compilations and corresponding exhaustive search solution time. Figure 2.2 displays the required compilations, in plot (a) when constant number of 500 application is assigned to a range of 1 to 100 VMs; and plot (c) when a range of 10 to 2000 applications are assigned to a constant number of VMs. Assuming that one compilation takes 1 nanosecond, the solution time of the compilations are plotted in plots (b) and (d). As observed from these plots, performing exhaustive search would take years to compile. Therefore, using exhaustive search, results in $m^n \approx \infty$ compilations/operations which is not a feasible approach.

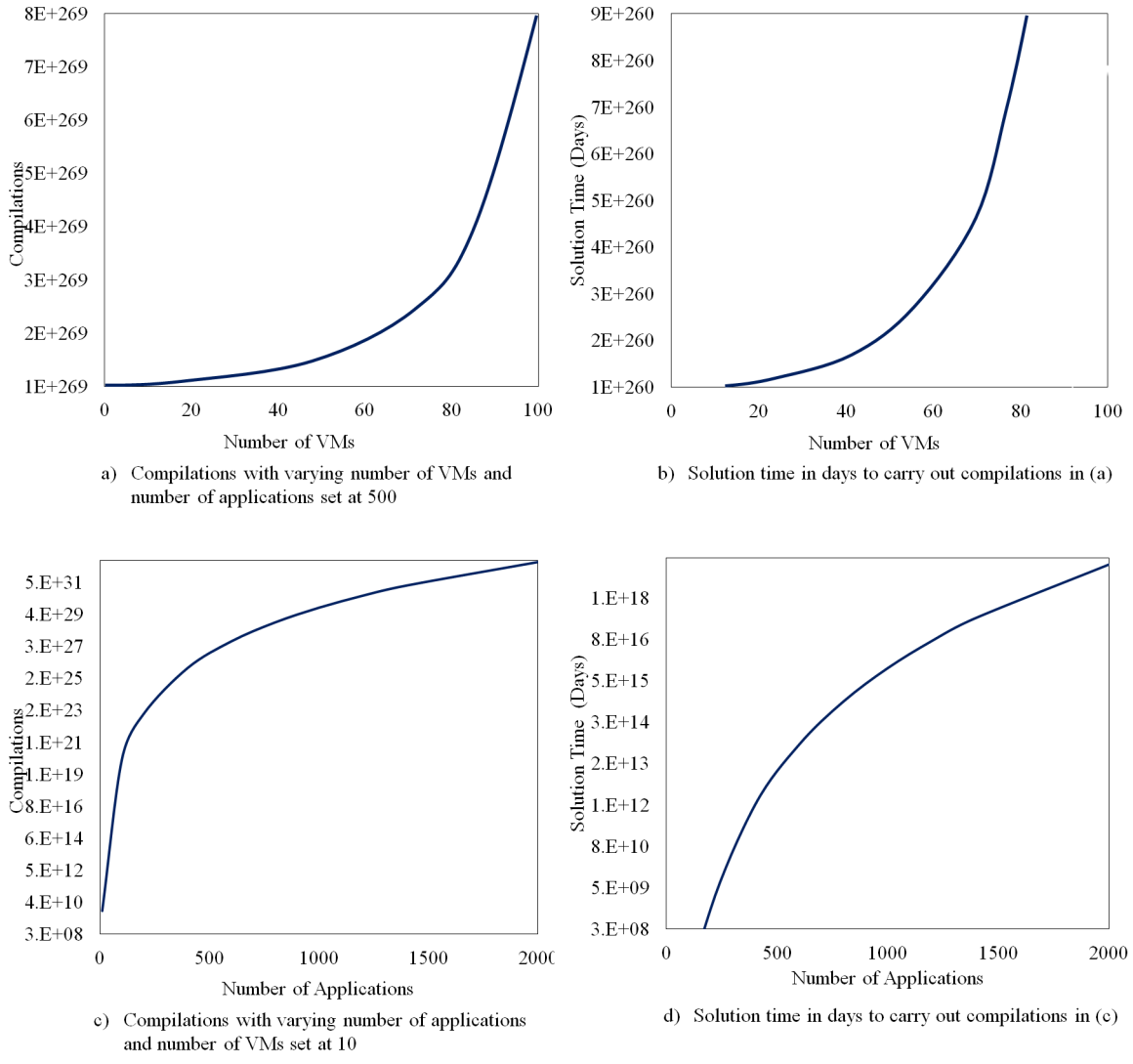


Figure 2.2: Analysis of problem size in terms of compilations and algorithm solution time using exhaustive search for optimal application assignment solutions.

The conclusion from the above calculations is that the problem of finding optimal assignments for a set of applications is NP-hard. Therefore, there is no polynomial time algorithm that can yield optimal solutions for large problems. This motivates us to design a heuristic-based algorithm that is feasible, scalable and provide near-optimal (good) energy-efficient solutions for the large problem size. There needs to be a trade-off between the optimality of assignment solutions and time complexity (time taken to arrive at that solution) of the algorithm.

The motivation to use a heuristic-based algorithm can also be justified by time complexity analysis. The time complexity of existing approaches is calculated from combining the complexity inside and outside the loops (iterations). The time complexity outside the iterations is assumed to be equal to the number of applications, i.e. $O(n)$. To calculate the time complexity inside the iterations, we consider two functions: A) estimated (possible) assignment function and the B) actual assignment to VM function.

For the estimated assignment function, the number of execution is equal to the number of applications n. Consider a set of unassigned applications $\bar{A}$ with $size_i = n - i, 1 \leqslant i \leqslant n$. Initially, $\bar{A}$ contains all the unassigned applications making $size_i = n$. During the course of the algorithm, each application is assigned to a VM thereby decreasing $size_i$ by one. When all applications are assigned then $size_i = 0$. The time complexity for the estimated assignment function is $O(size_i * log_m)$ (m is the number of machines). The time complexity for the actual assignment to VM function is assumed to be $O(log\ size_i)$. Therefore, the time complexity inside the iterations is calculated as:

$$O(size_i * log_m) + O(log\ size_i) \tag{2.1}$$

$$O((n-i) * log_m) + O(log\ (n-i)\ \ \because size_i = n - i \tag{2.2}$$

$$\text{Since } n >> m \text{ and } O(n-i) > O(log\ (n-i)) \tag{2.3}$$

$$\text{Simplified as: } O(n-i) \tag{2.4}$$

The total time complexity of the algorithm combines time complexity inside iterations $O(n-i)$ and outside iterations $O(n)$.

$$\text{Total time complexity of algorithm} \approx O(n(n-i)) \approx O(n^2) \tag{2.5}$$

The total time complexity ($O(n^2)$) means that as the number of applications (input) size (n) increases, the time to solve the problem increases by n^2. Therefore, a heuristic-based algorithm is required to control the large solution space. This thesis uses GA-based heuristic due to its ability to provide a feasible assignment solution on termination at any time. For example there may be a scenario where the assignment solution is required in a short amount of time (on interruption) or another scenario where assignment solution is obtained on the natural (without interruption) termination of the algorithm. GA is well suited to both these scenarios. GA also avoids becoming trapped in a local optima and converges towards a global optima [Bajpai and Kumar, 2010]. That is, it provides good solutions in user specified amount of time. The GA is modified to improve solutions in terms of energy and resource utilization by designing a Repairing Genetic Algorithm (RGA).

The work done in this research presents a profile-based application assignment framework for optimizing the energy of data centres while maintaining the Quality of Service (QoS). Profiles provide prior knowledge of the run-time characteristics of applications, VMs and servers (PMs). Profiling approach follows the sequence: i) build profiles; ii) predict workload; and iii) plan assignment in advance. That is profiles are used to predict future workload and compile assignment solutions in advance, thereby reducing computational complexity. This thesis breaks down the research problem of energy- and performance-efficient profile-based assignment of applications to VMs into four smaller research questions: Profiling, Static Assignment, Dynamic Assignment and Consolidation Problems. Depending on the problem, the objective function and constraints vary. Therefore, the problems are formulated differently and solved using different heuristics as discussed in the following chapters. Assignment solutions are created by combining profiles with a heuristic-based algorithm such as Genetic Algorithm (GA) (discussed in Chapter 4). To the best of our knowledge, an application assignment strategy built on profiling for energy optimization of data centres has not been found in literature.

In the next Chapter, we discuss the development of a solution for the Profiling Problem (Section 1.2) resulting in our first Contribution (Section 1.4).

Chapter 3

Profiling and Profile Building

This Chapter is dedicated to the first research problem of the thesis: the Profiling Problem. Profiles provide prior knowledge of the run-time characteristics of the workload. The concept of profiles and profile building for applications, VMs and servers are discussed here. The key issue of profiling is to build profiles such that future workload can be predicted and assignments can be planned in advance. A profile-based application assignment model is formulated as a combinatorial optimization problem. The model is solved by a scalable Penalty-based Profile Matching Algorithm (PPMA). Experimental studies of the profiling approach is demonstrated to be feasible, scalable and up to 22% more energy-efficient when compared to benchmark approaches.

Some of the work presented in this Chapter was published in our preliminary work [Vasudevan et al., 2014], which introduced profiling as an application assignment method. The paper formulated application assignment as linear optimization with consideration of CPU resources and demonstrated the feasibility and scalability of application assignment based on fully synthetic profiles. This Chapter discusses the following aspects of the Profiling method:

- Profiles: The concept of profiles and profile building. Profiles record power consumption, resource requirements and availability to determine energy cost of application assignment to VM. They are built from the raw data of a real data centre for both VMs and Physical Machines (PMs). Application profiles are built using a well-established workload model.
- Optimization Framework. A penalty-based linear optimization framework is formulated for profile-based application assignment with consideration of memory constrains in addition to CPU resources.

- Solution Algorithm. A penalty-based profile matching algorithm (PPMA) that uses some heuristics is presented to solve the penalty-based optimization problem with considerations of memory, CPU and performance constraints.

Comprehensive case studies are carried out to demonstrate the effectiveness of the profiling method for application management. The experimental results are compared with the Hungarian algorithm, benchmark general and workload history based application management methods.

This Chapter is organized as follows. The notations used throughout this Chapter are listed in Table 3.1. Section 3.1 discusses the concept of profiles. Section 3.2 presents the methodology of building profiles. The following three sections form a profile-based application assignment framework. Section 3.3 models each aspect of the profile-based application assignment problem. This is formulated as a penalty-based constrained optimization problem in Section 3.4. Section 3.5 provides an algorithm to solve the optimization problem. Experimental studies are conducted in Section 3.6. Finally, Section 3.7 summarizes the Chapter.

Table 3.1: Description of notations used in Chapter 3.

Notation	Description
α_j	Ratio of power consumed at max to min util of host V_j
A_i	Application, $i \in I = \{1, \cdots, n\}$
C_{ij}	Energy cost of A_i, $i \in I$, assigned to V_j, $j \in J$
$\eta_{cpu(j)}, \eta_{mem(j)}$	CPU utilization and memory allocation efficiency of V_j, $j \in J$
$f_r(i)$	Requested CPU of A_i, $i \in I$
$f_{vc}^{max}(j)$	Max CPU Capacity of V_j, $j \in J$
i, j, k	Subscripts or indices for applications, VMs and PMs, respectively
I, J, K	Integer sets $I = \{1, \cdots, n\}, J = \{1, \cdots, m\}, K = \{1, \cdots, l\}$
$M_a(i)$	Allocated Memory of A_i, $i \in I$
$M_r(i)$	Requested Memory of A_i, $i \in I$
n, m, l	Total numbers of applications, VMs and PMs, respectively
$p_{cpu(j)}$	Penalty function for CPU utilization of V_j, $j \in J$
P_k	Power consumed (Watts) of S_k, $k \in K$
P_k^{max}, P_k^{min}	Power at respect max and min utilizations of S_k, $k \in K$
S_k	Physical Machine (PM) or Server, $k \in K = \{1, \cdots, l\}$
$t_c(i)$	Completion time of A_i, $i \in I$
$t_r(i)$	Runtime of A_i, $i \in I$, respectively
T	Time Interval
V_j	Virtual Machine (VM), $j \in J = \{1, \cdots, m\}$
x_{ij}	Binary assignment decision variable

3.1 The Concept of Profiles

This Section describes the relevance and effectiveness of profiling for a deterministic application assignment problem. With different purposes, various data centres contribute to the energy consumption and carbon footprint differently. Hyper-scale large data centres are mainly used to host public clouds with dynamic workload. Typical hyper-scale large data centres are those from giant IT corporations like Microsoft, Google, Apple, Amazon, and Facebook. In comparison, small- and medium-scale data centres are typically run by business companies, universities and government agencies and form 95% of the world's total data centres [Whitney and Delforge, 2014]. They generally provide services via private clouds or clusters/grids with virtualized management. Therefore, they have relatively consistent workload. Energy management for small- to medium-scale data centres is globally more significant than that for hyper-scale large data centres with very dynamic workload. This research targets the widely deployed small- to medium-scale data centres.

A nearly consistent workload trace from a real data centre was collected over a period of 14 days to build the VM and PM profiles offline. It is further observed that the pre-set VM and PM parameters like CPU, VCPU, and memory are reviewed every 6-12 months and seldom changed in small- to medium-density data centres. Therefore, the VM and PM profiles are stable for application assignment. Applications are habitually processed over time with varying instructions per cycle and memory. This is incorporated by the profiles on regular update, thereby validating the application profiles for allocation. From our continuous monitoring of a real data center over 14 days, only a very small number of new applications have been observed, for which the profiles built offline have not captured. In other words, data centers managed by universities, government agencies and corporate businesses have relatively consistent applications with varying parameters.

In our study of the workload, applications are categorized as web requests, data analyses, media streaming, e-commerce, social network and others. Some applications are executed in a single task whereas others like data analysis with MapReduce may consist of multiple tasks. Each of the single-task applications has a single profile. For applications with multiple tasks, each of the tasks in an application is treated as a sub-application with a profile. All profiles of the sub-applications in the application share the same application ID.

Occasional new applications whose profiles have not been captured previously will be handled differently. There are three scenarios: 1) a completely new application, 2) an old application with a different dataset for all parameters, and 3) an old application with a different dataset for some of the parameters. For scenario 1, a new application profile is created. The application is assigned to the first VM with low energy cost and available resources. For scenario 2, the application is considered as new and undergoes the same process as scenario 1. Applications in scenario 3 are treated differently. Here, the profiles are updated and the application is assigned to the highest processing VM with low energy cost such that application resource demands are satisfied. To maintain the performance efficiency of the application assignment, the profiles are updated regularly.

Profiles are a set of well-organized information about specific data centre components and their impact on energy consumptions. They provide prior knowledge of the run-time characteristics of the workload. A profile is created initially for each of the applications, VMs and PMs for the data centre. Application profiles include those data related to CPU, memory requirements, actual submission and execution times of individual applications. VM profiles include the data related to CPU processing and memory availability of each node corresponding to interval hours. PM profiles represent the workload and energy consumption of the data centre. Other performance metrics can also be easily integrated into the profiles. After these profiles are created, they are used to create an energy cost matrix to identify the best possible application to VM assignment. Profiling has been previously considered for behavioural and performance analysis. However, to the best of our knowledge, applying profiles in the decision making stage of applications assignment has not been investigated, and thus is a novel concept.

Typically, an extensive amount of data is readily available from the raw data logs of a data centre to build these profiles off-line. Once the profiles are built, regular updates take considerably less processing time. As a result the overhead of creating and maintaining profiles is insubstantial. Application and VM profiles enhance the functions of the allocation manager through 1) retrieval of resource requirements and availability information, and 2) prediction of application arrival and VM workload. This enhancement helps make prompt decisions of application assignment. Applications with profiles are mapped to VMs incurring the least possible energy cost whilst maintaining a trade-off with CPU utilization efficiency, memory and application completion time requirements.

3.2 Profile Building

This Section discusses the process of building profiles for PMs, VMs and Applications. Building profiles requires the accumulation of a large amount of specific data such as energy, CPU, memory, execution times and frequency, standard deviation and interval time. As this data is collected offline from data centre workload logs, the overhead of creating profiles remain insubstantial. Initially, a completely synthetic workload was used to build all profiles. Subsequently, VM and PM profiles are built using the workload trace of a real data centre. This is done to make the profiles more realistic with VM and PM resources corresponding to actual data centre workload. The application profiles are built synthetically by using a commonly used workload model.

3.2.1 Physical Machine Profiles

PM profiles are directly derived from the raw data collected from a data centre. In industrial practice, every data centre keeps logs of their usage and performance measures for various purposes. Our research used the raw data of servers over a period of 24 hours for 14 days (the 5th to 19th of May, 2014) from a real data centre. The observed data centre consists of 263 servers and 1282 virtual machines. The name of the data centre is omitted here due to the commercial confidentiality.

Some of the raw data that have been collected include:

1. Power consumption counted multiple times in 5 minute intervals. The data centre uses StruxureWare Data Center Expert which is a centralized monitoring software that collects energy use on a server rack level amongst other physical infrastructure. This software enables documenting and analysis of regular power consumption of all data centre components.

2. Percent CPU and memory utilization (%) counted multiple times in 60 minute intervals. The data centre contains Red Hat Enterprise Linux Servers and Microsoft 2008 R2 Enterprise Windows Servers. Table 3.2 presents an example of the performance detail report produced by the monitoring software of all data centres. The minimum, maximum and average percent utilization of the resources are displayed along with the standard

deviations in utilization. The count represents the random number of times the monitoring software retrieves server utilization data during the interval.

Table 3.2: Server performance detail report; hourly aggregation for Interval: 12/05/2014 1:00:00 AM

Server Name	Component	Count	Min	Average	Max	SD
RHEL6	% Used Memory	2016	16	17.61	23	0.717
	% Used CPU	2016	0	5.28	23	5.090
RHEL6	% Used Memory	2014	37	39.91	42	0.279
	% Used CPU	2014	2	25.85	80	7.521
Win2008	% Used Memory	809	33.16	37.21	45.2	0.892
	% Used CPU	537	4.336	11.53	50.39	5.613
Win2008	% Used Memory	422	46.48	51.69	76	2.862
	% Used CPU	436	0.8463	7.01	45.94	7.577
RHEL5	% Used Memory	2016	8	8.79	13	0.531
	% Used CPU	2016	8	23.36	68	4.735
RHEL5	% Used Memory	2014	60	61.52	67	0.573
	% Used CPU	2014	5	10.15	46	2.600
Win2003	% Used Memory	411	52.13	60.17	85.98	2.870
	% Used CPU	417	0.5178	3.98	27.03	4.831
Win2003	% Used Memory	355	46.52	48.78	77.39	2.692
	% Used CPU	411	1.007	5.74	49.01	8.660

An analysis of server behaviour for the PMs in the data centre was conducted. For a randomly chosen physical server (server ID: PH015), Figure 3.1 displays the behaviour pattern with respect to CPU utilization over a 24-hour period for four days. The standard deviation of CPU utilization over 24 hours for PH015 is as low as 2.36. Similar analysis is carried out for all servers with respect to minimum, maximum and average CPU utilizations per hour interval. The results demonstrate low variance, therefore showing a near consistent workload to build and utilise the PM and VM profiles for a reasonably realistic allocation strategy.

3.2.2 Virtual Machine Profiles

VM Profiles basically encapsulate the workload history of each of the VMs. The CPU and memory statistics of virtualized physical servers are collected from a real data centre over a period of 14 days. For initial test purposes, it is assumed that each server is capable of hosting up to 15 VMs. The VMs have varying sizes in terms of the CPU and memory allocated to them. The number of VMs per server and their sizes are pre-set during configuration.

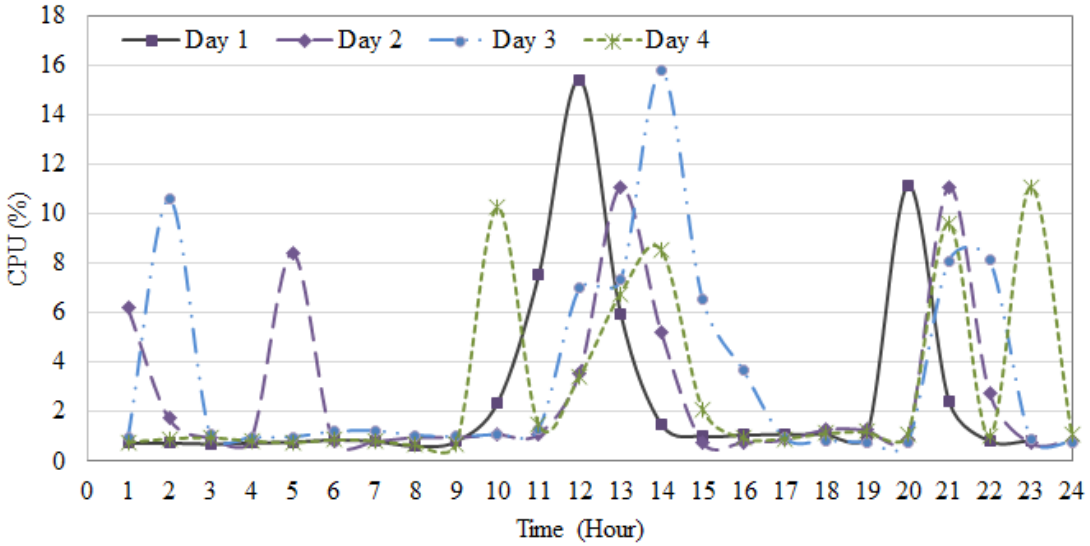


Figure 3.1: The behaviour pattern of physical server PH015.

The parameters of the VM profiles are explained as follows:

1. VM ID: Each VM has a unique identifier. For purposes of testing the feasibility, scalability and efficiency of the profiling method; the maximum number of VMs considered in this Chapter is 2000. Therefore, the VM ID ranges from 0 to 1999.

2. Physical Host: Each VM is hosted by a server. The server can host upto 15 VMs, a VM can only be hosted by one server. This Chapter considers a maximum of 150 physical servers.

3. Total resource capacity: The physical host determines the total resource capacity available to the VM. These parameters can be modified during configuration. The CPU capacity is measured in Million Instructions Per Second (MIPS) and the memory capacity is measured in Bytes. The resource capacity of the VM is determined during the creation of a VM and later modified according to user or application demands.

4. Interval: Represents the time period under consideration. For our experiments, we consider an hourly interval.

5. Used Resource (%): Each VM has a CPU and memory utilization associated with the corresponding time interval. This value represents the resource used by all the applications assigned to the VM. The remaining resource is calculated to determine whether a new application can be assigned to the VM or not and the corresponding efficiency of the

new assignment. These values also determine if the VM is under-loaded or over-loaded, thereby affecting the performance efficiency. VM utilization below 20% is underloaded and above 80% is overloaded.

6. Pointer: The pointer directs to a linked list consisting of all the applications allocated to the VM under consideration. This also records the resource utilized by individual application assigned by the VM during the time interval.

Figure 3.2 presents the profile data structure of 5 random VMs during the interval of 10.00 to 11.00. Each VM has a host ID, total CPU and memory capacity, percentage CPU and memory used, and a pointer to the applications running on the VM associated with it. Example, VM 23 has CPU and memory utilization of 26% and 20% respectively. It hosts 3 applications: A_1, A_8 and A_{10}.

VM ID	PH #	Total CPU (MIPS)	Total Mem (Bytes)	CPU (%)	Mem (%)	pointer
23	31	1000	1000	26.07	19.95	
40	34	1500	3000	8.96	3.00	
63	47	1000	2000	11.64	5.85	
72	65	2000	5000	24.04	21.89	
97	88	1500	4000	6.29	2.12	

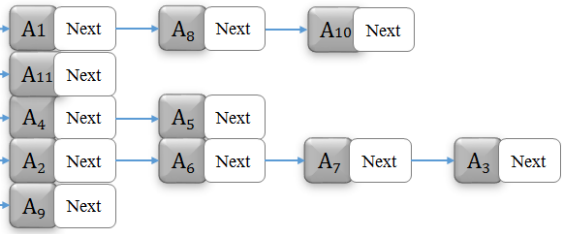


Figure 3.2: Profile data structure of randomly chosen five VMs in interval 10.00-11.00.

3.2.3 Application Profiles

A data centre hosts hundreds of thousands of applications, which are also referred as to tasks. Each application comes with a configuration file, which specifies the CPU, memory and disk space requirements for task execution. In this research, the generated application profiles consider CPU, memory, actual submission time and runtime parameters. While the PM and VM profiles are generated directly from the data logs of the data centre, the data logs from the data centre do not include all information for building application profiles directly. Therefore, a commonly used synthetic workload model designed by Lublin and Feitelson [2001] is adopted to build application profiles through some distributions. This is particularly for creating application parameters such as submission time, runtime and resource requirements. For example, the workload generation model uses a gamma distribution to generate application waiting times for workload simulation.

The application profile parameters are generated as follows for our workload simulation. Initially, the number of applications is calculated for every hour using a cumulative distribution function. The submission time, which is a random variable, is modelled with gamma distribution for each application (Algorithm 1) and is an input variable for our simulation experiments. The approximate CPU percentage and memory required to run the application is calculated using a two-stage uniform distribution. The application runtime is calculated using a hyper-gamma distribution [Lublin and Feitelson, 2001] (Algorithm 2).

Algorithm 1: Application Submission Time

```
1 Calculate number of applications per hour using Cumulative Distribution Function;
2 for Each Application do
3 |   Generate random variable from gamma distribution;
4 Set submission time to generated random variable;
```

Algorithm 2: Application Runtime

```
1 Define parameters for gamma distributions 1 and 2, respectively;
2 Define relation probability between the two gamma distributions;
3 Generate a uniformly distributed random number between the range of 0.0 to 1.0;
4 if (Generated a random number ≤ Relation probability) then
5 |   Gamma distribution 1 is active;
6 else
7 |   Gamma distribution 2 is active;
8 Generate a random variable from the active gamma distribution;
9 Set runtime to the generated random variable;
```

It is worth mentioning that the runtime of workloads have been measured after an application executes on a VM. But allocating an application to different VMs leads to different completion times. However, before the runtime can be actually measured, the application must be allocated to one of the VMs in terms of some criteria determined by a number of parameters including an estimated runtime to maximize the optimization function. Therefore, an initial runtime of workloads is derived using a distribution and is included in the application profiles initially. In general, the VM where the completion time is closest to the initially generated runtime is preferred.

Application profiles are generated using the workload model in C programming language. Figure 3.3 shows the data structure of randomly chosen five application profiles with the following four parameters:

1. Application ID: A unique identifier associated with each application. For our initial experiments, 5000 applications are considered with the application IDs ranging from 0 to 4999.

2. Submission Time (s): The submission time represents the time instant in seconds at which the application arrives at the data centre. Collecting the submission times of applications allows us to segregate applications arriving at a certain time interval and utilize this record to make energy-efficient assignment decisions.

3. Max Runtime (s): The runtime represents the maximum time duration (in seconds) in which the application is active to successfully complete execution. The runtime of an application varies with the VM host. Therefore, the maximum runtime of the application determines if the current assignment of the application to a VM resulting in a current runtime is efficient or not; i.e. current runtime is lower than the max runtime.

4. Requested CPU (%) and Memory (Bytes): The requested CPU and memory represent the resource requirements to successfully execute the application. Any application assignment decisions will be carried out only if the requested resources can be provided by the VM considering resource availability.

Submission Time (s)	App ID	pointer	Runtime (s)	CPU (%)	Mem (Bytes)
16	9199		193	74	122281
61	1310		211	15	57688
76	24183		96	55	97573
296	45276		88	9	7200
17197	45299		1108	24	36516

Figure 3.3: Profile data structure of randomly chosen five applications.

After the application profiles are generated, a parser code is written in C++ programming language to process and incorporate the Profile data into our heuristic algorithm for application assignment.

3.3 Formulation of Problem Elements

With various profiles built in the last Section, this Section models each aspect of the profile-based application assignment problem. In addition to energy saving, other objectives of the

framework include effective performance levels in terms of execution time and CPU utilization efficiency. In essence, the framework aims to minimise the CPU energy of the physical node, which hosts the VMs for the timely and successful execution of applications.

For model development, some notations are defined below. Let us denote:

- Application: $A_i, i \in I = \{1, \cdots, n\}$;
- Virtual Machine (VM): $V_j, j \in J = \{1, \cdots, m\}$;
- Physical Machine (PM): $S_k, k \in K = \{1, \cdots, l\}$;
- Integer sets: $I = \{1, \cdots, n\}, J = \{1, \cdots, m\}, K = \{1, \cdots, l\}$; and
- Total numbers of applications, VMs and PMs, respectively: n, m, l.

A binary decision variable $x_{ij}, i \in I, j \in J$ represents the assignment of an application A_i, $i \in I$, onto a VM V_j, $j \in J$:

$$x_{ij} = \begin{cases} 1 & \text{if } A_i \text{ is allocated to } V_j;\ i \in I, j \in J, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Furthermore, for an application $A_i, i \in I$, CPU and memory requirements are denoted by $f_r(i)$ and $M_r(i)$, respectively. If the application $A_i, i \in I$, is hosted by the VM $V_j, j \in J$, the actual memory allocated from the V_j to the A_i is represented by $M_a(i)$. The CPU capacity of $V_j, j \in J$, is denoted by $f_{vc}^{max}(j)$.

A profile-based linear programming model has been designed to identify and carry out near-optimal assignment of applications on VMs. The objectives include resource utilization efficiency, application completion time within its deadline and minimised energy cost. This will be discussed in more detail below.

CPU Utilization Efficiency.

The CPU utilization efficiency of a VM $V_j, j \in J$, is a ratio of the total CPU percentage in use by the applications to the total CPU capacity of the VM. It is represented by $\eta_{cpu}(j) \in [0, 1]$ and derived at a time instance before application assignment as follows:

$$\eta_{cpu(j)} = \frac{\sum_{i=0}^{n} f_r(i) x_{ij}}{f_{vc}^{max}(j)};\ i \in I, j \in J \tag{3.2}$$

where $f_r(i)$ represents the variable CPU requirement of the application $A_i, i \in I$; and $f_{vc}^{max}(j)$ is the total CPU capacity of VM $V_j, j \in J$, as defined previously.

A penalty function is introduced to encourage applications to be packed onto active VMs such that the maximum CPU capacity is utilized. Higher the CPU utilization, lower the penalty. If the CPU utilization efficiency falls below 0.5, then a penalty equal to the capacity of the VM $f_{vc}^{max}(j)$ is applied. When the utilization efficiency increases, the penalty decreases by half $f_{vc}^{max}(j)/2$. The maximum CPU utilization efficiency incurs 0 penalty. The CPU utilization efficiency constraint restricts overloading the VMs by considering any solution with $\eta_{cpu(j)} > 1$ as infeasible by assigning a high penalty of ∞. Therefore, the penalty $p_{cpu(j)}$ for the different values of $\eta_{cpu(j)}$ is set as follows:

$$p_{cpu(j)} = \begin{cases} f_{vc}^{max}(j), & \eta_{cpu(j)} \leqslant 0.5 \\ f_{vc}^{max}(j)/2, & 0.5 < \eta_{cpu(j)} < 1 \\ 0, & \eta_{cpu(j)} = 1 \\ \infty, & \eta_{cpu(j)} > 1 \end{cases} \tag{3.3}$$

Memory Allocation.

When application $A_i, i \in I$, with memory requirement $M_r(i)$ is hosted by VM $V_j, j \in J$, the memory assigned from V_j to A_i is denoted by $M_a(i)$, as notationally defined previously. The memory allocation efficiency $\eta_{mem(j)}$ is the ratio of $M_a(i)$ to $M_r(i)$ as per the application profiles:

$$\eta_{mem(j)} \quad = \quad M_a(i)/M_r(i),\ i \in I, j \in J \tag{3.4}$$

Because $M_a(i) \geqslant M_r(i)$, it follows from Equation (3.4) that $\eta_{mem(j)} \geq 1$. The memory allocation constraint ensures that the application has the required memory to successfully execute.

Application Completion Time.

The application profiles include approximate average runtime $t_r(i)$ for application $A_i, \forall i \in I$. In order to ensure application assignment efficiency, the actual completion time $t_c(i)$ taken by the individual VM $V_j, j \in J$, to successfully execute the application $A_i, i \in I$, must fall

within a threshold value set at $1.5 \cdot t_r(i)$, i.e. the scope of $t_c(i)$ is expected to fall within 50% more than $t_r(i)$.

$$t_c(i) \leqslant 1.5 \cdot t_r(i); \;\; i \in I, j \in J. \tag{3.5}$$

This constraint is put in place to ensure that the execution efficiency of the application is not compromised when producing energy-efficient assignment solutions. Every application has discrete completion times corresponding to different VM hosts. The completion times depend on the CPU availability, speed and memory available to a VM.

For example, an application assigned to VM V_1 may have the smallest energy cost and a long completion time. However, the same application executed in VM V_2 results in a slightly higher energy cost but shorter completion time. The latter provides a better solution in terms of computing performance efficiency.

Energy Cost.

Energy efficiency of the presented profile-based application assignment approach for data centres is the main objective of this research. It is modelled by minimizing the total energy cost of the application assignment. Energy cost is directly proportional to the approximate power required to carry out an application in a VM. Approximate power consumed by a physical node is calculated from the power model defined by Blackburn [2008]. The power model represents a linear relationship between the power and CPU utilization as graphically shown in Figure 3.4. From this linear model, the energy cost, C_{ij}, of executing application $A_i, i \in I$, on VM $V_j, j \in J$, is calculated as the product of the CPU requirement $f_r(i)$ of the application A_i and a coefficient α_j:

$$C_{ij} = \alpha_j \cdot f_r(i) \tag{3.6}$$

where the coefficient α_j characterizes how energy-efficient the VM V_j is to host the application A_i, and it is the ratio of power consumed at maximum and minimum utilizations.

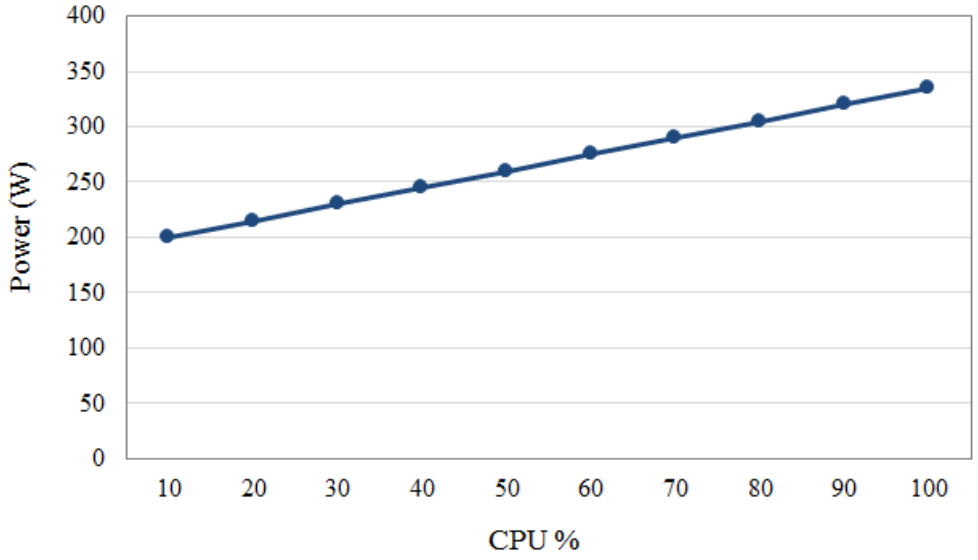


Figure 3.4: Power consumption versus CPU utilization [Blackburn, 2008].

3.4 Profile-based Application Assignment Model

The application assignment problem is formulated as a penalty-based constrained optimization problem in this Section. The profile-based application assignment model seeks to make the best possible use of the available resources for greener and more energy-efficient assignment solutions.

The Profile-based Energy-Efficient Application Assignment Model is mathematically defined as follows:

$$
\begin{aligned}
\min z \quad &= \quad \sum_{j=1}^{m}\sum_{i=1}^{n} C_{ij}x_{ij} + \sum_{j=1}^{m} p_{cpu(j)} \\
\text{s.t.} \quad & \quad \eta_{mem(j)} \geqslant 1,\ \forall j \in J; \\
& \quad \sum_{i=1}^{n} f_r(i)x_{ij} \leq f_{vc}^{max}(j),\ \forall j \in J; \\
& \quad t_c(i) \leq 1.5 \cdot t_r(i),\ \forall i \in I, j \in J; \\
& \quad \sum_{j=1}^{m} x_{ij} = 1,\ \forall i \in I; \\
& \quad x_{ij} = 0 \text{ or } 1,\ \forall i \in I, j \in J.
\end{aligned}
\tag{3.7}
$$

The following are the constraints for our linear programming model:

- Constraint 1: Memory Allocation. The constraint ensures that the memory provisioned

for applications assigned to a VM does not exceed the memory capacity of the VM (Equation 3.4).

- Constraint 2: CPU Allocations. The total CPU required by all applications in a VM must not exceed the maximum VM CPU capacity.

- Constraint 3: Application Completion Time as shown in Equation (3.5). The assignment needs to consider the discrete completion time that individual applications take in distinct virtual machines. These times are mainly dependent on the CPU speed and memory available to a virtual machine. Therefore, assignments that allow for application completion time close to the overall average completion time should take place. This ensures that the applications deadlines are met.

- Constraint 4: Each application A_i must be assigned to one virtual machine V_j only, in order to avoid redundancy in the form of multiple virtual machines attempting to execute the same application.

- Constraint 5: Binary Constraint as shown in Equation (3.1).

Figure 3.5 gives a flowchart for the working of the profile-based application assignment framework. The model is solved using a Penalty-based Profile Matching Algorithm proposed in the following subsection.

3.5 Penalty-based Profile Matching Algorithm

The Penalty-based Profile Matching Algorithm (PPMA) is designed to solve the profile-based application assignment model in Equation (3.7). The primary objective of PPMA is to improve energy efficiency with respect to application assignment problem in data centres. The side constraints include CPU utilization efficiency, memory and application completion time as discussed in the previous Section. To address the issues of low computing efficiency and scalability in deriving optimal solutions, PPMA aims to obtain near-optimal assignment solutions with high computing efficiency and good scalability. Therefore, PPMA makes use of some heuristics to derive solutions. Developing heuristics rather than employing conventional solution techniques simplifies the problem-solving process, thus improving the scalability of the problem-solving (discussed in Section 2.6). This is advantageous to conventional assignment algorithms

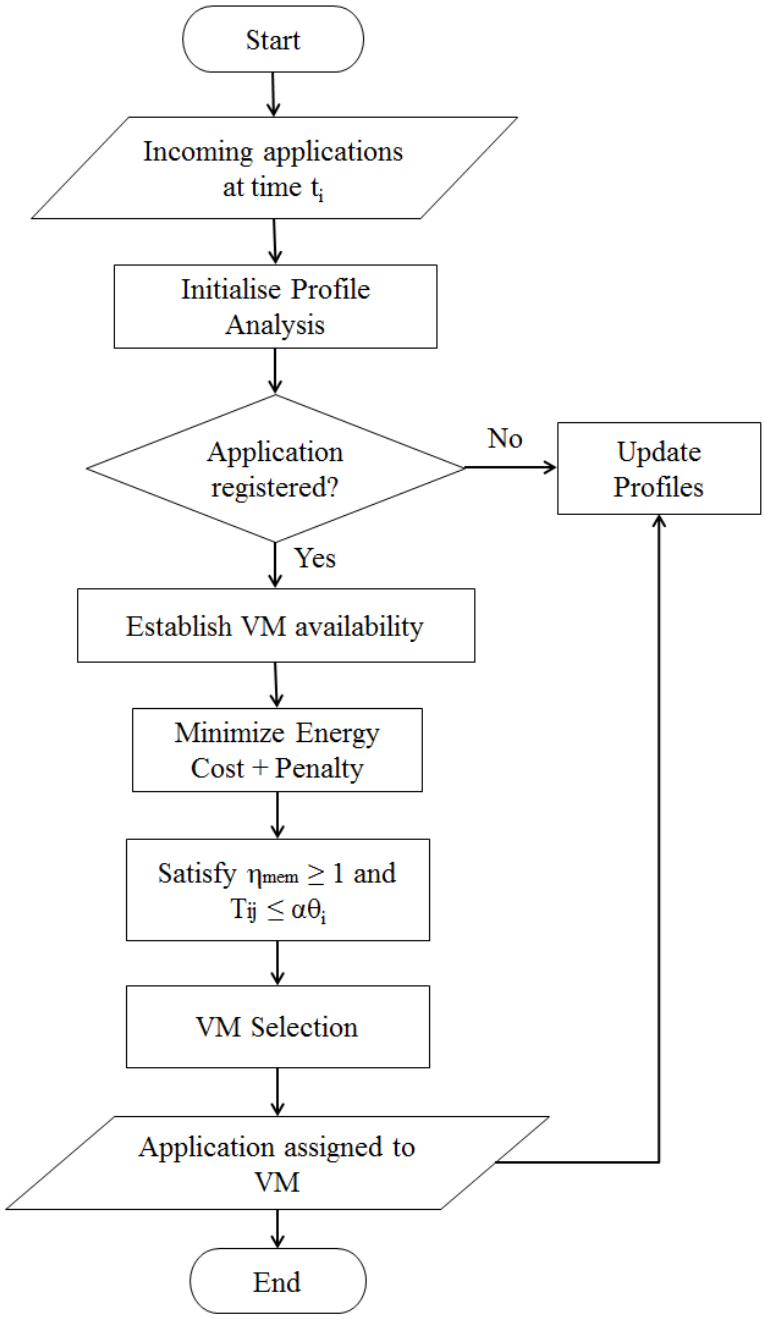


Figure 3.5: Profile-based linear programming model.

such as Hungarian Algorithm, which obtains optimal solutions but has poor scalability [Chaobo and Qianchuan, 2008, Vasudevan et al., 2014, Winter and Albonesi, 2008].

Algorithm 3 presents the pseudo-code for PPMA. The initial and most crucial element of the algorithm is the deciphering of the Profiles. Once the necessary data have been retrieved, the energy cost matrix $[C_{ij}]_{n\times m}$ is built. As the profiles are updated periodically, the energy cost of allocation is also updated regularly. This allows real-time events to be taken into consideration,

Algorithm 3: Penalty-based Profile Matching Algorithm (PPMA)

```
1  Read energy cost [C_ij]_{n×m} data from profiles;
2  Read application CPU and memory requirements from profiles;
3  Set scope to number of applications to be allocated;
4  while Within Scope do
5      Initialise [x_ij]_{n×m} and [Temp[i][j]]_{n×m} as null matrices;
6      Copy matrix E_ij to a temporary matrix Temp[i][j];
7      for Every Application do
8          Set Temp[i][1] as the minimum value;
9          for Every VM do
10             if Temp[i][j] is minimum then
11                 Update Temp[i][j] as the minimum value;
12         Subtract minimum value from each value;
13     for Each matrix Temp value do
14         if Zero then
15             Calculate penalty;
16             Check memory allocation constraint;
17             Check application completion time constraint;
18         if Constraints are satisfied then
19             Confirm allocation as x_ij = 1;
20             break;
21         else
22             Set value Temp[i][j] to a large number;
23             goto step 7;
```

thus improving the efficiency of the allocation manager.

The assignment solution is verified in the algorithm by determining the CPU utilization, memory efficiency and application completion time achieved. If all the conditions are satisfied, the algorithm moves on to the next assignment. In case of assignment unsuitability, the next best assignment is considered and the same process follows until a suitable assignment is achieved and the matrix $[C_{ij}]_{n\times m}$ and penalty functions are modified accordingly.

3.6 Experimental Studies

This Section conducts experimental studies to demonstrate the profile-based application assignment approach presented in this Chapter. The effectiveness of the approach is evaluated from the following three aspects: feasibility, scalability, CPU utilization, and energy efficiency. The Section begins with experimental setups followed by detailed experimental studies.

Experimental Setups. The experimental studies are conducted using two different test setups: Test Setup 1 and Test Setup 2. Originally investigated in our preliminary work [Vasudevan et al., 2014], Test Setup 1 is used to determine the feasibility and scalability of our approach. Test Setup 2 is used to determine the efficiency of the profiling application management approach over general and workload history approaches. As there are no benchmarks available for application assignment to VMs, we assume General/Random Assignment and Workload History based Assignment [Luo et al., 2013] as benchmarks to conduct evaluations of the profile-based application assignment approach. Shown in Table 3.3, the two setups are described below:

Table 3.3: Two test setups with different scenarios for Chapter 3 experiments.

Test Setup 1 (100 PMs)						
Scenario	**1**	**2**	**3**	**4**	**5**	
VMs	400	400	800	800	1000	
Applications	500	1500	2000	2500	4000	
Test Setup 2 (150 PMs)						
Scenario	**6**	**7**	**8**	**9**	**10**	**11**
VMs	100	400	800	1200	1600	2000
Applications	500	1000	2000	3000	4000	5000

- **Test Setup 1:** A data centre consisting of 100 PMs with an average of four to ten VMs each is considered. The total number of VMs ranges from 400 to 1000. The total number of applications varies from 500 to 4000. The scenarios of Test Setup 1 are presented in the first half of Table 3.3.

- **Test Setup 2:** A data centre consisting of 150 PMs is considered. Each server is capable of hosting up to 15 VMs. For our evaluation, six different scenarios are considered where the number of applications ranges from 500 to 5000 with corresponding number of VMs as shown in the second half of Table 3.3.

In our experiments, a real-world system is investigated. Data logs are derived from a real data centre to create realistic VM and PM profiles. The application profiles are synthetically generated as in our preliminary work [Vasudevan et al., 2014].

All evaluations are carried out on a Windows platform of Intel(R) Core(TM) i7-2640M CPU at 2.80GHz using C and Python programming. As there are no off-the-shelf products for

simulating test scenarios of application assignment in data centres, we have developed our own simulation for all experiments.

Evaluation criteria for Test Setup 1. The results derived from PPMA will be compared with those obtained from Hungarian Algorithm (HA). The HA is an ideal algorithm that provides optimal solutions in terms of energy-efficiency. However, HA is only feasible for small problem sizes (discussed in Section 2.6, Chapter 2). Comparison with HA demonstrates the proximity of our algorithm's energy-efficiency to that of the optimal solution. Other factors comprising the efficiency criteria include feasibility, scalability and CPU utilization efficiency. A high-level description of Hungarian Algorithm is depicted in Algorithm 4, which is self-explained.

Algorithm 4: Hungarian algorithm (HA).

```
1  Convert [C_ij]_{n×m} into square energy cost matrix using dummy values ;
2  for Each Row do
3      Identify and subtract minimum value from all elements;
4  for Each Column do
5      Identify and subtract minimum value from all elements;
6  while Solution matrix not complete do
7      if Column contains more than one '0' element then
8          Repeat step 2 forall columns ;
9      for Each Column do
10         Identify columns with negative elements ;
11         Select minimum value and add to each element ;
12     Flag rows and columns with '0' elements ;
13     Identify and subtract minimum value from unflagged elements ;
14     Add minimum value from unflagged elements to twice flagged elements ;
```

Evaluation criteria for Test Setup 2. The effectiveness of the Profile-based application assignment approach in terms of execution time and energy-efficiency is evaluated with comparison with benchmark general and workload history based assignment approaches. The general application assignment is the simplest form of allocation and does not implement any efficiency strategy. The applications are allocated at the time of their arrival to the first available VM that fits the execution requirements in CPU, memory and runtime. Workload History based application assignment utilizes the recorded logs of CPU cycles with corresponding time of the VMs to make allocation decisions. This approach functions on the assumption that workload behaviour of a data centre varies little during day-to-day operations. However,

it only considers the VM information, whereas our Profiles are built for applications, VMs and PMs. Moreover, workload history approach does not have the option of updating data unlike the profiling approach.

All three approaches: General, Workload History and Profiling are implemented with a simple First-Fit Decreasing (FFD) assignment algorithm in the three-layer energy management (Figure 1.1). The application assignment problem resembles a bin-packing problem:

1. **General Assignment** - The applications arrive at the data centre. The CPU requirement is determined. Applications are arranged in terms of decreasing CPU requirement. The FFD is invoked and the application assigned to the first VM that can accommodate the requirements. Algorithm 5 gives the process of this approach.

2. **Workload History Assignment** - During time interval $T-1$, the VMs are arranged in decreasing order of CPU availability as per the workload logs. Applications arriving at the data centre during time interval T are assigned to the first suitable VM with the help of FFD algorithm. Algorithm 6 represents the process of this approach.

3. **Profile-based Assignment** - During time interval $T-1$, the energy cost of allocation of each predicted application to a VM is retrieved. A VM yielding the lowest cost is selected. At interval time T, the FFD allocates the application to the pre-selected VM with the minimum energy cost incurred. Algorithm 7 describes the process of this approach.

Algorithm 5: General Application Assignment

```
for Each Application do
    Determine CPU and memory requirement;
    if Requirement satisfied then
        Invoke FFD Algorithm (Allocate to first available VM);
```

Algorithm 6: Workload History Application Assignment

```
for Interval Time T-1 do
    Workload Logs: VM arranged in decreasing order of CPU availability at Time T;
for Interval Time T do
    for Each Application do
        if Requirement satisfied then
            Invoke FFD Algorithm (Allocate to first available VM);
```

Algorithm 7: Profiling-based Application Assignment

```
for Interval Time T-1 do
    Profiles: Determine applications arriving at Time T;
    Profiles: Retrieve associated energy cost of allocation of each application;
    Profiles: Select best possible VM hosts Selected_VM = {VM_1, VM_2, ...};
for Interval Time T do
    for Each Application do
        if Requirement satisfied then
            Invoke FFD Algorithm (Allocate to first available VM);
```

3.6.1 Feasibility

Test Setup 1 is used to validate the feasibility of the profile-based application assignment framework. The application assignment results for Scenario 2 are presented in Figure 3.6. Analysing the results shows that an average number of 15 applications are hosted by each server through VMs, with a maximum of 32 applications hosted by a server. More than 25 applications are hosted by 15% of the servers individually. 11% of the total servers are idle and can be switched off by the allocation manager. The proposed approach successfully solves the penalty-based linear optimization model (Equation 3.7) and satisfies the resource constraints, thereby supporting the feasibility of the presented Profile-based Assignment Model.

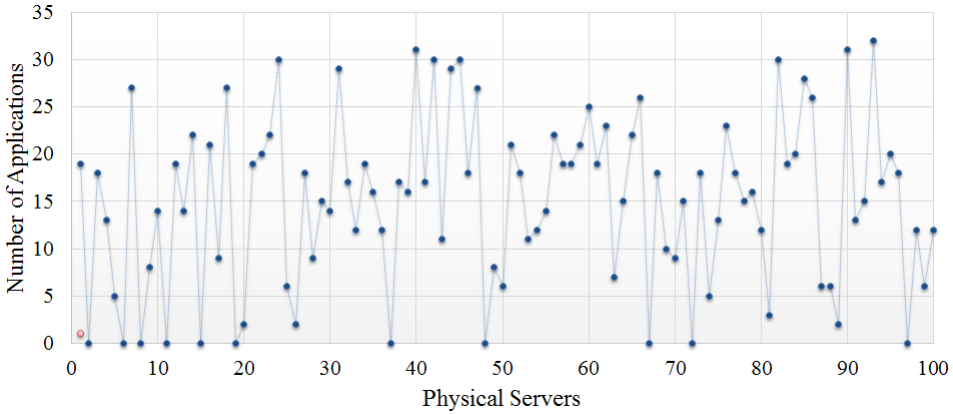


Figure 3.6: Application assignment for Scenario 2 of Test Setup 1.

3.6.2 Scalability

The scalability of the Penalty-based Profile Matching Algorithm (PPMA) is compared with that of the Hungarian Algorithm (HA) described in Algorithm 4. Both algorithms are applied to

the five scenarios of Test Setup 1 (Table 3.3). The solution time in seconds for each of the two algorithms is obtained and tabulated in Table 3.4. The results demonstrate that as the numbers of applications and VMs increase, the PPMA is capable of finding near-optimal solutions in much lesser time than the HA. The HA gives optimal assignment solutions but compromises heavily on the time taken to obtain the solution due to the large problem size. This demonstrates that the presented PPMA scales well for large problem sizes.

Table 3.4: Comparisons of solution time (sec) of PPMA and HA for Test Setup 1.

Scenario	**1**	**2**	**3**	**4**	**5**
Hungarian Algorithm	4	27	41	72	248
PPMA	5	22	26	31	52

3.6.3 CPU Utilization Efficiency

This subsection demonstrates that the presented profile-based assignment framework makes the best possible use of available resources such as the CPU of the server nodes. The scenarios from Test Setup 1 (Table 3.3) are considered in this case study. There is a high variance in PPMA and HA results for problems of smaller sizes. This is demonstrated in Figure 3.7 for the CPU utilization efficiency variation graph over a 24 hour period for Scenario 1. However, as the ratio of applications assigned to VMs increases with the problem size, the CPU utilization efficiency also increases. The average CPU utilization efficiency of the presented PPMA and HA for all scenarios of Test Setup 1 (in the first half of Table 3.3) is compared in Table 3.5. The PPMA achieves results that are close in utilization efficiency to the HA. This is evidenced by a decrease in variation from 19% to 1.1%.

Table 3.5: Average CPU utilization efficiency (η_{cpu}) for Test Setup 1.

Scenario	**1**	**2**	**3**	**4**	**5**
Hungarian Algorithm	0.486	0.531	0.545	0.578	0.702
This work (profile-based)	0.394	0.499	0.526	0.569	0.694

Although HA is more energy-efficient than PPMA, it compromises on the consistency and scalability of the assignment solutions. The PPMA maintains consistent CPU utilization efficiency with the increase in the scale of the assignment problems.

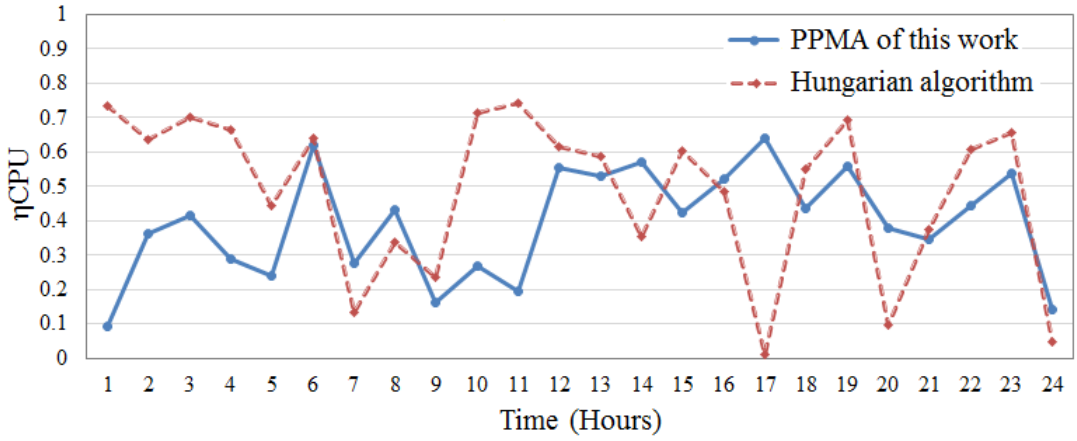


Figure 3.7: Average CPU utilization efficiency over 24 hours for Scenario 1 of Test Setup 1.

3.6.4 Energy Efficiency

This subsection demonstrates the energy efficiency of the profile-based application assignment approach for Test Setups 1 and 2 implementations. Test Setup 1 (discussed in our previous work [Vasudevan et al., 2014]) is used to compare the energy results derived by the PPMA and the optimal HA. Test Setup 2 is used to validate the energy-efficiency of the Profiling application management approach when compared with the commonly used General and Workload application management approach.

Energy Efficiency in Test Setup 1

The energy-efficient solutions provided by the PPMA is compared with that of the optimal results provided by the HA. In order to evaluate the energy-efficiency, the average CPU utilization is deduced after the application assignment using both algorithms. The energy consumption E for the Test Setup 1 scenarios (shown in the first half of Table 3.3) is then calculated using the following equations Blackburn [2008].

$$P_k = \frac{(P_k^{max} - P_k^{min}) * \eta_{cpu(k)}}{100} + P_k^{min}, \tag{3.8}$$

$$E = \int_{t_0}^{t_1} P_k(t)dt \tag{3.9}$$

The power consumed for machine k at maximum and minimum utilization is given by P_k^{max} and P_k^{min}, respectively. For calculation purposes, it is assumed without losing generality that $P_k^{max} = 350$W and $P_k^{min} = 200$W. Total CPU utilization of the server is represented by $\eta_{cpu(k)}$.

Figure 3.8 demonstrates the energy consumption graph of a server in a 24 hour period for both PPMA and HA. The average energy consumptions for the PPMA and HA are 286.5 Wh and 269.75 Wh, respectively. The results confirm that the PPMA is only 5.85% worse in energy-efficiency than the ideal HA. In order to demonstrate the decrease in variation of energy consumption results as the problem size increases, a bar graph displaying the total energy consumption for all scenarios of Test Setup 1 is presented in Figure 3.9. The PPMA results show a 11.8% to 0.4% variation from the HA solutions. This proves that the proposed PPMA is sufficiently energy-efficient.

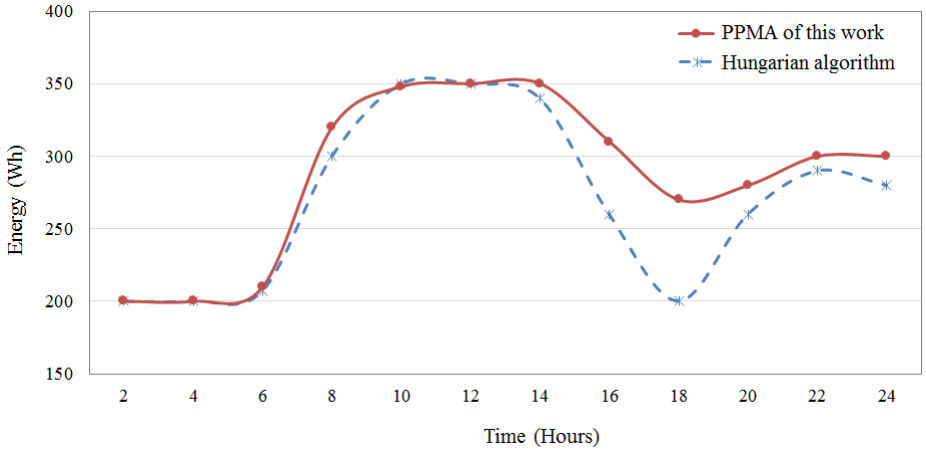


Figure 3.8: Energy consumption of a server using PPMA and HA.

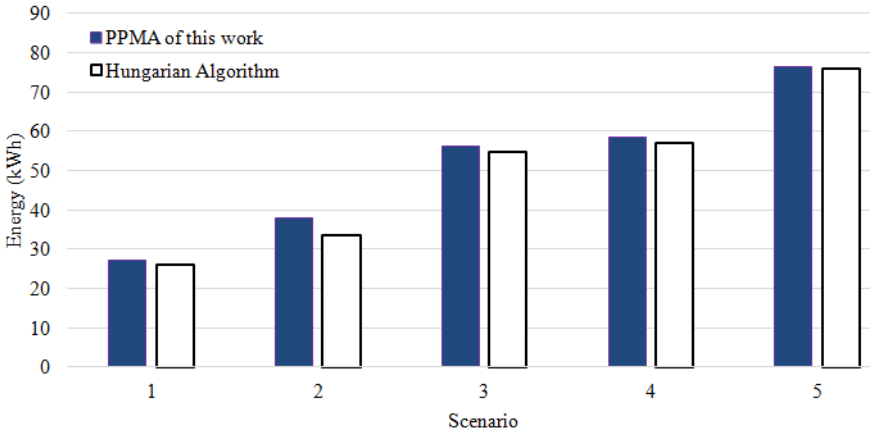


Figure 3.9: Total energy consumption for Test Setup 1 scenarios.

Energy-Efficiency in Test Setup 2

The effectiveness of the Profile-based application assignment approach in terms of execution time and energy-efficiency is evaluated with comparison with benchmark general and workload

history based assignment approaches. Test Setup 2, as seen in the second half of Table 3.3, is used to compare the results obtained by the following three approaches:

- General Application Assignment;
- Workload History based Application Assignment; and
- Profile-based Application Assignment

The results of energy-efficiency and execution time for the six different test scenarios in Test Setup 2 are presented in Table 3.6 for the general, workload history and profiling approaches.

Table 3.6: Energy-efficiency and execution time from Test Setup 2.

Scenario	Profiling		Workload History		General	
	Energy (Wh)	Time (s)	Energy (Wh)	Time (s)	Energy (Wh)	Time (s)
6	25623	1.2	26186	1	27015	1
7	25948	1.9	26748	2.2	28975	1.5
8	26782	4.4	27493	4.8	32675	4.1
9	27835	5.7	30759	7.1	33402	8.3
10	28940	6.9	32472	7.9	37238	12.4
11	32752	8.6	35871	9.2	39478	14.7

Consider the execution time behaviour as seen in Figure 3.10. The General allocation initially has the lowest execution time upto 1500 applications. However, as the number of applications increases, there is a corresponding increase in the execution time. Both workload history and profiling approaches have a steady, consistent, and linear increase with the number of applications. On examination, the profiling approach is 5% more efficient than the workload history approach in execution time.

Figure 3.11 shows the total energy consumption of the data centre with respect to the increasing number of applications for all three approaches. The Profiling method is 22% and 10% more energy-efficient than the benchmark approaches of General and Workload History based application assignment. The General approach consumes the most energy as application-VM allocations are not optimal due to the absence of energy cost constraints. The Workload History approach is efficient upto 2000 applications, however increases significantly with the increase in the number of applications. The Profiling approach outperforms the other two approaches in energy consumption due to energy cost based allocations derived from the profiles.

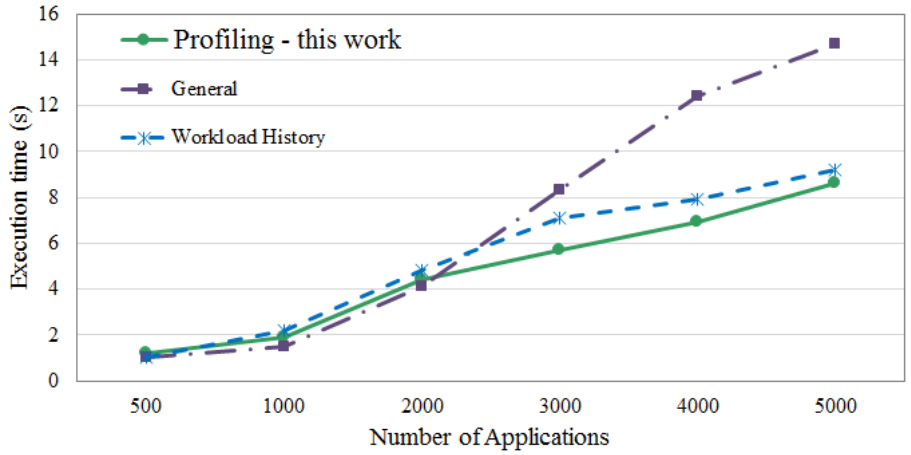


Figure 3.10: Comparisons of execution time for General, Workload History and Profiling approaches.

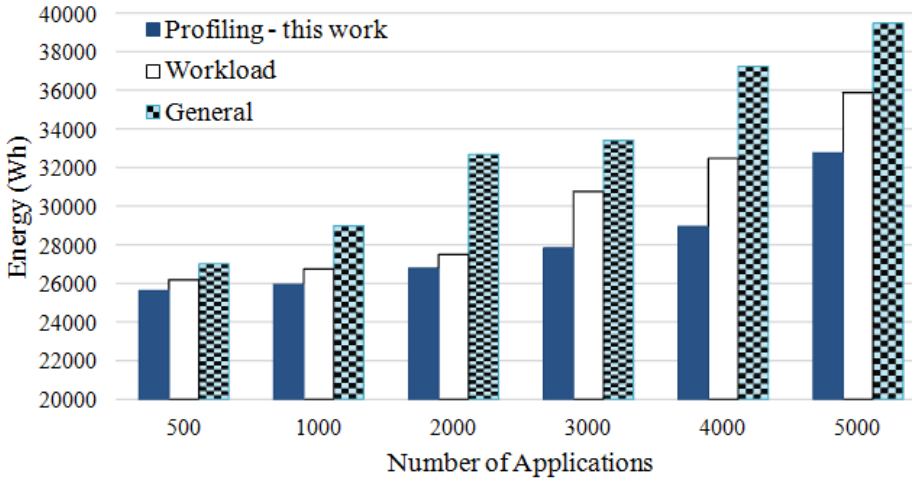


Figure 3.11: Comparisons of energy-efficiency for General, Workload History and Profiling approaches.

Table 3.7 provides an overview of the three different approaches considered in the experimental evaluations. The standard deviations in terms of energy and execution time demonstrate that our Profiling approach is consistent and more energy-efficient than the other approaches.

Table 3.7: Overview of the three approaches: general, workload history and profiling ($\surd$: known; $\times$: unknown).

Data/Strategy	**General**	**Workload History**	**Profiling**
Virtual Machine	$\times$	$\surd$	$\surd$
Application	$\times$	$\times$	$\surd$
Std Deviation - Energy	4735.38	3808.49	2639.53
Std Deviation - Exec Time	5.74	3.27	2.87

3.6.5 Further Discussions on Experimental Studies

Experimental studies have been conducted on the feasibility, scalability, effectiveness, CPU utilization efficiency and energy-efficiency of the proposed Profile-based Application Assignment approach. The experimental results are summarized as follows:

- The PPMA is feasible and scalable within the tested range of 100 to 2000 VM nodes;
- There is a trade-off between scalability and CPU utilization efficiency for increasing problem sizes;
- The Profile-based Application Assignment approach is up to 22% more energy-efficient with steady execution times when compared to commonly used benchmark General and Workload History assignment approaches; and
- The energy efficiency achieved from our profile-based approach is close to that of the optimal Hungarian Algorithm solution.

It is worth mentioning that the overhead of the profile-based application assignment is minimal for data centres with relatively consistent workloads considered in this Chapter. The profiles of the data centres can be established offline. The un-profiled workload that requires online processing is insubstantial. With the established profiles, static assignment of applications to virtual machines can be scheduled in advance.

3.7 Summary of the Chapter

One of the significant research problems concerning data centres and cloud computing is how to reduce the energy consumption whilst maintaining high performance efficiency. This Chapter explores the concept of profiles and develops a systematic approach for profile building. Profiles are built using readily available data centre workload logs. This allows the improved profile-based assignment framework to achieve near-optimal assignment solutions without unduly increasing the computational effort. The theory of utilising application, VM and PM profiles in terms of energy-efficient application management is novel and as yet has remained unexplored. The following aspects of profiling have been discussed: 1) Building of more realistic VM and PM profiles by using real data centre workload trace, and building of application profile

by using the workload model designed by Lublin and Feitelson; 2) An improved assignment framework with a penalty-based optimization model and constraints of memory in addition to CPU; and 3) A Penalty-based Profile Matching Algorithm for near-optimal solution by using some heuristics. Comprehensive experimental studies demonstrate that profile-based application assignment approach is up to 22% more energy-efficient with steady execution times when compared to benchmark General and Workload History assignment approaches.

Profiles have been established as an efficient application assignment approach in this Chapter. Therefore, the next few chapters discuss the use of profiles for architecture development of profile-based system involving: 1) Static Assignment; 2) Dynamic Assignment; and 3) Consolidation. Chapter 4 addresses the Static Assignment Problem and discusses implementation of Profiles for static application assignment to VM as our second contribution.

Chapter 4

Repairing Genetic Algorithm

This Chapter is dedicated to the second research problem of the thesis: Static Assignment Problem. In Section 2.6, it has been indicated that the problem size (solution space) is too large for the application assignment problem. Therefore, a heuristic-based algorithm is required to solve the problem. This thesis uses Genetic Algorithm (GA) as the heuristic due to its ability of providing a feasible assignment solution on termination of the algorithm at any time [Bajpai and Kumar, 2010]. That is, the GA provides a feasible solution even on interruption of the algorithm. The GA is modified to improve solutions by designing a Repairing Genetic Algorithm (RGA) to solve the large-scale optimization problem. It enhances the GA by incorporating two components: 1) the Longest Cloudlet Fastest Processor (LCFP) and; 2) an Infeasible-solution Repairing Procedure (IRP). The LCFP generates initial population to minimize makespan and maximize resource utilization. The IRP converts infeasible chromosomes into feasible application assignment solutions. The RGA is integrated with First Fit Decreasing (FFD) VM placement policy to form a complete energy management solution. Experiments are conducted to demonstrate the effectiveness of the presented approach, e.g., 23% less energy consumption and 43% more resource utilization in comparison with GA under investigated scenarios. Some of the work presented in this Chapter has been published in our preliminary work on solving the profile-based application assignment problem with simple GA [Vasudevan et al., 2015].

The Chapter is organized as follows. Table 4.1 lists the notations used throughout this Chapter. Section 4.1 describes and formulates the static profile-based application assignment problem. Some case studies on using Genetic Algorithm for application assignment is presented

in Section 4.2. The repairing genetic algorithm (RGA) is developed in Section 4.3. This is followed by experimental studies in Section 4.4 to demonstrate our approach. Finally, Section 4.5 summarizes the Chapter.

Table 4.1: Description of notations used in Chapter 4.

Notation	Description
A_i	Application, $i \in I = \{1, \cdots, n\}$
β_1, β_2	Coefficients in fitness function $F(X)$
C_{ij}	Energy cost of A_i, $i \in I$, assigned to V_j, $j \in J$
$F(obj), F(X)$	Objective function, and fitness function, respectively
I, J, K	Integer sets $I = \{1, \cdots, n\}, J = \{1, \cdots, m\}, K = \{1, \cdots, l\}$
$IC(i)$	Instruction Count of A_i, $i \in I$
$IC_v(j)$	Number of instructions to be executed on V_j
M_j^{max}	Memory Capacity (MB) of V_j, $j \in J$
$M_r(i)$	Requested Memory (KB) of A_i, $i \in I$
$MIPS(j)$	MIPS rate of V_j, $j \in J$
n, m, l	Total numbers of applications, VMs and PMs, respectively
P	Total power consumption of the data centre
$P_{min},\ P_{max}$	Idle & peak server power consumption
PUE	Power Usage Efficiency of the data centre
S_k	Physical Machine (PM) or Server, $k \in K = \{1, \cdots, l\}$
$t_c(j)$	Completion time of all applications on V_j, $j \in J$
$t_e(i)$	Execution time of A_i, $i \in I$
U_c^{avg}	Average CPU utilisation of all virtual machines
V_j	Virtual Machine (VM), $j \in J = \{1, \cdots, m\}$
$U_c^{tot}(j),\ M_j^{tot}$	Total resource requirements of all A_i on V_j
x_{ij}	Binary assignment decision variable

4.1 Static Assignment Problem Formulation

The static assignment research problem is to design a profile-based application assignment to Virtual Machine (VM) framework. The objective of the assignment is to reduce energy consumption whilst maintaining high performance levels. The proposed solution is a profile-based Repairing Genetic Algorithm (RGA). To form a complete solution with integration of our profile-based RGA, a three-layer energy management strategy is implemented. It is developed with the following components:

1) **Application Assignment** implements the proposed profile-based RGA; and

2) **VM Placement** implements the First-Fit Decreasing (FFD) algorithm

Application, VM and server profiles are built from workload traces of a real data centre. The workload data include CPU and memory utilisation recorded multiple times in 60 minute intervals along with energy consumption in 5 minute intervals.

The RGA utilizes profiles to obtain data related to energy, CPU, memory availability and requirements in order to evaluate the following formulations. Unlike other approaches, profiling helps estimate energy cost prior to assignment by collecting and reusing data. The data includes instruction count requirement of a profiled application, the pre-set VM CPU capacity and the server power usage range. The RGA exploits a Longest Cloudlet Fastest Processor (LCFP) generated initial population to improve the makespan of the allocation. It also incorporates an infeasible-solution repairing procedure to re-assign the applications that violate the constraints. This converts infeasible solutions to feasible ones.

The application assignment problem is formulated as a linear optimization problem. The assignment of an application $A_i, i \in I$, onto a VM $V_j, j \in J$ is represented by a binary decision variable $x_{ij}, i \in I, j \in J$. It incurs an energy cost of C_{ij}, which is the product of the peak and idle server power ratio and the execution time of application A_i on VM V_j. The total number of instructions to execute application A_i is given by IC_i.

$$x_{ij} = \begin{cases} 1 & \text{if } A_i \text{ is allocated to } V_j;\ i \in I, j \in J, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

$$C_{ij} = \frac{P_{max}}{P_{min}} \cdot \frac{IC_i}{MIPS(j)}. \tag{4.2}$$

The assignment of a set of applications to VMs is given by a **constrained combinatorial optimisation model** as:

$$F(obj) = \min \sum_{j=1}^{m} \sum_{i=1}^{n} C_{ij} \cdot x_{ij} \tag{4.3}$$

$$\text{s.t.} \quad IC_v(j)/t_c(j) \leq MIPS(j),\ \forall j \in J; \tag{4.4}$$

$$\sum_{i=1}^{n} x_{ij} \cdot M_r(i) \leq M_j^{max},\ \forall j \in J; \tag{4.5}$$

$$\sum_{j=1}^{m} x_{ij} = 1,\ \forall i \in I; \tag{4.6}$$

$$x_{ij} = 0 \text{ or } 1,\ \forall i \in I, j \in J. \tag{4.7}$$

The constraints in Equations (4.4) and (4.5) ensure that the allocated resources are within the total capacity of the VM. Constraint in Equation (4.6) restricts an application from running on more than one VM. The binary constraint of the allocation decision variable x_{ij} is given by Equation (4.7). $IC_v(j)$, the number of instructions to be executed on $V_j, j \in J$ is calculated as:

$$IC_v(j) = \sum_{i=1}^{n} x_{ij} \cdot IC_i. \tag{4.8}$$

Makespan is the total length of time required for all applications in a VM to finish processing. The makespan of the assignment is calculated as:

$$makespan = max(t_c(j)) \; ; \; j \in J \tag{4.9}$$

$$t_c(j) = t_e(i) + CPUReadyTime \; ; \; i \in I, j \in J \tag{4.10}$$

where $t_c(j)$ is an iterative variable representing the total time required to execute all applications assigned to the VM, $V_j, j \in J$. CPU ready time is the time at which the last application of $V_j, j \in J$ finishes processing. The execution time of the next application $A_i, i \in I$ is represented by $t_e(i)$.

Average CPU utilization of all VMs across the data centre for the time interval under consideration is given by:

$$U_c^{avg} = \frac{1}{m} \sum_{j=1}^{m} \frac{IC_v(j)}{t_c(j)} \cdot \frac{1}{MIPS(j)} \tag{4.11}$$

The **total power consumption** associated with a data centre is:

$$P = l \cdot [P_{min} + (PUE - 1)P_{max} + (P_{max} - P_{min})U_c^{avg}], \tag{4.12}$$

where P_{max} and P_{min} represent the power consumed at the maximum and idle server utilization, respectively. The Power Usage Efficiency is represented by PUE. l represents the total number of active servers in the data centre.

The profile-based application assignment problem has been formally formulated as a constrained optimization problem. The size of this problem is generally large with multiple constraints and consequently the problem is NP-hard. The following Section conducts some case

studies on solving the optimization problem with genetic algorithm. This is then improved to develop a Repairing Genetic Algorithm (RGA).

4.2 Genetic Algorithm Case Studies

The static assignment problem is a combinatorial optimization problem, which is NP-hard. A simple steady-state genetic algorithm (GA) is first used to solve the problem prior to developing Repairing Genetic Algorithm (RGA). The GA case studies test/demonstrate scalability, energy- and resource-efficiency of utilizing evolutionary computing to solve the profile-based application assignment problem.

In our GA case studies, a data centre consisting of upto 2000 VMs is considered. Six different scenarios are investigated where the number of applications ranges from 500 to 5000 with corresponding number of VMs:

Scenario	**1**	**2**	**3**	**4**	**5**	**6**
VMs	100	400	800	1200	1600	2000
Applications	500	1000	2000	3000	4000	5000

The implementation of GA is carried out with a pre-set population size of 200 individuals in each generation. The termination condition is reached when there is no change in the average and maximum fitness values of strings for 10 generations. The number of maximum generations is set to be 200. The probabilities for crossover and mutation are configured to be 0.75 and 0.02, respectively.

Scalability. The high scalability of the GA is established by solving the assignment problem for up to 2000 VMs and 5000 applications. Figure 4.1 displays the algorithm solution time with respect to the increasing problem size. As the test problem size $[m * n]$ increases, the solution time of the GA increases linearly.

Energy Consumption The energy consumption of the GA is compared with that of a greedy algorithm. The GA is stochastic in nature. The quality of solutions in terms of energy consumption are assessed by using GA to solve 30 configurations of each of the test problems as shown in Figure 4.2. The resulting mean of energy consumption and solution times is given in

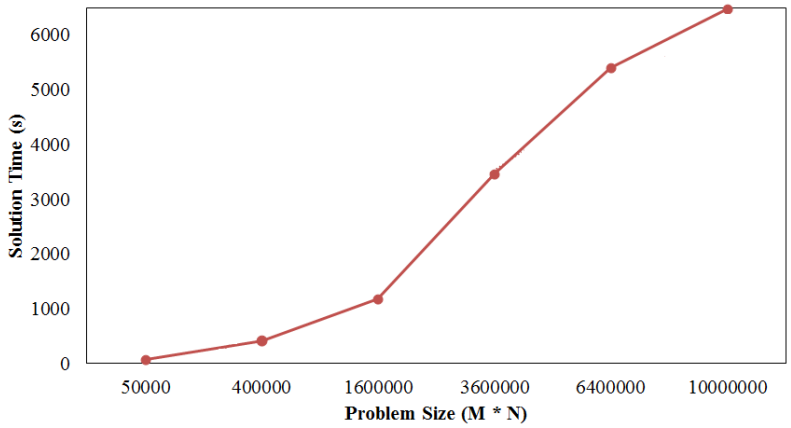


Figure 4.1: Scalability of the Genetic Algorithm.

Table 4.2. According to the results, the GA produces 16% to 32% better solutions in terms of energy consumption than the greedy algorithm. Although the solution times are higher compared to the greedy approach for the increasing number of applications, the GA maintains an efficient trade-off with energy consumption.

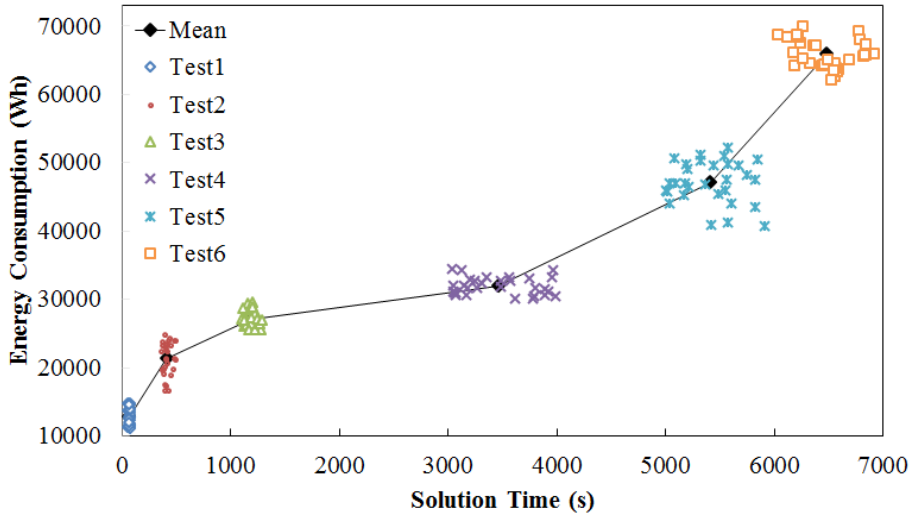


Figure 4.2: GA energy consumption and solution time for 30 configurations of each test problem set.

A paired t-test is conducted for the two independent means provided by the GA and greedy algorithm for each of the six test problems. The null hypothesis is that there is no difference between the GA and greedy energy consumption means. The confidence interval is set at 95% and a two-tailed hypothesis is assumed. The t-stat values are recorded in Table 4.2 and the p-values are all significantly less than 0.05. The results show that the difference between the means are significant and thus, the null hypothesis is rejected.

Table 4.2: Energy and solution time performance of GA vs. Greedy (Energy unit: Wh; time unit: sec).

Genetic Algorithm			Greedy			T-Test			
Energy	**SD**	**Time**	**SD**	**Energy**	**Time**	**t-stat**	**std. err**	**DF**	**crit 2-tail**
12878.47	1227.90	69	7.53	15017.28	2	-9.54	224.18	29	2.045
21379.27	2404.85	412	38.98	28234.01	6	-15.61	439.06	29	2.045
27113.47	1086.97	1189	50.39	33482.86	9	-32.091	198.45	29	2.045
32001.33	1264.83	3459	341.61	38416.58	18	-27.778	230.92	29	2.045
47149.83	3107.03	5412	276.58	56115.70	27	-15.81	567.26	29	2.045
65904.90	2104.70	6484	250.01	78025.54	40	-31.54	384.26	29	2.045

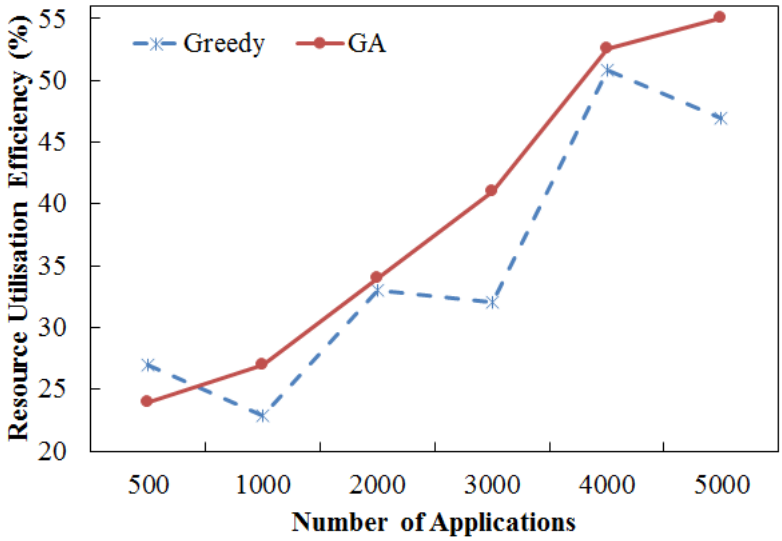


Figure 4.3: Resource utilization efficiency of GA vs. Greedy.

Resource Utilization. As shown in Figure 4.3, the average sum of CPU and memory utilization efficiency of the penalty-based GA is 3% to 22% more efficient when compared to the greedy approach. Also the variance of the CPU utilization of the GA (0.56) is lower than that of the Greedy algorithm (0.90). This indicates the GA is more consistent in resource allocation.

Therefore, the energy- and resource-efficiency of utilizing GA for the profile-based application assignment have been established. The GA will now be improved with the addition of LCFP-generated initial population and an infeasible-solution repairing procedure. The following Section describes the development of the RGA.

4.3 Repairing Genetic Algorithm

This Section aims to solve the the linear optimization model (Equation 4.3) for the profile-based application assignment problem in data centres. The previous Section explored solving the model using a simple steady-state GA. However, due to the large data set of the optimization problem, the performance of the classic GA is degraded due to imperfect, slow or no convergence [Zhu et al., 2011]. Therefore, this Section improves the GA by developing a Repairing Genetic Algorithm (RGA) to solve the static assignment problem. It aims to minimize energy consumption and makespan whilst maximizing resource utilization.

4.3.1 High-Level Description of RGA

A high-level description of the RGA is given in Algorithm 8, which is self-explained. In comparison with a simple GA, RGA incorporates Longest Cloudlet Fastest Processor (LCFP) generated initial population (Lines 1 and 2) and an Infeasible-solution Repairing Procedure (IRP) (Lines 8 to 10). This will be further discussed in detail in the following subsections.

4.3.2 LCFP-Generated Initial Population

The Longest Cloudlet Fastest Processor (LCFP) heuristic assigns the longest application to the fastest processing VM. It considers the computational complexity of applications and the computing power of processors to generate an initial population composed of a set of chromosomes. This ensures that the lengthier applications finish quickly thereby minimizing makespan. For faster convergence and better solutions, the initial population can be better chosen through heuristics. Conversely, in a steady-state GA, the initial population is randomly generated as in other general GA methods.

The LCFP heuristics pursue three main steps. The steps are:

1. Sort applications $A_i, i \in I$ in descending order of execution time;

2. Sort VMs $V_j, j \in J$ in descending order of processing power (MIPS); and

3. Pack sorted applications into fastest processing VM

Algorithm 8: Repairing Genetic Algorithm

```
Find output of solutions generated by LCFP;
Initialize population with LCFP output;
Evaluate fitness of each candidate chromosome;
while Termination condition is not satisfied do
    for Each Generation do
        for Each chromosome do
            Evaluate fitness;
            if Chromosome is infeasible then
                Apply Infeasible-solution Repairing Procedure;
                Evaluate Fitness;
            Parents selected using Roulette Wheel Selection;
        for Parent chromosomes do
            Apply uniform crossover as per P_c;
            Mutate resulting offspring as per P_m;
            Offspring chromosomes generated;
        for Offspring chromosomes do
            Evaluate fitness of new candidates;
            Replace low-fitness chromosomes with better offspring;
        for Each chromosome do
            Select chromosomes for next generation;
Output the best fit chromosome as solution;
```

The initial population undergoes the encoding, selection, mutation and crossover as seen from Figure 4.4. Once the LCFP-generated initial population is created, parent chromosomes are selected from the pool. Figure 4.4 displays a set of applications $\{A_1, A_2, \cdots, A_9\}$ and VMs $\{V_1, V_2, V_3\}$. Parent Chromosome 1 is selected from the pool using roulette wheel selection such that VM V_1 hosts three applications A_1, A_5, A_8; V_2 hosts A_2, A_6, A_7, A_9; and V_3 hosts A_3, A_4. Similarly, Parent Chromosome 2 is also selected. Value encoding is used to represent Parent Chromosome 1 and 2 such that each box in the Figure represents an application and the number inside represents the host VM ID (gene). For example, A_1 is assigned to VM V_1 in Parent Chromosome 1 and V_3 in Parent Chromosome 2. Keep in mind that the parent chromosomes are possible assignment solutions and not the final assignment solution. A Binary Crossover Mask is randomly generated and used to create crossover offspring. If the mask is 1, then the corresponding genes in that specific position of both chromosomes are swapped with each other. Crossover swaps the genes between two separate parent chromosomes. Whereas, Mutation randomly selects two exact positions (A_4 and A_8 in Figure 4.4) and swaps the genes within the chromosome.

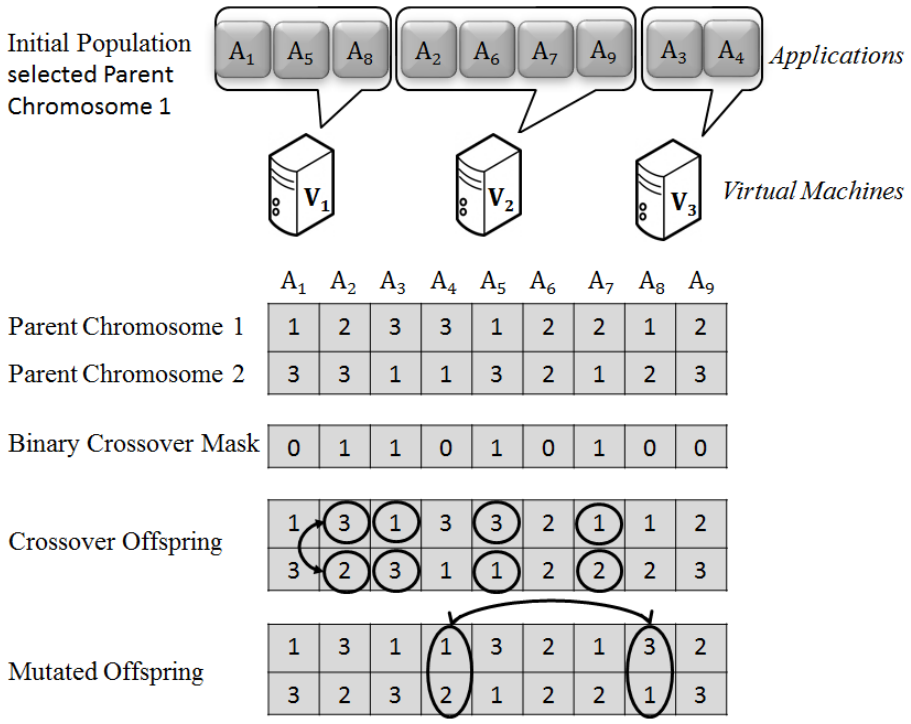


Figure 4.4: Value encoding, uniform crossover using binary mask and mutation by selection and exchange of two genes

The working of the genetic operators to produce a new population consisting of offspring chromosomes is described as follows. Algorithm 9 presents the pseudocode of the working of the genetic operators.

1. **Encoding.** The chromosomes are represented by value encoding. This allows individual genes to be represented as positive integers derived from the actual VM numbers to build chromosomes. Each chromosome consists of $|n|$ genes. Each gene has a value ranging from 1 to $|m|$.

2. **Selection.** The selection operator (Lines 1 to 11 in Algorithm 9) identifies and assigns possible solution chromosomes as parents. The parent chromosomes are selected from the mating pool with the help of roulette wheel selection, which is a fitness-proportionate selection technique. The greater the fitness value of a chromosome, the more probability of being selected as a parent. The parents chromosomes are then used to generate a consequent population.

3. **Crossover.** Uniform crossover (Lines 12 to 23 in Algorithm 9) is applied to the parent solutions to produce the offspring solutions. A binary crossover mask is randomly generated for each parent, where 1 and 0 indicate that the gene will be copied from the first

Algorithm 9: Genetic Operators - Crossover, Mutation and Roulette Wheel Selection

1 **Roulette Wheel Selection**;
2 Evaluate constant $Fitness_{sum}$ = total fitness of all chromosomes in the population;
3 Let variable $itSum$ represent the iterative fitness sum;
4 Let variable Ch represent a chromosome;
5 Generate random number $G \in [0, 1]$;
6 Set $itSum \leftarrow 0$;
7 **for** *Each Chromosome* **do**
8 Evaluate Probability of Selection $prob_{Ch} \leftarrow fitness(Ch)/Fitness_{sum}$;
9 $itSum \leftarrow itSum + prob_{Ch}$;
10 **if** $itSum < G$ **then**
11 Select chromosome Ch to be parent p;
12 **Crossover**;
13 Set length of chromosome, λ to number of applications to be allocated;
14 Parents : $p(1) = g_1^1, g_2^1, ..., g_\lambda^1$ and $p(2) = g_1^2, g_2^2, ..., g_\lambda^2$;
15 Let Offspring : $O(3) = g_1^3, g_2^3, ..., g_\lambda^3$ and $O(4) = g_1^4, g_2^4, ..., g_\lambda^4$;
16 **for** *Length of chromosome* **do**
17 Assign 0 or 1 randomly to individual genes of Mask;
18 **if** *Mask is 0* **then**
19 $g_\lambda^3 \leftarrow g_\lambda^2$;
20 $g_\lambda^4 \leftarrow g_\lambda^1$;
21 **else**
22 $g_\lambda^3 \leftarrow g_\lambda^1$;
23 $g_\lambda^4 \leftarrow g_\lambda^2$;
24 **Mutation**;
25 Randomly select two numbers between 1 and λ and assign to A and B;
26 **for** *Each Child* **do**
27 $temp \leftarrow g_A^3$;
28 $g_A^3 \leftarrow g_B^4$;
29 $g_B^4 \leftarrow temp$;

and second parents, respectively.

4. **Mutation.** The mutation (Lines 24 to 29 in Algorithm 9) is carried out by selecting and exchanging two genes from the offspring chromosomes. The mutation probability is set to a low number, in order to control the search space.

5. **Termination Condition.** The termination condition specifies that the cycle is repeated for each generation until a maximum number of generations is reached or an individual is found which adequately solves the problem. Every iteration of the algorithm creates a population consisting of a set of chromosomes,which represent a possible assignment solution.

The fitness function determines the quality of the solution when compared to an optimal solution. A lower energy cost and higher resource utilization result in a higher solution fitness. Feasible solutions have a positive fitness value, whereas infeasible solutions incur a negative fitness. The fitness function is derived as:

$$F(X) = \beta_1 \cdot \bar{F_{obj}} - \beta_2 \cdot \frac{1}{m} \cdot \sum_{j=1}^{m} \left(F_j^{cpu} + F_j^{mem} \right). \tag{4.13}$$

The weights $[\beta_1, \beta_2]$ associated with the fitness function is currently set to $[2, 1]$. The multiplicative inverse of the objective function discussed in Equation (4.3) is represented by $\bar{F_{obj}}$. In order to normalise and scale the objective function $\bar{F_{obj}}$ to a range of $[1, 10]$, we use:

$$\bar{F_{obj}} = \frac{F_{worst} - F_{obj}}{F_{worst} - F^{\star}} \cdot \frac{F^{\star}}{F_{obj}} \cdot range + 1, \tag{4.14}$$

where the $range = 9$. The best (minimized) and worst objective functions are represented by $F^{\star}$ and F_{worst}, respectively. The functions to ensure higher CPU and memory utilizations by penalizing constraint violations is given by:

$$F_j^{cpu} = \begin{cases} 0, & \text{if } U_c^{avg} = 1 \\ MIPS(j)/IC_j, & \text{if } 0 < U_{avg} < 1 \\ 2 & \text{if } U_c^{avg} = 0 \end{cases} \tag{4.15}$$

$$F_j^{mem} = \begin{cases} 2 \cdot (1 - 1/\gamma), & \text{if } \gamma \geqslant 1 \\ 2, & \text{otherwise} \end{cases}, \text{ where } \gamma = \frac{M_j^{max}}{\sum_{i=1}^{n} x_{ij} \cdot M_r(i)} \tag{4.16}$$

4.3.3 Infeasible-solution Repairing Procedure

The RGA incorporates an Infeasible-solution Repairing Procedure (IRP) to convert infeasible chromosomes to feasible ones. An infeasible allocation of applications to VMs is characterised by a negative fitness as a result of high penalty due to CPU and memory constraint violations. The applications assigned to the infeasible VMs are re-assigned to other VMs

until the violations are resolved. Consequently, the fitness becomes a positive value. To the best of our knowledge, a solution repairing procedure has only been previously applied for VM management problems and has not been implemented for application assignment to VM problems.

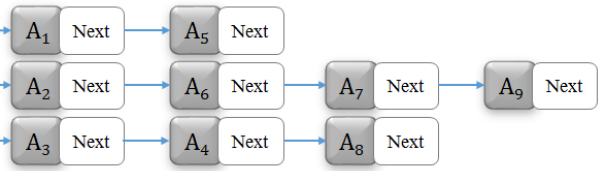
Each VM in a chromosome is linked to a data structure. As shown in Figure 4.5, the data structure consists of a $violation$ indicator, total CPU and memory requirements of the applications allocated to the VM ($U_c^{tot}(j)$ & M_j^{tot}), the VM's CPU and memory capacity ($MIPS(j)$ and M_j^{max}), and a pointer to a linked list to indicate the application id, CPU and memory requirement of each application allocated to the VM. The violation indicator is set to 0 if the application allocations do not violate the CPU and memory constraints, and is set to 1 if otherwise.

$$violation = \begin{cases} 0 & \text{if resource constraints are not violated} \\ 1 & \text{otherwise.} \end{cases} \tag{4.17}$$

Total CPU and memory requirements of the applications on VM V_j is derived by:

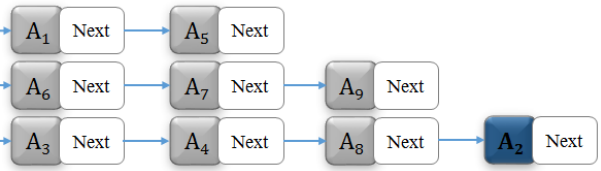
$$U_c^{tot}(j) = \frac{IC_v(j)}{t_c(j)},\ M_j^{tot} = \sum_{i=1}^{n} x_{ij} \cdot M_r(i). \tag{4.18}$$

viol	VM ID	Total Used		Max Capacity		pointer
		CPU (MIPS)	Mem (Bytes)	CPU (MIPS)	Mem (Bytes)	
0	1	900	1000	1000	2000	
1	2	1750	3010	1500	3000	
0	3	2000	3000	3000	5000	



a) Before applying IRP

viol	VM ID	Total Used		Max Capacity		pointer
		CPU (MIPS)	Mem (Bytes)	CPU (MIPS)	Mem (Bytes)	
0	1	900	1000	1000	2000	
0	2	1500	3000	1500	3000	
0	3	2250	3010	3000	5000	



b) After applying IRP

Figure 4.5: Data structure used in the Infeasible-solution Repairing Procedure.

With the help of the violation indicator, the data structure is used to identify the VMs that

violate the constraint. The constraint violation makes the chromosome infeasible. Once the VM $V_j, j \in J$ is identified, the CPU and memory availability of the next VM $V_{j'}, j' \in J$ is calculated. If the resource availability is greater than the resource requirement of the first application in the linked list of $V_j, j \in J$, the application is re-allocated to the new VM $V_{j'}, j' \in J$. The process is repeated until the violations are fixed and the chromosome is feasible. For example, in Figure 4.5 (a) VM V_2 hosting four applications is violating resource constraints. Therefore, IRP is applied and the first application A_2 of VM V_2 is re-assigned to VM V_3. This solves the violation which returns to 0 (Figure 4.5 (b)). The working of the infeasible-solution repairing procedure is presented in Algorithm 10, which is self-explained.

Algorithm 10: Infeasible-solution Repairing Procedure

for *j = 1 to Total number of VMs* **do**
  if $V_j.violation = 1$ **then**
    $select = V_j.pointer$;
    $V_j.pointer = V_j.pointer.next$;
    $fixed = false$;
    $j' = j + 1$;
    $avail_{CPU} = V[j'].U_c^{tot}(j) - V[j'].CPU_{j'}^{max}$;
    $avail_{mem} = V[j'].M_j^{tot} - V[j'].MEM_{j'}^{max}$;
    while $j' \neq j$ **do**
      if $(avail_{CPU} \geq select.a^{CPU})$& $(avail_{mem} \geq select.a^{mem})$ **then**
        $select.next = V_{j'}.pointer$;
        $V_{j'}.pointer = select$;
        $V_{j'}.U_c^{tot}(j) += select.a^{CPU}$;
        $V_{j'}.M_j^{tot} += select.a^{mem}$;
        $fixed = true$;
      Set $j' = j' + 1$;

After the applications are allocated to the VMs using RGA, we have servers $\{S_1, \cdots, S_k, \cdots, S_l\}$ and VMs $\{V_1, \cdots, V_j, \cdots, V_m\}$. The FFD algorithm packs the VMs using as few servers as possible. The FFD algorithm sorts the servers in decreasing order of resource capacity. Each active VM is placed onto the first server with adequate space remaining. All active VMs are eventually packed onto PM servers.

4.4 Experimental Studies

This Section conducts experiments to demonstrate energy-efficiency and quality of solutions of profile-based application assignment using RGA. It begins with an introduction into experimental design. This is followed by a discussion of evaluation criteria. Then, experimental results are presented.

Experimental Design

Profiles are created for every application, VM and PM from real data centre workload logs. For building the profiles, the workload logs are collected over a period of seven days (the 12th to 19th of May, 2014). They include information about CPU, memory and energy utilizations. The length of each application is determined by the Instruction Count (IC). The computing capacity of each VM is in Million Instructions Per Second (MIPS). The application and VM parameter settings are shown below.

Table 4.3 depicts application and VM parameter settings.

Table 4.3: Parameter settings for applications and VMs.

Parameter	Value
IC	$[5, 10] \times 10^9$ instr
IPS	$[1, 2] \times 10^9$ instr/sec
Memory	$[1000, 5000]$ Bytes
P_{max}	350 W
P_{min}	150 W
PUE	2

In our experiments, a data centre with 100 PM servers that hosts upto 1000 VMs is considered. For the evaluation, 11 different problem test sets are considered where the number of applications ranges from 20 to 5000 with corresponding number of VMs as seen in Table 4.4. For the first five test sets, the number of VMs is kept constant while the number of applications varies.

The three-layer energy management system integrating profile-based application assignment with VM placement is implemented for evaluation of overall energy consumption. At the application management layer, profile-based application assignment is implemented using RGA. It incorporates LCFP-generated initial population and IRP. RGA is carried out with a

Table 4.4: Problem test sets for Chapter 4 experiments.

Test set	VMs	Applications
1	10	20
2	10	40
3	10	60
4	10	80
5	10	100
6	50	200
7	100	500
8	250	1500
9	500	2500
10	750	3500
11	1000	5000

pre-set population size of 200 individuals in each generation. It is terminated when there is no change in the average and maximum fitness values of strings for 10 generations. The number of the maximum generations is set to be 200. The probabilities for crossover and mutation operations are configured to be 0.75 and 0.02, respectively. At the VM management layer, the widely used First-Fit Decreasing (FFD) algorithm is implemented for VM placement to PM servers.

Evaluation Criteria

In order to evaluate the quality and efficiency of the solutions for profile-based application assignment to VMs, RGA presented in this Chapter is compared with the steady-state GA [Portaluri et al., 2014] which is assumed as a benchmark (there is no actual benchmark available for application assignment and further evaluations on the RGA with existing assignment methods are conducted in the following Chapter 5). Also, FFD is implemented for VM placement to PMs, forming RGA-FFD for RGA, and GA-FFD for GA, respectively. The evaluation criteria for testing RGA include the following:

1. Scalability
2. Energy efficiency and computing efficiency
   (a) Energy consumption
   (b) Solution time
   (c) Statistical T-Test analysis

3. Quality of solutions

   (a) VM resource utilization

   (b) Speed of convergence

   (c) Makespan performance with respect to initial population

4.4.1 Scalability of RGA

The high scalability of the RGA is demonstrated through solving the static assignment problem for a problem size ranging from 200 to $5,000,000$. Figure 4.6 shows the algorithm solution time with respect to the problem size. As the test problem size $[m * n]$ increases, the solution time of RGA increases linearly. A nearly linear increase in the solution time with respect to the problem size well characterizes RGA's good scalability. For test sets 1 to 5 where the number of VMs is a constant of 10, the elapsed times for algorithm solution are small.

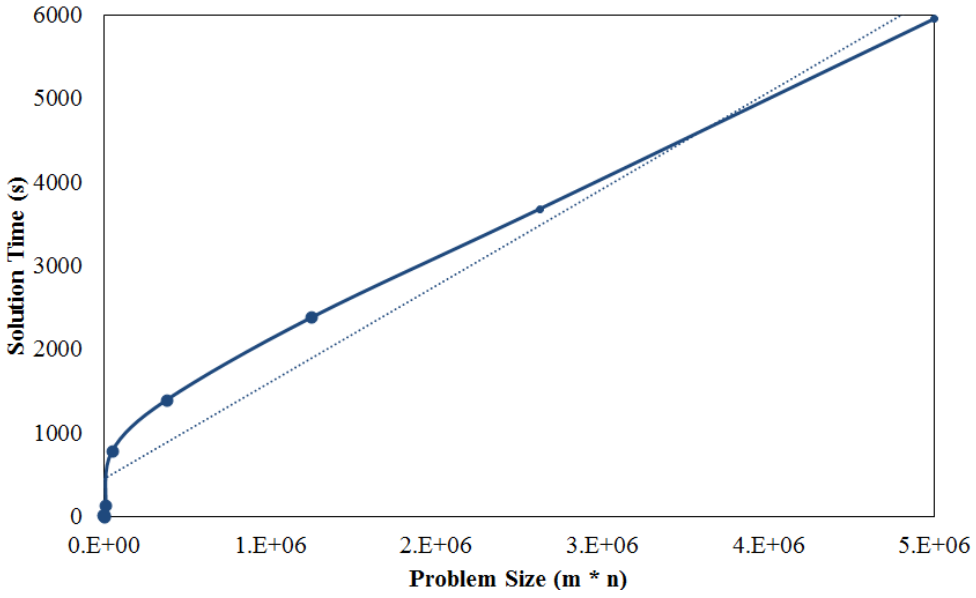


Figure 4.6: Scalability of the Repairing Genetic Algorithm.

4.4.2 Energy Efficiency and Computing Efficiency

In order to calculate the actual energy consumption and computing time for both genetic algorithm (GA) and RGA, FFD is implemented as the policy for VM placement to PMs. The results of energy efficiency and computing efficiency are tabulated in Table 4.5.

Table 4.5: Energy efficiency and computing efficiency of GA and RGA incorporating with FFD (SD: standard deviation). For each of the test sets, the results are derived from 30 runs.

Test set	VMs	PMs Required	Total energy of all active servers (Wh)				Daily energy (KWh)		Solution time (sec)	
			GA	SD	RGA	SD	GA	RGA	GA	RGA
1	10	1	219	35.79	194	13.85	5.26	4.66	0.8	0.6
2	10	1	247	41.69	222	14.48	5.92	5.32	2.8	2.3
3	10	2	409	42.06	315	28.76	9.82	7.57	6.0	4.7
4	10	2	425	46.26	337	30.12	10.19	8.09	6.4	5.2
5	10	2	432	50.47	398	34.90	10.37	9.56	15	12
6	50	5	1422	45.28	1308	39.41	34.12	31.38	162	134
7	100	11	2737	32.38	2249	38.70	65.70	53.97	1298	804
8	250	27	7027	69.87	5875	45.12	168.64	141.00	2293	1406
9	500	62	17127	63.91	13407	52.45	411.04	321.77	5127	2387
10	750	75	21093	53.94	17461	35.94	506.24	419.07	8004	3689
11	1000	100	27591	57.74	24605	58.35	662.19	590.52	13070	5999

The first observation from Table 4.5 is that RGA-FFD gives smaller energy consumption than GA-FFD for all test sets. The energy savings of RGA-FFD in comparison with GA-FFD are from 7.8% up to 23% for the considered test sets. For Test Set 11, which represents a realistic size of a small data centre, both GA and RGA are respectively used to allocate 5000 applications to 1000 VMs, which are then placed by FFD to 100 active PMs. The resulting daily energy consumption of the data centre is 662.59 KWh for GA-FFD and 590.52 KWh for RGA-FFD, indicating an about 72 KWh daily energy saving.

The second observation from Table 4.5 is that the energy consumption results from RGA-FFD show smaller average standard deviations than those from GA-FFD. This implies that RGA is more stable than GA for deriving the results. This conclusion is drawn from 30 runs for each the test sets.

For computing efficiency, as the problem size increases, GA-FFD takes up to twice as much time as RGA-FFD does to solve the problem. This is clearly shown in average Solution Time in Table 4.5. This means that RGA converges faster with better solutions than GA.

To demonstrate the confidence level of the experimental results, a paired t-test is conducted for the two independent methods of GA and RGA for each test set. As GA and RGA are stochastic in nature, both of them are individually run 30 times for each Test set. The null

hypothesis is that there is no difference between GA and RGA methods. The confidence interval is set at 95% and a two-tailed hypothesis is assumed. The t-stat values are recorded in Table 4.6. The two-tailed P-value is less than 0.0001 and is extremely statistically significant. The results show that the difference between the two methods are significant, and thus the null hypothesis is rejected.

Table 4.6: T-test of the solutions by GA and RGA.

Test set	T-Value	std. error	t-crit	df
1	-3.66	6.866	2.045	29
2	-3.15	7.888	2.045	29
3	-9.98	9.416	2.045	29
4	-8.04	10.901	2.045	29
5	-2.43	13.960	2.045	29
6	-11.25	10.139	2.045	29
7	-29.12	9.943	2.045	29
8	-56.87	16.726	2.045	29
9	-62.08	12.753	2.045	29
10	-60.11	11.841	2.045	29
11	-54.17	17.145	2.045	29

4.4.3 Quality of Solutions

The quality of solutions is determined by measuring resource utilization, convergence, and makespan performance due to LCFP-generated initial population.

VM Resource Utilization. The results of VM resource utilization for both GA and RGA are shown in Figure 4.7. It is clearly seen from Figure 4.7 that RGA uses VM resources more efficiently than GA. Overall, RGA gives solutions that are 10.42% to 42.86% more resource efficient than GA. The highest average VM resource utilization achieved by RGA is 70%, while this metric is only 56% for GA. The initial steep incline in Figure 4.7 is a result of keeping the number of VMs constant to 10 whilst increasing the number of applications from 20 to 100 for Test sets 1 to 5. This implies that for Test set 5, the VM resources are under maximum utilization due to an approximate average of ten applications assigned per VM. The drop in the figure indicates the increase in the number of VMs in Test set 6. The resource utilization increases linearly with the gradual increase of both the number of VMs and applications.

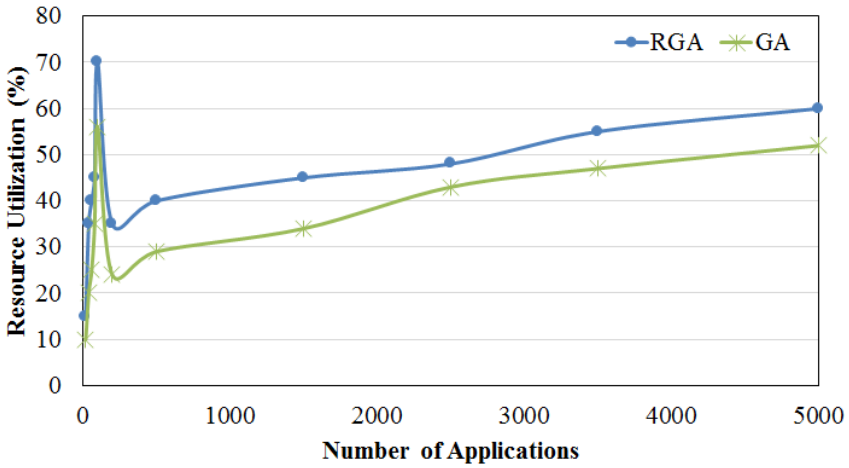


Figure 4.7: Resource utilisation efficiency of RGA vs. GA.

Convergence. Table 4.7 tabulates the best results in terms of energy and the corresponding genetic iteration for both GA and RGA. It is clearly seen from Table 4.7 that RGA gives better solutions with much fewer iterations. With the increase in the problem size from Test Sets 1 to 6, the numbers of iterations from RGA are over three quarters fewer that those from GA. From test sets 7 to 11, GA has reached the pre-set maximum number of iterations of 200, while RGA achieves better solutions with less than half of the pre-set number of iterations. For Test set 11, RGA uses 93 iterations to derive a better solution than GA with 200 iterations. All these results show faster convergence of RGA than that of GA.

Table 4.7: Comparisons of GA and RGA with regard to convergence.

Test	GA - Best Result		RGA - Best Result	
set	Energy (Wh)	Iteration	Energy (Wh)	Iteration
1	191	47	175	4
2	205	94	190	13
3	372	132	280	18
4	399	159	300	23
5	417	172	349	25
6	1398	197	1294	39
7	2719	200	2212	44
8	6998	200	5839	57
9	17098	200	13379	62
10	21038	200	17439	81
11	27554	200	24584	93

Makespan Performance due to Initial Population. Makespan is the maximum completion time of all applications allocated to a VM. In our preliminary work [Vasudevan et al., 2015], GA

is applied with random initial population. This Section will use random initial population and LCFP-generated initial population for RGA. The overall average makespan for GA with random initial population is denoted by GA-Rand. Similarly, use RGA-Rand and RGA-LCFP to denote RGA with random and LCFP-seeded initial population, respectively. Figure 4.8 depicts the makespan performance of GA-Rand, RGA-Rand and RGA-LCFP. With the increase of problem size, the steep initial incline of all three strategies represent the solutions of Test sets 1-5, where the number of VMs are constant whilst the number of applications varies from 20 to 100. It is seen from the figure that both RGA-Rand and RGA-LCFP provide better solutions than GA-Rand does. However, RGA-LCFP outperforms RGA-Rand in the sense that it generates better solution with faster convergence. This is because the LCFP strategy seeds the initial populations by assigning the longest application to the fastest processing (MIPS) VM. This in turn minimizes the makespan and maximizes the CPU utilization.

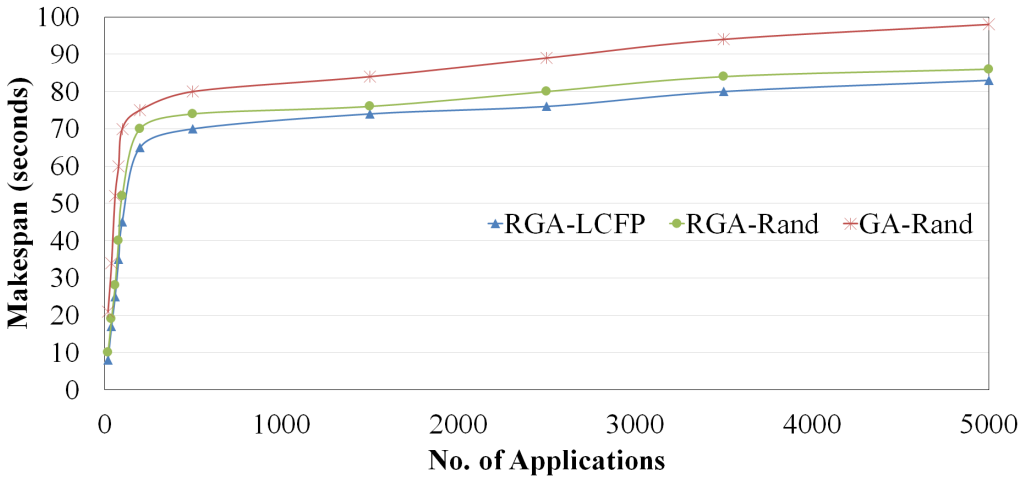


Figure 4.8: Makespan performance due to initial population.

4.5 Summary of the Chapter

This Chapter addresses the Static Assignment Problem of using profiles for energy- and performance-efficient application assignment to VMs. It makes the second contribution of our research: a Repairing Genetic Algorithm (RGA). The RGA implements the profile-based static application assignment framework. It improves the steady-state GA by incorporating the following two main components:

1. Longest Cloudlet Fastest Processor (LCFP) generated initial population for faster convergence and minimized VM makespan; and

2. Infeasible-solution Repairing Procedure (IRP) that convert infeasible solutions that violate resource usage constraints to feasible solutions. This is executed by re-assigning applications from a bad VM host to other VMs until the violations are null and the solution fitness is satisfactory.

The RGA is implemented at the top application management layer in the three-layer energy management. In addition, for a complete energy management system, the First Fit Decreasing (FFD) heuristic has been implemented at the middle VM management layer in the three-layer energy management. Experiments have been conducted to demonstrate the effectiveness and efficiency of the profile-based RGA. For the investigated scenarios, RGA has shown 23% less energy consumption and 43% more resource utilization in comparison with steady-state GA. The better solutions are achieved with faster convergence and shorter computing time than GA. Therefore, the profile-based RGA is a promising tool for energy-efficient application assignment to VMs in data centres.

The next Chapter 5 addresses the Dynamic Assignment Problem. It discusses implementation of Profiles for dynamic application assignment to VM as our third contribution.

Chapter 5

Profile-based Dynamic Application Assignment

This Chapter is dedicated to the third research problem of the thesis: the Dynamic Assignment Problem. To solve this problem, a profile-based dynamic application assignment framework is presented. The framework handles real-time applications and workload. The key issue of this problem is to plan a response to sudden changes in the workload such as VM failure and new applications. Experiments demonstrate the effectiveness of the dynamic approach to application management. The dynamic approach produces 48% better energy savings than existing application assignment approaches under investigated scenarios. It also performs better than the static application management approach with 10% higher resource utilization efficiency and lower degree of imbalance.

In the following sections, 1) a dynamic approach will be presented to deal with profile-based application assignment. It uses profiles derived from actual workload logs and implements the Repairing Genetic Algorithm (RGA, developed in Chapter 4); 2) The research problem will be formally formulated by considering fluctuations in real-time application arrivals, resource demands and VM availability; 3) Application finishing times will be estimated for satisfying deadline constraints and reducing waiting time and assignment overhead; 4) A real-time assignment strategy will be developed to address infrequent issues such as new/random applications or inoperative VMs; and 5) Actual energy savings will be demonstrated through the three-layer energy management of data centres as shown in Figure 1.1. The implemented VM placement policy is first-fit decreasing (FFD).

The remainder of this Chapter is organized as follows. The notations used throughout this Chapter are listed in Table 5.1. The dynamic research problem is described and formulated in

Section 5.1. A profile-based dynamic application assignment framework for real-time applications is presented in Section 5.2. Section 5.3 discusses the repairing genetic algorithm for dynamic application assignment solution. The dynamic application assignment framework and RGA are evaluated in Section 5.4. Finally, Section 5.5 summarizes the Chapter.

5.1 Dynamic Assignment Problem Formulation

Initially, the application, VM and server profiles have already been created off-line using data centre workload logs. Then, the profiles are expanded and updated online as needed in real-time.

5.1.1 Characterizing Application Dynamics

In our application assignment problem, multiple real-time applications are to be allocated to VMs. The application assignment is required to satisfy deadline, waiting time and performance constraints. Consider a set of real-time applications, $\{A_1, A_2, \cdots, A_i, \cdots, A_n\}$. The dynamic behaviour of an application is characterised by varying arrival times, real-time constraints and resource demands. Thus, the real-time application $A_i, 1 \leq i \leq n$ is characterized by the following parameters: 1) submission time $t_s(i)$; 2) deadline constraint $t_d(i)$; and 3) required resources such as cores $N_r(i)$, instruction count $IC(i)$ and memory $M_r(i)$. Therefore, the real-time application A_i is defined as

$$A_i = \langle t_s(i), t_d(i), N_r(i), IC(i), M_r(i) \rangle, i \in I = \{1, 2, \cdots, n\} \tag{5.1}$$

The application profiles incorporate the above information. They also calculate waiting times, the maximum execution times, periodicity and finishing times in order to improve the application assignment efficiency. The waiting time $t_w(i)$ is counted from the time instant $t_s(i)$ when the application arrives till the time instant at which the application is allocated. Once the application is placed on a VM with allocated resources, it executes for a maximum execution time of $\max t_e(i)$ for $i \in I$. The maximum execution time can be calculated by:

$$\max_{i \in I} t_e(i) = t_d(i) - t_s(i), i \in I \tag{5.2}$$

The length of the application $A_i, i \in I$ is represented by instruction count $IC(i)$, which is

Table 5.1: Description of notations used in Chapter 5.

Notation	Description
α_j	Ratio of power consumed at max to min util of host V_j
A_i	Application, $i \in I = \{1, \cdots, n\}$
β_1, β_2	Coefficients in fitness function $F(X)$
C_{ij}	Energy cost of A_i, $i \in I$, assigned to V_j, $j \in J$
CPI	Cycles Per Instruction
$f_{vc}^{max}(j)$	Max CPU Frequency (MHz) of V_j, $j \in J$
$f_{vc}^{used}(j)$	Used vCPU frequency (MHz) of V_j, $j \in J$
$f_c(k)$	Frequency of cores in S_k, $k \in K = \{1, \cdots, l\}$
$f_c^{total}(k)$	Total CPU frequency (MHz) of S_k, $k \in K$
$f_c^{used}(k)$ (%)	Total CPU Usage (%) of S_k, $k \in K$
$F(obj), F(X)$	Objective function, and fitness function, respectively
i, j, k	Subscripts or indices for applications, VMs and PMs, respectively
I, J, K	Integer sets $I = \{1, \cdots, n\}, J = \{1, \cdots, m\}, K = \{1, \cdots, l\}$
$IC(i)$	Instruction Count of A_i, $i \in I$
M_j^{max}	Memory Capacity (MB) of V_j, $j \in J$
M_j^{used}	Used Memory (KB) of V_j, $j \in J$
$M_r(i)$	Requested Memory (KB) of A_i, $i \in I$
$MIPS(j)$	MIPS rate of V_j, $j \in J$
n, m, l	Total numbers of applications, VMs and PMs, respectively
$N_c(k)$	Total number of cores in S_k, $k \in K$
$N_r(i)$	Requested cores for A_i, $i \in I$
$N_{vc}j$	Number of vCPUs of V_j, $j \in J$
P_k	Power consumed (Watts) of S_k, $k \in K$
P_k^{max}, P_k^{min}	Power at respect max and min utilizations of S_k, $k \in K$
S_k	Physical Machine (PM) or Server, $k \in K = \{1, \cdots, l\}$
$t_c(j)$	Completion time of all applications on V_j, $j \in J$
$t_d(i), t_e(i), t_f(i)$	Deadline, exec. time & finish time of A_i, $i \in I$, respectively
$t_s(i), t_w(i)$	Submission time and waiting time of A_i, $i \in I$, respectively
T_λ	Time slot ID, $\lambda \in \{1, 2, \cdots, \Lambda\}$
V_j	Virtual Machine (VM), $j \in J = \{1, \cdots, m\}$
x_{ij}	Binary assignment decision variable
$X, \hat{X}$	Assignment decision matrix and its estimation, respectively
y_{jk}	Binary VM-server host constant
Δt	Time interval

measured in million instructions. Periodicity is set to 1 if the application is regular to the data centre or 0 if it is not periodic. The application releases the VM resources and exits the VM on execution completion.

The finishing time of the application $A_i, i \in I$ can be estimated using the application profiles before the allocation for satisfaction of the deadline constraints.

$$\text{Estimated } t_f(i) = t_s(i) + \frac{IC(i)}{MIPS(j) \cdot N_r(i)} \tag{5.3}$$

5.1.2 Characterizing Virtual Machine Dynamics

A virtualized data centre consists of a set of PMs $\{S_1, \cdots, S_k, \cdots, S_l\}$ and a set of VMs $\{V_1, \cdots, V_j, \cdots, V_m\}$. From our collected information from a real medium-density data centre, the processing capabilities of the PMs and VMs, such as MIPS rate, CPU frequency, memory and storage are pre-configured. They are reviewed every 6-12 months and updated if necessary. Therefore, the maximum server resources available to VMs are considered to be constant in this research.

For a PM S_k, the total CPU processing capacity $f_c^{total}(k)$ is the product of the frequency of the processors, $f_c(k)$, and the number of cores, $N_c(k)$. Similarly, for a VM V_j, the total CPU processing capacity $f_{vc}^{max}(j)$ is the frequency of the processors, $f_c(k)$, times the number of vCPUs, $N_{vc}(j)$. The million instructions per second (MIPS) rate, $MIPS(j)$, of the VM is calculated by assuming constant Cycles Per Instruction, CPI.

$$f_c^{total}(k) = f_c(k) \cdot N_c(k) \tag{5.4}$$

$$f_{vc}^{max}(j) = f_c(k) \cdot N_{vc}(j) \tag{5.5}$$

$$MIPS(j) = f_{vc}^{max}(j)/CPI \tag{5.6}$$

For example, a server has 4 cores, each running at the frequency of 2 GHz. The server hosts 2 VMs with 1 vCPU each. In that case, the total CPU frequencies available in MHz from the server ($f_c^{total}(k)$) and VM ($f_{vc}^{max}(j)$) are 8,000 MHz and 2,000 MHz, respectively. The processing rate of the VM is 1,000 MIPS. If the CPU usage of the VM is above 80%, the VM is over-loaded and the VM status is set to 1. If the CPU usage falls below 20%, the VM is under-loaded and the VM status is set to 2. If there is a VM failure or deactivation, the VM status is set to 3. The dynamic VM model is described by the following parameters:

1) the status of the VM V_j; status(j)

2) the resource usage by all real-time applications: $f_{vc}^{used}(j)$, M_j^{used}, and

3) the linked list of all allocated applications: pointer.

5.1.3 Formulation of Profile-based Dynamic Assignment

The decision matrix X at time t for application assignment to VM is given by:

$$X(t) = [x_{ij}]_{n\times m}, i \in I, j \in J \tag{5.7}$$

where,

$$x_{ij}(t) = \begin{cases} 1, & \text{if } A_i \text{ is assigned to } V_j,\ i \in I, j \in J \\ 0, & \text{otherwise} \end{cases} \tag{5.8}$$

At time t, the CPU and memory resources used by VM V_j is $f_{vc}^{used}(j,t)$ and memory $M_j^{used}(t)$, respectively. The completion time of all applications on the VM is $t_c(j)$.

$$f_{vc}^{used}(j,t) = \frac{\sum_{i=1}^{n} IC(i)\cdot x_{ij}(t)}{t_c(j)} \cdot CPI \quad \text{(MHz)} \tag{5.9}$$

$$M_j^{used}(t) = \sum_{i=1}^{n} M_r(i)\cdot x_{ij}(t) \quad \text{(KB)} \tag{5.10}$$

In physical servers, CPU is the main power consumer compared with other system components like memory and storage, which have limited dynamic power ranges Barroso and Holzle [2007]. The power consumption of the physical server S_k at time t is calculated using the total CPU utilization percent $f_c^{used}(k,t)$ of the server in the corresponding time period:

$$f_c^{used}(k,t)[\%] = \frac{\sum_{j=1}^{m} f_{vc}^{used}(j,t)\cdot y_{jk}}{f_c^{total}(k)} \times 100 \tag{5.11}$$

$$P_k(t) = \left[(P_k^{max} - P_k^{idle})\cdot f_c k^{used}(k,t)/100\right] + P_k^{idle} \tag{5.12}$$

where, y_{jk} is 1 if VM $V_j, j \in J$, is hosted by server $S_k, k \in K$, and is 0 otherwise.

The ratio of power consumed at maximum to minimum utilization of the host server is given by α_j. Therefore, the energy cost of allocating application A_i to VM V_j is calculated as a

measure of CPU given by:

$$C_{ij} = \alpha_j \cdot IC(i)/MIPS(j) \tag{5.13}$$

The constrained combinatorial optimization model for the assignment of a set of applications to VMs is given as:

$$\begin{aligned} F(obj) &= \min \textstyle\sum_{j=1}^{M} \sum_{i=1}^{N} C_{ij} \cdot x_{ij} && (5.14)\\ \text{s.t.} \quad & \sum_{i=1}^{n} \left[\frac{x_{ij} \cdot IC(i)}{max\,(t_e(i))} \right] \leq MIPS(j),\ \forall j \in J; && (5.15)\\ & \textstyle\sum_{i=1}^{N} x_{ij} \cdot M_r(i) \leq M_j^{max},\ \forall j \in J; && (5.16)\\ & \textstyle\sum_{j=1}^{M} x_{ij} = 1,\ \forall i \in I; && (5.17)\\ & x_{ij} = 0 \text{ or } 1,\ \forall i \in I, j \in J. && (5.18) \end{aligned}$$

The constraints in Equations (5.15) and (5.16) ensure that the allocated resources are within the total capacity of the VM. The constraint in Equation (5.17) restricts an application from running on more than one VM. The binary constraint of the allocation decision variable x_{ij} is given by (5.18).

For a time interval of $\Delta t = 30$ min, the total daily energy consumption of all servers in the data centre is given by:

$$\text{Total Daily Energy Consumption} = \int_0^{48 \cdot \Delta t} \left(\sum_{k=1}^{L} P_k(t) \right) d(t) \tag{5.19}$$

5.2 Profile-based Dynamic Application Management Framework

The workload of an application is dynamic in nature due to a number of factors such as change in resource requirements and load surges due to increased user requests. The work in Fahim et al. [2014] has reported two dynamic allocation algorithms: efficient response time load balancer and minimum processing time load balancer. While these algorithms update the allocation tables with respect to VM load, the update does not happen before the completion of processing current applications. In comparison, our dynamic assignment strategy presented in this Chapter updates the allocation tables periodically. Prior to the assignment of applications to VMs using

RGA, the FFD algorithm sorts the PMs in decreasing order of resource capacity. Each active VM is placed onto the first server with adequate space remaining. All active VMs are eventually packed onto PM servers.

5.2.1 Dynamic Application Assignment

After application, VM and server profiles are built off-line, the real-time allocation of applications to VMs is initiated. The data centre operation is divided into 30 min time slots Δt, the resulting time slot sequence is represented by $T_1, \cdots, T_\lambda, \cdots, T_\Lambda$. The periodic real-time allocation works as follows: The estimated task submission times in the application profiles are used to determine the arrivals of applications in specific time slots. In time slot T_λ, the estimated applications arrival in time slot $T_{\lambda+1}$ is batch processed for allocation of applications to potential VM hosts using the profiles and a Repairing Genetic Algorithm (RGA), which is discussed later.

The potential assignment considers the following information:

1) the resource requirement history of the applications

2) the load history of the VMs; and

3) the current assignment of applications to VMs

The resulting allocation solution is mapped onto an Estimated Assignment Decision Matrix $\hat{X}(T_\lambda)$. Therefore, during time slot $T_{\lambda+1}$, actual arriving applications are treated in a First In First Out (FIFO) order. They are allocated to the pre-determined host VMs according to the estimate decision matrix. After that, the actual decision matrix $X(T_\lambda)$ and profiles are updated.

If the deadline is exceeded in the execution of the application, the overhead is set to a high value, e.g., 10. If the application finishes execution at the exact deadline, set $Overhead = 1$. Otherwise, the overhead of allocating applications using profiles is calculated as:

$$Overhead = \frac{t_w(i)}{t_d(i) - t_f(i)} \tag{5.20}$$

where, $t_f(i)$ represents actual finishing time of the application. The overhead is normalized to the range of [0,1], where 0 and 1 represent the lowest overhead and maximum overheads,

respectively. It is seen from Equation (5.20) that the overhead of allocating profiled applications to VMs is small when the waiting time $t_w(i)$ is low.

The process of the dynamic allocation scheme is described briefly in Algorithm 11. The algorithm executes for at each time slot T_λ. It consists of two sequential processes: Process 1 in Lines 2 to 6 for estimation of application allocation in the next time slot $T_{\lambda+1}$, and Process 2 from Lines 7 to 18 for actual application allocation. Process 1 collects application profiles (Line 3) and VM profiles (Line 4). Then, it deploys RGA for an application assignment solution (Line 5). After that, it creates an estimated allocation decision matrix $\hat{X}(T_{\lambda+1})$. In Process 2, applications arrive in a FIFO queue waiting for assignment to a VM (Line 8). If an application is profiled and expected (Line 9), then allocate the application to a VM using the estimated $\hat{X}_{\lambda+1}$ (Line 10), and terminate the process. Otherwise, the application cannot be allocated to a VM from the estimated $\hat{X}_{\lambda+1}$. There are generally three scenarios: 1) if the application is not profiled, implying that it is a new application (Line 12), then profile this new application (Line 13); 2) the application is an unanticipated applications with random load; and 3) the application is a periodic existing application with different parameters. In all these three scenarios, the application is allocated to the first VM that meets the resource requirements (Lines 14 to 16). After this allocation, the application and VM profiles need to be updated (Line 18) before terminating the process.

5.2.2 Dealing with Infrequent Applications

For infrequent applications described above in the three scenarios, dynamic application assignment to VM requires some special treatments. These treatments are described in the following.

Scenario 1: Arrival of new applications. Arriving applications that do not have profiles are new applications. For a new applications, its resource requirements need to be determined, and its profile will be created online. The new application is then allocated to the first available VM that meets the resource requirements and deadline constraints. The waiting times and allocation overhead of new applications are higher than those of expected applications with profiles. The overhead of allocating new applications falls between 0.75 and 1. However, there are not many new applications in the data centres of universities, government agencies and other small- to medium-scale business companies.

Algorithm 11: Dynamic application assignment algorithm

```
for Time slot T_λ do
    Process 1 - Estimate Allocation for T_{λ+1} ;
    Collect application profiles for submission times within = T_{λ+1};
    Collect VM profiles, set VM status = 0, normal;
    Deploy repairing genetic algorithm RGA (given later in Algorithm ??);
    Create estimated allocation decision matrix X̂(T_{λ+1});
    Process 2 - Actual Allocation for T_λ;
    Applications arrive in a FIFO queue;
    if A_i is profiled & expected then
        Allocate A_i to VM using X̂(T_{λ+1});
    else
        if A_i is not profiled then
            Profile this new application;
        for Each VM do
            if VM resource availability ≥ Application requirement then
                Allocate application A_i to the VM;
                break;
        Update application and VM profiles;
```

Scenario 2: Unanticipated existing applications with random load. Non-periodic applications with random load and variable submission/arrival times are considered as unanticipated applications. Although these applications have profiles, their resource demands are unknown prior to execution. Such applications are allocated promptly to the first available high-MIPS VM that satisfies the resource requirements and deadline constraints. The overhead for such allocations is in the range of 0.4 to 0.75. It is lower than that of allocation of new applications.

Scenario 3: Periodic existing applications with different parameters. This scenario considers a profiled periodic application that arrives at T_λ with different parameters. In this case, if the deadline of the application is less than the estimated finishing time on the pre-determined VM, then the application is newly allocated to the first available high-MIPS VM that satisfies the resource requirements and deadline constraints. Otherwise, the application is allocated to the pre-determined VM host without change. The overhead for such allocations falls between a wider range of 0.01 to 1 depending on the validity of the pre-determined VM host.

Setting VM Status. Considering real-time usage of a VM, the status parameter of the VM profile is set to an integer from 0 to 3. The integer value 0 is for normal workload, 1 for over-loaded VM, 2 for under-loaded VM, and 3 for an inactive scenario due to VM

failure or deactivation. If the status is normal (0), allocation proceeds successfully. The over-loaded and under-loaded CPU usage thresholds are set to 80% and 20%, respectively. Once the thresholds are crossed, a repairing procedure discussed later in Section 5.3 is used to transfer one or more allocated applications from the affected VM to the next available VM. In the case of over-loading, the transfer of applications continue until the CPU usage falls below the higher threshold level. Under-loaded VMs are prioritised as available hosts for new and unanticipated applications with random load. If a VM becomes suddenly inactive due to failure or deactivation, then the executing applications stop and must be requested again by the user. These applications are then directed to the first available VM with high MIPS rate. Concurrently, the energy cost matrix is updated to reflect very high value for the failed VM. This ensures that the affected VM is not selected for hosting applications until the status returns to normal.

5.3 Repairing Genetic Algorithm

An integrated component of the dynamic application assignment framework described in Section 5.2 is the Repairing Genetic Algorithm (RGA). It is implemented during Δt time intervals. At time T_λ, the RGA uses profiles and the expected submission times of the applications to derive estimated allocations solutions for the next time slot $T_{\lambda+1}$, giving an estimated allocation decision matrix $\hat{X}$ (Lines 5 and 6 in Algorithm 11).

In general genetic algorithms, there are two issues that affect the efficiency of deriving an application assignment solution: initial population and infeasible solutions. General steady-state GA methods utilize randomly generated initial population [Vasudevan et al., 2015]. In contrast, the RGA incorporates Longest Cloudlet Fastest Processor (LCFP) heuristics. It considers computational complexity of applications and computing power of processors, to generate an initial population. The advantages of using LCFP include faster convergence and better solutions. The steps involved in implementing LCFP are:

1) Sort applications $A_i, i \in I$ in descending order of execution time;

2) Sort VMs $V_j, j \in J$ in descending order of processing power (MIPS); and

3) Pack sorted applications into fastest processing VM.

For infeasible solutions, an infeasible-solution repairing (IRP) process is employed to convert infeasible solutions to feasible solutions. An infeasible application assignment to a VM is denoted by a 'violation' indicator as a result of VM status violation or resource constraint violation:

$$\text{violation} = \begin{cases} 0, & \text{if resource constraints are not violated} \\ 1, & \text{otherwise} \end{cases} \tag{5.21}$$

When violation $= 1$, the applications assigned to infeasible VMs are re-assigned to other VMs until the violations are resolved. The IRP pursues the following steps:

1) Identify VM $V_j, j \in J$, with violation $= 1$;

2) Calculate resource availability of next VM $V_{j'}, j' \in J$;

3) Assign the first application of V_j to new $V_{j'}$ if the new $V_{j'}$ has more than sufficient resources to host the application; and

4) Repeat above steps until violation $= 0$.

In RGA, the genetic operator settings are given in Table 5.2. The quality of the assignment solution is determined by the fitness function. A lower energy cost and higher resource utilization efficiency result in a higher fitness function. The fitness function is derived as:

$$F(X) = \beta_1 \cdot \bar{F}_{obj} - \beta_2 \cdot \frac{1}{m} \sum_{j=1}^{m} \left[\frac{f_{vc}^{used}(j)}{CPI \cdot MIPS(j)} + \frac{M_j^{used}}{M_j^{max}} \right]. \tag{5.22}$$

The weights $[\beta_1, \beta_2]$ associated with the fitness function is set to $[2, 1]$. The multiplicative inverse of the objective function discussed in Equation (5.14) is represented by $\bar{F_{obj}}$. It is scaled to a range of $[1, 10]$.

Table 5.2: Genetic operator settings.

Initial Population	Randomly generated
Genetic Encoding	Value encoding
Selection	Roulette wheel selection
Crossover	Uniform crossover with binary mask
Mutation	Select & exchange 2 offspring genes
Termination Condition	Until max generations/ suitable solution

A high level description of RGA is given in Algorithm 8, Section 4.3 of Chapter 4. Generally, RGA incorporates LCFP to generate initial population (Lines 1 and 2). Then, it evaluates the fitness of each candidate chromosome (Line 3). If the termination condition is met, output the best fit chromosome as solution. Otherwise, do the following operations sequentially for each generation (Line 5):

1) for each chromosome (Line 6), evaluate the fitness (Line 7); and if the chromosome is infeasible apply IRP process and re-evaluate the fitness (Lines 8 to 10); then select parents (Line 11);

2) for parent chromosomes, do crossover, mutation, and generation of offspring chromosomes (Lines 12 to 15);

3) for offspring chromosomes, evaluate fitness of new candidates, and replace low-fitness chromosomes with better offspring (Lines 16 to 18);

4) for each chromosome, select chromosomes for next generation.

After these sequential operations, the best fit chromosome is selected for output (Line 21).

5.4 Experimental Studies

This Section undertakes experiments to demonstrate the efficiency of the profile-based dynamic application management framework and RGA. It begins with an introduction into experimental design. This is followed by a discussion of evaluation criteria. Then, experimental results are presented.

Experimental Design

In order to fully realize the efficiency of the dynamic application management framework, we have built profiles and conducted all experiments using workload traces from a real data centre. This Chapter has used the MetaCentrum2 workload trace Klusacek et al. [2015] provided by the Czech National Grid Infrastructure MetaCentrum. We have extracted seven days of workload from this trace. The total number of applications considered for our experiments varied per day as expected in a real data centre. On average, there were 2583 applications per

day, each with an instruction count of [5000, 10000] million instructions and [500, 1000] KB of memory.

We have investigated a real data centre with 100 physical servers, each containing 2 to 4 CPU cores running at 1000, 1500 or 2000 MIPS and 8 GB of memory. The total number of VMs considered is 300. Each VM contains 1 vCPU, which we assume is equal to 1 server CPU core. The vCPU runs at 250, 500, 750 or 1000 MIPS and 128 MB of memory. The server power consumptions at maximum and idle utilizations (Equation 5.12) are set at 350 and 150 Watts, respectively as per our observation of the real data centre.

The RGA has a population size of 200 individuals in each generation. The maximum number of generations is set to 200. Probabilities for crossover and mutation are set to 0.75 and 0.02, respectively. The mutation probability is set to a low value in order to control the exploration space. The termination condition is reached when there is no change in the average value and maximum fitness values of strings for 10 generations.

Evaluation Criteria

In order to evaluate the quality and efficiency of the solutions for profile-based dynamic application assignment to VMs, the dynamic scheme presented in this Chapter is compared with static application assignment using RGA (discussed in Chapter 4). As there are no benchmarks for application management, we assume the benchmarks to be existing application assignment strategies: Random (commonly used in data centres today) and First-Fit [Chandio et al., 2013]. The evaluation criteria for testing dynamic-RGA against static-RGA include the following:

1) Energy efficiency: (a) energy consumption; and (b) statistical T-Test analysis; and

2) Quality of solutions: (a) VM resource utilization; (b) makespan performance; (c) degree of imbalance; and (d) estimated finishing time performance.

5.4.1 Energy Efficiency

In our case experiments, FFD is implemented as a policy for VM placement to PMs in the three-layer energy management shown in Figure 1.1. This allows to determine actual energy savings and efficiency of our dynamic application assignment solutions. The presented dynamic-RGA

method is compared with three individual assignment methods used as benchmarks: static-RGA, random, and First-Fit. Random and First-Fit methods are existing assignment methods. Actual energy consumption of all servers across seven days is calculated using Equation 5.19. The results are mapped onto Figure 5.1. It is observed from the figure that both static and dynamic RGAs provide good results in terms of energy consumption. Considering the mean energy consumption of both methods over seven days, the energy savings of dynamic-RGA is 7% more than static-RGA. When compared with existing application assignment strategies (benchmark) dynamic-RGA is 48% and 34% more energy-efficient than Random and First-Fit methods respectively.

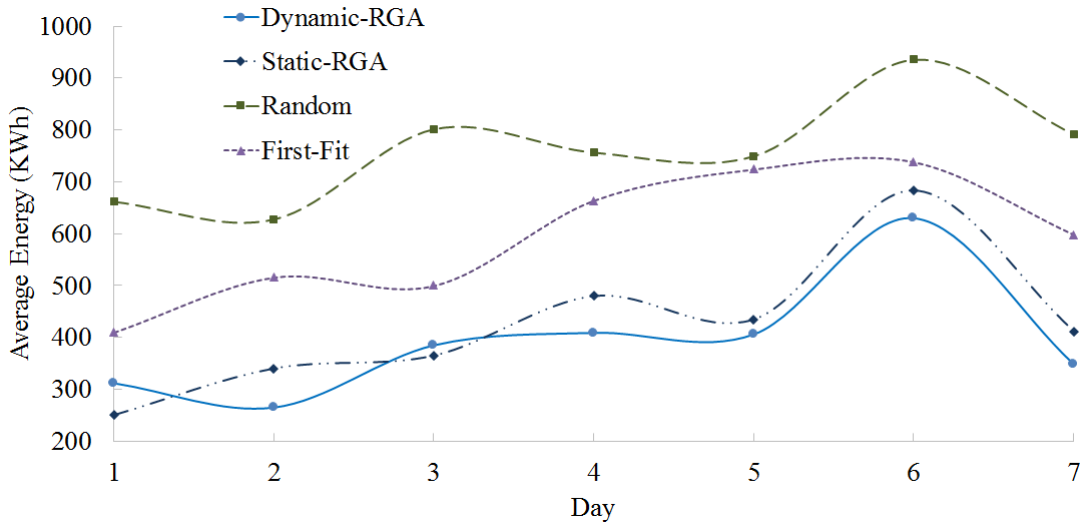


Figure 5.1: Average energy consumptions over seven days for Dynamic-RGA, Static-RGA, Random and First-Fit strategies.

A paired t-test is conducted to determine the confidence level of the experimental results for the two independent strategies of static-RGA and dynamic-RGA. As genetic algorithm is stochastic in nature, both strategies are individually run 30 times for each day. A two-tailed hypothesis is assumed and the confidence interval is set to 95%. The null hypothesis is that there is no difference between the means of static-RGA and dynamic-RGA. Table 5.3 records the average energy consumption across seven days for the two strategies and the corresponding t-stat values. The two-tailed P-value is less that 0.0001 and is extremely statistically significant. The results demonstrate that the difference between dynamic-RGA and static-RGA are significant and thus the null hypothesis is rejected.

Table 5.3: Daily energy consumption of the data centre: static-RGA vs. dynamic-RGA.

Day	Energy (KWh)		T-value	std. error	df
	Static-RGA	Dynamic-RGA			
1	250.57	316.53	-6.05	10.886	29
2	336.73	258.47	-4.96	15.764	29
3	370.63	384.6	-1.45	9.604	29
4	491.97	417.23	-4.75	15.714	29
5	441.77	391.43	-4.49	11.201	29
6	685.57	629.93	-10.58	5.257	29
7	423.63	343.23	-8.28	9.705	29

5.4.2 Quality of Solutions

The quality of solutions is determined by measuring resource utilization, makespan performance, VM degree of imbalance and estimated finishing time performance

VM Resource Utilization. The results of VM resource utilization on implementation of static-RGA and dynamic-RGA on the increasing number of applications is demonstrated in Figure 5.2. Both static-RGA and dynamic-RGA realize a linear progression in terms of utilization efficiency. However, the dynamic-RGA performs upto 10% better in terms of resource utilization than static-RGA. The figure also demonstrates the scalability of the presented approach with respect to increasing problem size.

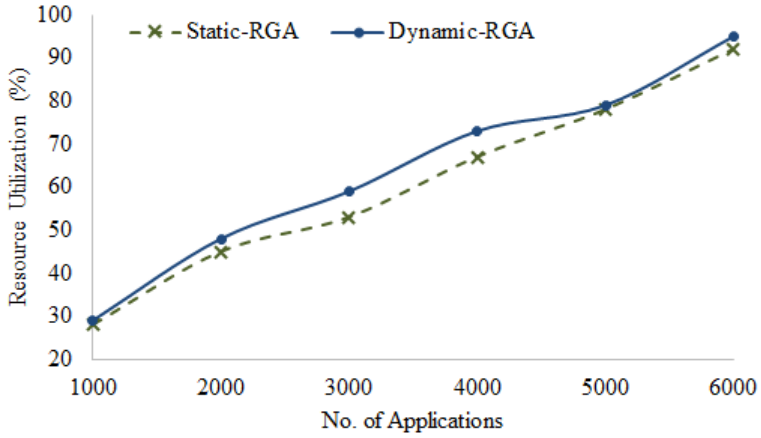


Figure 5.2: VM resource utilization of static-RGA vs. dynamic-RGA.

Makespan. Makespan is the maximum completion time of all applications allocated to a VM:

$$Makespan = \max_{j \in J} t_c(j). \tag{5.23}$$

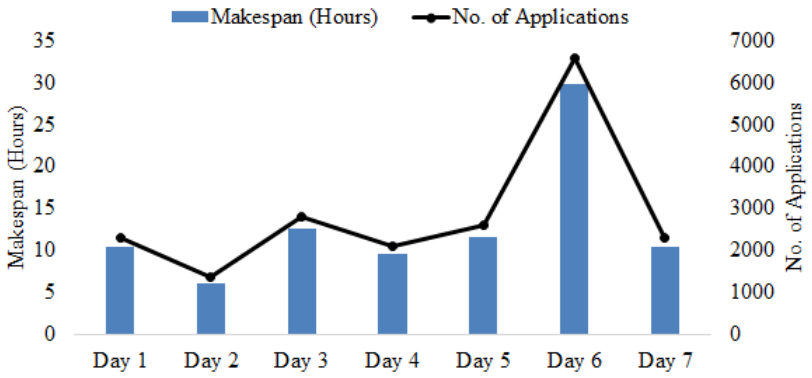


Figure 5.3: Makespan of dynamic RGA.

Completion time is the final time at which all applications conclude processing among all VMs. The number of VMs is a constant 300 and the number of applications varies every day. Figure 5.3 illustrates the maximum makespan incurred on implementing dynamic-RGA for the seven days under consideration. Makespan is linearly dependant on the number of applications.

Degree of Imbalance (DI). The degree of imbalance represents the imbalanced distribution of load among VMs:

$$DI = \frac{Makespan - min(t_c(j)}{avg(t_c(j)}. \tag{5.24}$$

The lower the DI, the more balanced the load distribution. DIs on implementation of the methods dynamic-RGA and static-RGA is 1.2 and 1.8 respectively. This demonstrates that dynamic-RGA is more efficient than static-RGA in terms of generating the lest imbalanced load distribution.

Estimated Finishing Time Performance. The estimated and actual finishing times of the allocated applications over 24 hours is shown in Figure 5.4. The data shown in the figure are sampled at an interval of one hour time slot. While the estimated finishing time deviates from the actual finishing time, the mean of the deviations over 24 hours is as low as 5.551 sec. This demonstrates that the estimated finishing time (Equation 5.3) is near-accurate to the actual finishing times. Especially, all applications allocated through the estimated finishing times meet the deadline constraints, demonstrating the effectiveness of the dynamic application assignment approach presented in this Chapter.

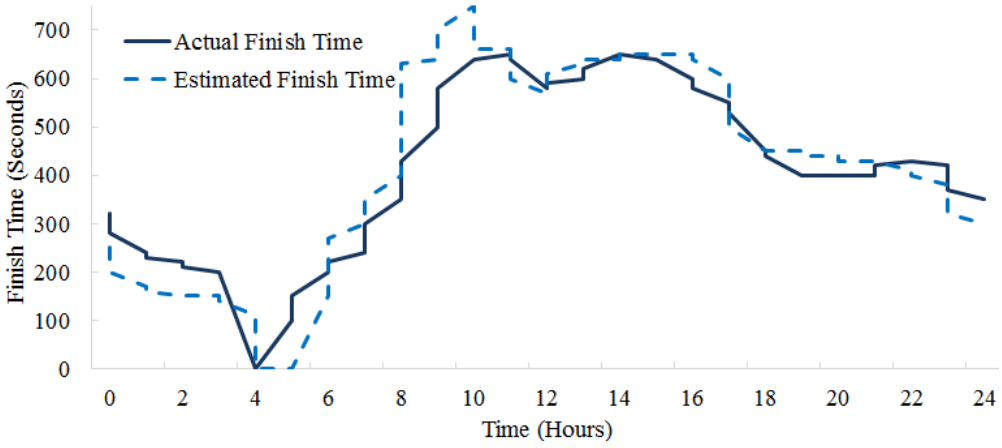


Figure 5.4: Estimated finishing time performance.

5.5 Summary of the Chapter

This Chapter addresses the Dynamic Assignment Problem of implementing profile-based assignment for real-time applications with dynamic workload. It makes the third contribution of our research: a profile-based dynamic application assignment framework. The framework is implemented with a repairing genetic algorithm (RGA). The finishing times of applications are estimated by using profiles to satisfy deadline constraints and reduce waiting times and assignment computation overhead. Strategies are developed to handle infrequent management scenarios such as new/random applications or failed/deactivated VMs. To derive actual energy savings, the dynamic assignment assignment approach has been embedded into a three-layered energy management system. The VM management layer implements a first-fit decreasing (FFD) VM placement policy and the application management layer implements RGA.

Experiments have been conducted to demonstrate the effectiveness and efficiency of the dynamic approach. For the investigated scenarios, the dynamic application assignment approach has shown 48% more energy savings than existing assignment approaches. Dynamic method also displays 10% more resource utilization efficiency and lower degree of VM load imbalance in comparison with the static method. Therefore, the profile-based dynamic assignment is a promising technique for energy-efficient application management in data centres.

The next Chapter 6 addresses the Consolidation Problem. It develops a method for consolidating applications in data centres as our fourth contribution.

Chapter 6

Application Consolidation

Among the four research problems of the thesis, we have addressed profile building, static and dynamic assignment problems in previous chapters. This Chapter is dedicated to the fourth research problem of the thesis: the Consolidation Problem. It develops a methodology for a profile-based application assignment consolidation approach. Consolidation is used to further improve the energy-efficiency of the application assignment solution. Consequently, we seamlessly incorporate a Local Search Optimization (LSO) heuristic to improve the algorithm (RGA) developed in our previous chapters for profile-based application assignment. First-Fit Decreasing (FFD) based VM placement is implemented to complete the three-tiered energy management solution. Experimental studies demonstrate that consolidation improves static and dynamic assignment energy-efficiency by 36.5% and 26.5% respectively.

Consolidation can be implemented at each of the three layers of a data centre: Application, VM and server. Server and VM consolidation of virtualized data centres are popular methods. Both methods involve migration of VMs or workload to empty and switch-off the server [Ahmad et al., 2015, Baruchi et al., 2015, Rao and Thilagam, 2015]. Data centres virtualized with software such as VMware vSphere is equipped with powerful tools for live migration of VMs and server consolidation. However, few small- to medium- data centres choose to employ these tools risking reliability and performance. Most of the existing consolidation techniques fall short of satisfying the Quality of Service (QoS) of applications that run on VMs during consolidation.

To tackle this challenging problem, we implement consolidation at the application layer. Consolidation of applications requires minimum number of VMs executing applications with

high load balance and resource utilization. It involves emptying VMs running on low utilization and shutting them down. Consequently, the host servers of these VMs consume less energy, thereby resulting in actual energy savings. The method of shutting down VMs reduces energy consumption, although not as significant as shutting down servers directly. However, VMs can be brought back online with considerably less time than servers. Also, for sudden spikes in workload, load redistribution can be managed well with the former method as each server is capable of hosting upto 15 VMs. The former method of shutting down VMs has a lower impact on data centre reliability.

In the following sections; 1) the consolidation problem will be formally formulated to minimize active VMs whilst considering fluctuating workload, resource demands, deadline constraints and VM availability; 2) an application consolidation procedure implemented by an Improved-RGA is presented. It incorporates a Local Search Optimization (LSO) heuristic; 3) a three-tiered profile-based energy management framework that implements consolidated application assignment and VM placement is presented; and 4) actual server energy savings are demonstrated when compared with unconsolidated static and dynamic profile-based application assignment.

This Chapter is organized as follows. The notations used throughout this Chapter are listed in Table 6.1. Section 6.1 formulates the application consolidation problem. Section 6.2 describes the application consolidation solution and the improved repairing genetic algorithm. Experimental studies conducted to demonstrate the effectiveness of our solution is given in Section 6.3. Finally, Section 6.4 summarizes the Chapter.

6.1 Consolidation Problem Formulation

A virtualized data centre consists of a set of l servers represented by $\{S_1, \cdots, S_k, \cdots, S_l\}$. The servers host a set of m VMs $\{V_1, \cdots, V_j, \cdots, V_m\}$. Consider a set of n applications $\{A_1, A_2, \cdots, A_i, \cdots, A_n\}$ that must be assigned to these VMs. The consolidation problem has two components: 1) Application Consolidation; and 2) VM Placement. Solving the problem leads to a complete application management solution that saves energy consumption, is performance reliable and load balanced.

Table 6.1: Description of notations used in Chapter 6.

Notation	Description
A_i	Application, $i \in I = \{1, \cdots, n\}$
β_1, β_2	Coefficients in fitness function $F(X)$
C_{ij}	Energy cost of A_i, $i \in I$, assigned to V_j, $j \in J$
$f_{vc}^{used}(j)$	Used vCPU frequency (MHz) of V_j, $j \in J$
$F(obj), F(X)$	Objective function, and fitness function, respectively
i, j, k	Subscripts or indices for applications, VMs and PMs, respectively
I, J, K	Integer sets $I = \{1, \cdots, n\}, J = \{1, \cdots, m\}, K = \{1, \cdots, l\}$
M_j^{max}	Memory Capacity (MB) of V_j, $j \in J$
M_j^{used}	Used Memory (KB) of V_j, $j \in J$
n, m, l	Total numbers of applications, VMs and PMs, respectively
$R_r(i)$	Resource required for A_i, $i \in I$
$R_v^{max}(j)$	Max resource capacity of V_j, $j \in J$
$R_v^{used}(j)$	Used resource of V_j, $j \in J$
R_k^{max}	Max resource capacity of S_k, $k \in K$
S_k	Physical Machine (PM) or Server, $k \in K = \{1, \cdots, l\}$
V_j	Virtual Machine (VM), $j \in J = \{1, \cdots, m\}$
w_k	Max resource weight of S_k, $k \in K$
x_{ij}	Binary assignment decision variable
y_{jk}	Binary VM-server host constant
z_j	Binary variable representing active/inactive status of V_j, $j \in J$

6.1.1 Formulation of Application Consolidation

The assignment problem addressed in the previous chapters required applications to be assigned to VMs using profiles. Profile-based static and dynamic application assignment aimed for energy- and performance-efficiency. This Chapter deals with the additional problem of application consolidation that requires minimum number of VMs to be active, whilst the others are emptied and shut down.

The assignment of an application A_i, $i \in I$, onto a VM V_j, $j \in J$ is represented by a binary decision variable $x_{ij}, i \in I, j \in J$. The binary variable, $z_j(t)$ represents the active or deactivated status of the VM at a specific time interval.

$$z_j(t) = \begin{cases} 1, & \text{if } V_j,\ j \in J \text{ is active at time t} \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

The consolidation problem is mathematically represented as follows:

$$\min \quad \sum_{j=1}^{m} z_j(t) \tag{6.2}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} R_r(i) \cdot x_{ij} \leq R_v^{max}(j),\ \forall i \in I, j \in J; \tag{6.3}$$

$$\sum_{j=1}^{M} x_{ij} = 1,\ \forall i \in I; \tag{6.4}$$

$$z_j, x_{ij} = 0 \text{ or } 1,\ \forall i \in I, j \in J. \tag{6.5}$$

The constraints in Equations (6.3) ensure that the allocated resources are within the total resource capacity, $R_v^{max}(j)$ of the VM V_j, $j \in J$. The resource required by an application A_i, $i \in I$ is given by $R_r(i)$. The sum of the workload of every application assigned to a VM must not exceed the VM resource capacity. The constraint in Equation (6.4) restricts an application from running on more than one VM. The binary constraint of the allocation decision variable x_{ij} and $z_j(t)$ is given by (6.5).

6.1.2 VM Placement

We consider VM placement as a bin packing problem [Song et al., 2014b]. The VMs are placed onto as few servers (bins) as possible using First-Fit Decreasing (FFD). The VM placement problem can be formulated as follows: Each server has a weight $\{w_1, \cdots, w_k, \cdots, w_l\}$ represented by maximum resource capacity ($R_k^{max},\ k \in K$) associated with it. The weights are normalized to a range of $[1, 10]$.

The servers are sorted such that:

$$w_k \geqslant w_{k+1},\ 1 \leqslant k \leqslant l \tag{6.6}$$

Each active VM is placed onto the first server with adequate space by satisfying the following resource condition.

$$\sum_{j=1}^{m} R_v^{max}(j) \cdot y_{jk} \leq R_k^{max},\ \forall\ j \in J,\ k \in K \tag{6.7}$$

6.2 Application Consolidation Procedure

Consolidation of applications is implemented by local search optimization (LSO) heuristic. The number of active VMs required is scaled down. This ensures VM resource-efficiency, further reduces server energy and maintains workload balance. The consolidation approach completes the final solution of energy-efficient profile-based application management in data centres of this thesis. Figure 6.1 demonstrates the application consolidation process. V_2 and V_4 are under-utilized, i.e. operating under user-specified low threshold (in this case, threshold is set at 30%). The LSO procedure re-assigns the applications hosted by V_2 and V_4 to V_3 which has sufficient resources available. Then, V_2 and V_4 are shut down, thereby reducing the number of active VMs from four to one.

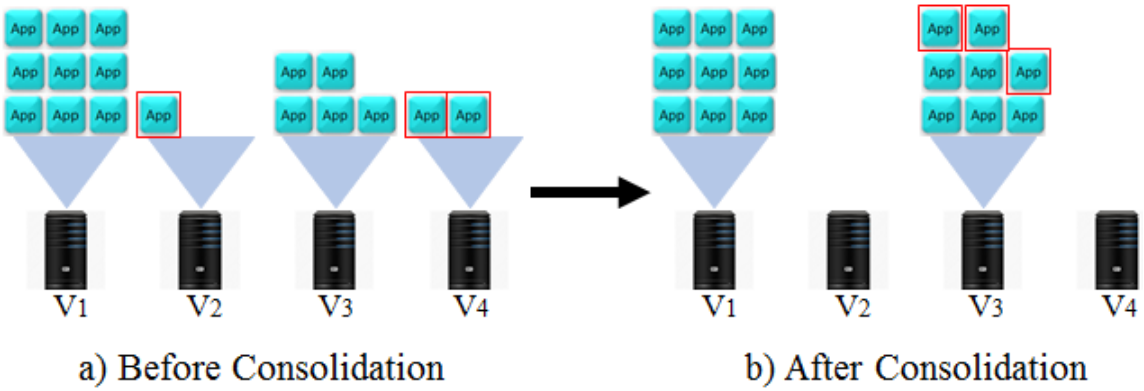


Figure 6.1: Application Consolidation Process.

6.2.1 Local Search Optimization (LSO)

Profile-based application assignment consolidation approach is implemented by LSO. It enables improvement of one design objective without sacrificing or even improving other design objectives. The design objective under current focus is the number of active VMs. Other design objectives which we have satisfied (in our previous chapters) are energy and resource utilization. The LSO is an upgrade to the profile-based application assignment framework. During periods of low workload, the problem of all VMs across the servers handling few or no applications each leads to unnecessary energy consumption. This problem is not obvious during high workloads.

LSO is used to improve the assignment solution chromosomes by increasing resource utilization efficiency. If a VM is functioning below a set threshold of its CPU and memory

capacity, then the local optimization procedure transfers the applications hosted by the VM to another VM. In this procedure, the threshold is set at 0.3, which is 30% of the CPU and memory capacity. The transfer of applications ensures that for a feasible solution, the CPU and memory resource utilizations of active VMs are enhanced. After the local optimization, the fitness of the new solution is compared with the old solution. If the former falls below the latter, the chromosome reverts to the old solution. Algorithm 12 describes the working of the LSO process.

Algorithm 12: Local Search Optimization

```
1  for j = 1 to m do
2      while V_j.pointer ≠ null do
3          if (V_j.R_v^used(j) < 0.3 * V_j.R_v^max(j)) then
4              select = V_j.pointer ;
5              V_j.pointer = V_j.pointer.next ;
6              j' = j + 1 ;
7              while j' ≤ m do
8                  if (V_j'.avail_CPU ≥ select.R_r(i)) then
9                      select.next = V_j'.pointer ;
10                     V_j'.pointer = select ;
11                     V_j'.R_v^used(j) += select.R_r(i) ;
12                 Set j' = j' + 1 ;
```

6.2.2 Improved-Repairing Genetic Algorithm

The LSO is incorporated into our algorithm to create the improved-RGA. The improved-RGA now has three components:

1. Longest Cloudlet Fastest Processor: The LCFP generated initial population packs the longest application to the fastest processor (VM). LCFP sorts applications in decreasing order of estimated execution times. The VMs are sorted in decreasing order of processing capacity. The applications are then packed onto the fastest processing VM. The generated initial population undergoes selection, mutation and crossover to create consequent offspring populations.

2. Infeasible-solution Repairing Procedure: IRP converts infeasible solutions to feasible assignments. VMs that violate resource constraints are identified. Each application hosted by such VMs are reassigned to other suitable VMs until the violation is null.

3. Local Search Optimization: LSO is applied to feasible solution chromosomes to consolidate the assignment of applications. It re-assigns applications and shuts down low-utilization VMs. This ensures better workload balance resulting in low degree of imbalance.

The chromosomes of the improved-RGA are represented by value encoding. The parent chromosomes are selected from the mating pool with the help of roulette wheel selection, which is a fitness-proportionate selection technique. The greater the fitness value of a chromosome, the more probability of being selected as a parent. Uniform crossover is applied to the parent solutions using a binary crossover mask to produce the offspring solutions. Mutation is carried out by selecting and exchanging two genes from the offspring chromosomes. The mutation probability is set to a low number, in order to control the search space. The termination condition specifies that the cycle is repeated for each generation until a maximum number of generations is reached or an individual is found which adequately solves the problem. Every iteration of the algorithm creates a population consisting of a set of chromosomes,which represent a possible assignment solution.

The fitness function defined in Equation 5.22 (Section 5.3) is recalled here:

$$F(X) = \beta_1 \cdot \bar{F}_{obj} - \beta_2 \cdot \frac{1}{m} \sum_{j=1}^{m} \left[\frac{f_{vc}^{used}(j)}{CPI \cdot MIPS(j)} + \frac{M_j^{used}}{M_j^{max}} \right]. \tag{6.8}$$

The weights $[\beta_1, \beta_2]$ associated with the fitness function is set to $[2, 1]$. The multiplicative inverse of the objective function discussed in Equation (5.14) is represented by $\bar{F_{obj}}$. It is normalized and scaled to a range of $[1, 10]$.

The high level description of the improved-RGA is given in Algorithm 13. The improved-RGA incorporates Longest Cloudlet Fastest Processor (LCFP) generated initial population (Lines 1 and 2). Then, it evaluates the fitness of each candidate chromosome (Line 3). If the termination condition is met, output the best fit chromosome as solution. Otherwise, do the following operations sequentially for each generation (Line 5):

1) for each chromosome (Line 6), evaluate the fitness (Line 7); and if the chromosome is infeasible apply IRP process and re-evaluate the fitness (Lines 8 to 10); then select parents (Line 11);

2) for parent chromosomes, do crossover, mutation, and generation of offspring chromosomes (Lines 12 to 15);

3) for offspring chromosomes, evaluate fitness of new candidates (Line 17), apply LSO process (Line 18), evaluate fitness of consolidated candidates (Line 19); compare candidate fitnesses and replace low-fitness chromosomes with better offspring (Lines 20 and 21);

4) for each chromosome, select chromosomes for next generation.

After these sequential operations, the best fit chromosome is selected for output (Line 24).

Algorithm 13: Improved Repairing Genetic Algorithm

```
1  Find output of solutions generated by LCFP;
2  Initialize population with LCFP output;
3  Evaluate fitness of each candidate chromosome;
4  while Termination condition is not satisfied do
5      for Each Generation do
6          for Each chromosome do
7              Evaluate fitness;
8              if Chromosome is infeasible then
9                  Apply IRP;
10                 Evaluate Fitness;
11             Parents selected using Roulette Wheel Selection;
12         for Parent chromosomes do
13             Apply uniform crossover as per crossover probability;
14             Mutate resulting offspring as per mutation probability;
15             Offspring chromosomes generated;
16         for Offspring chromosomes do
17             Evaluate fitness of new candidates;
18             Apply LSO;
19             Evaluate fitness of consolidated candidates;
20             if Consolidated candidate fitness is higher then
21                 Replace low-fitness chromosomes with better offspring;
22         for Each chromosome do
23             Select chromosomes for next generation;
24 Output the best fit chromosome as solution;
```

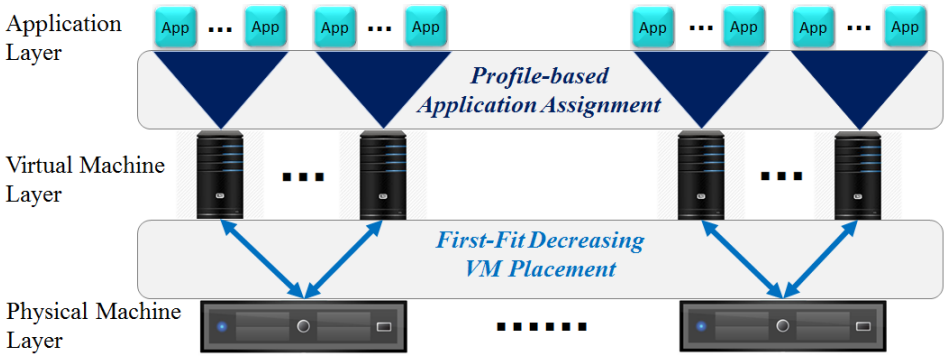


Figure 6.2: Three-tiered energy management.

6.2.3 Three-tiered Energy Management

Energy management is implemented at the three layers of the data centre architecture: application, VM and server as seen in Figure 6.2. It provides a complete solution in terms of energy savings. The profile-based consolidated assignment method is used at the application management layer. The VM management layer implements the first-fit decreasing (FFD) heuristic described in Algorithm 14 for VM placement to servers.

Algorithm 14: First-Fit Decreasing

```
1 Sort PMs in descending order of weights, w_k;
2 for Each PM, S_k, k = 1, 2..., l do
3     for Each VM, V_j, j = 1, 2, ..., m do
4         if V_j fits in S_k then
5             Pack V_j in S_k ;
6             Break ;
```

6.3 Experimental Studies

This Section conducts experiments to demonstrate the efficiency of the application consolidation process. Application consolidation makes up the final phase of our profile-based application assignment framework. We conduct experiments to test and demonstrate the significant improvement of energy- and performance-efficiency results on addition of the consolidation process. It begins with an introduction into experimental design. This is followed by a discussion of evaluation criteria. Then, experimental results are presented.

Experimental Design

The results of the proposed application consolidation scheme are compared with results prior to consolidation Chapter 4 and 5. Implementation of RGA without the local optimization procedure results in all VMs being active. Whereas, on implementation of the improved-RGA, the incorporated LSO minimizes the number of active VMs by migrating the applications in low utilized VMs to other VMs. As such we conduct experimental studies on the following:

1. Static vs Static-Consolidated

2. Dynamic vs Dynamic-Consolidated

Case Study 1: Static-Consolidated Assignment To ascertain the enhancement of the consolidation process, we first compare with the results of the static assignment method discussed in Chapter 4. Experiment design is similar to that of Chapter 4. A data centre hosts 100 servers hosting upto 1000 VMs. For our evaluation, 6 different problem test sets are considered where the number of applications ranges from 200 to 5000 with corresponding number of VMs as seen in Table 6.2.

Table 6.2: Problem test sets for static-consolidation experiments.

Test set	VMs	Applications
1	50	200
2	100	500
3	250	1500
4	500	2500
5	750	3500
6	1000	5000

Case Study 2: Dynamic-Consolidated Assignment To fully realize the improvement, the experimental design is similar to dynamic assignment Chapter 5. Experiments are conducted using MetaCentrum2 workload trace Klusacek et al. [2015] provided by the Czech National Grid Infrastructure MetaCentrum. The profiles are built using workload traces from real data centre systems.

The total number of applications considered for the experiments varied per day as expected in a real data centre. On average, there were 2583 applications per day, each with an instruction count of [5000, 10000] million and [500, 1000] KB of memory. A data centre consisting of 100 physical servers, each containing 2 to 4 CPU cores running at 1000, 1500 or 2000 MIPS and 8

GB of memory is considered. The total number of VMs considered is 300. Each VM contains 1 vCPU, which we assume is equal to 1 server CPU core. The vCPU runs at 250, 500, 750 or 1000 MIPS and 128 MB of memory. The server power consumption at maximum and idle utilizations is set at 350 and 150 Watts respectively.

The improved-RGA has a population size of 200 individuals in each generation. The maximum number of generations is set to 200. Probabilities for crossover and mutation are set to 0.75 and 0.02 respectively. The mutation probability is set to a low constant in order to control the exploration space. The termination condition is reached when there is no change in the average value and maximum fitness values of strings for 10 generations.

Evaluation Criteria

The evaluation criteria for testing assignment against consolidated assignment include the following:

1) Energy efficiency; and

2) Quality of solutions: (a) VM resource utilization; and (b) degree of imbalance.

6.3.1 Energy-Efficiency

Actual energy savings are derived from the consolidation process by implementing the three-tiered energy management framework. The energy-efficiency of static- and dynamic-consolidated assignment is demonstrated below.

Static-Consolidated Assignment. Figure 6.3 demonstrates the daily energy consumption of the static-consolidated method compared to the static method. Upto 36.5% of energy savings is possible with static-consolidated. Moreover, the static-consolidated uses on average half the servers required by static to host the VMs. For the Problem Set 6, the actual number of VMs which is 1000 is reduced to 804 active VMs with the help of improved-RGA. This is equated to saving approximately 165 KWh daily.

From Table 6.3, the VMs required to host the applications can be reduced upto 41%. Therefore, the number of physical machines required to host the active VMs for static-consolidated is much smaller than that of static method for each problem set. On average, seven applications are hosted by each VM. A server can host upto 15 VMs. Using FFD, number of servers required

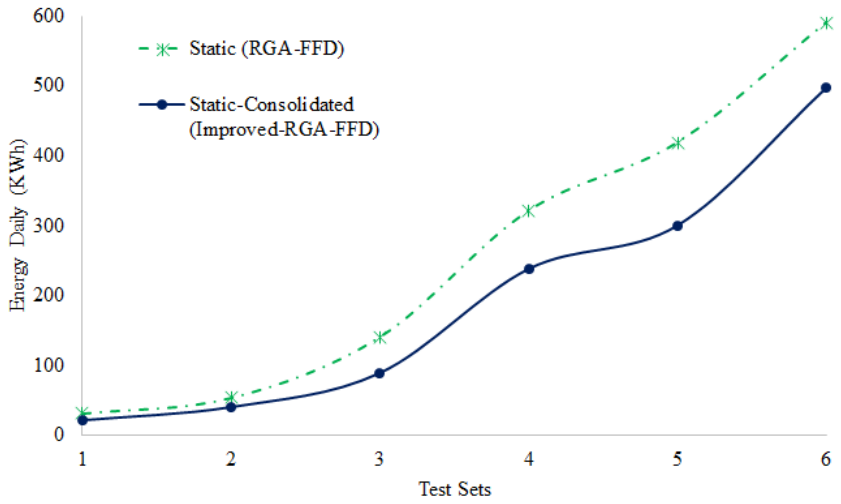


Figure 6.3: Energy consumption of static-consolidated assignment.

to host the VMs is scaled to 3-76.

It is observed that reducing the number of servers required, increases the individual server energy consumption. However, overall energy consumption of the data centre is decreased. Table 6.3 demonstrates that the average energy per hour of servers is generally higher for static-consolidated because of higher CPU utilization on packing multiple VMs to fewer servers.

Table 6.3: Energy consumption per active server for static-consolidated and static methods.

Test Set	Static-Consolidated			Static
	Active VMs	Required PMs	Energy/PM (Wh)	Energy/PM (Wh)
1	34	3	305	262
2	59	6	283	204
3	116	13	316	218
4	298	30	331	216
5	512	49	274	233
6	804	76	273	246

Dynamic-Consolidated Assignment. The dynamic-consolidated method exhibits 26.5% more energy efficiency than dynamic method. This can warrant upto saving 68 KWh per day for the data centre. From Figure 6.4, it can be inferred that the difference in energy consumption between both methods is low during high workload and vice versa. Day 6 had the highest influx of workload at nearly 6000 applications. The percentage difference in energy of both methods is low as 0.78%. Lowest workload appears on Day 2 with 1200 applications. The energy-efficiency of application consolidated is evident from the results.

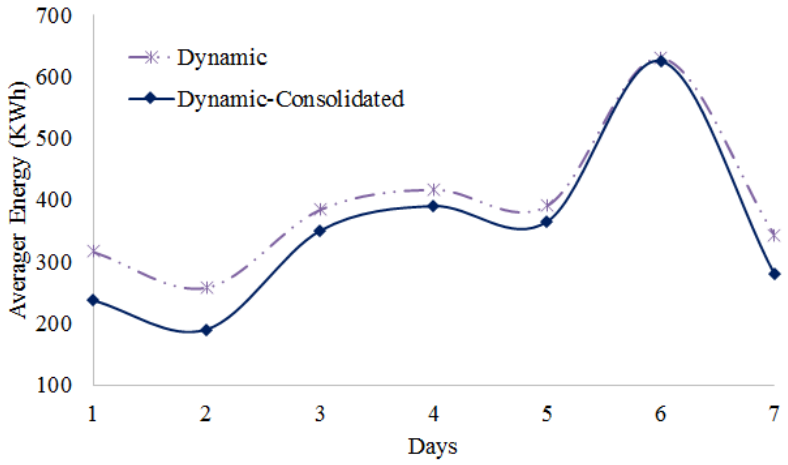


Figure 6.4: Energy consumption of dynamic-consolidated assignment.

6.3.2 Quality of Solution

The quality of solutions is determined by measuring resource utilization and VM degree of imbalance.

Resource Utilization.

Static-Consolidated. The consolidated process is capable of increasing average resource utilization of VMs to 83%. This is 38% more resource-efficient than the static method. The increase is attributed to low number of active VMs. Whereas the static method utilizes all available VMs.

Dynamic-Consolidated. From Figure 6.5, the consolidated method maintains the average resource utilization between 65% to 82%. This is due to reducing the number of active VMs and satisfying the upper- and lower- threshold constraint set between 30% to 85%. The number of active VMs is reduced to less than half the total number of VMs. The dynamic method uses the entire 300 VMs for application assignment. The dynamic-consolidated method uses less than 150 VMs on average during the seven days under consideration.

Degree of Imbalance.

Represents the imbalanced distribution of load among VMs. It is dependent on the makespan of the VMs. The lower the DI, the more balanced the load distribution. The DI of the static and dynamic assignment methods is 1.8 and 1.2 respectively. The consolidation process improves the DI of the methods to 1.4 and 1.1 for static- and dynamic-consolidated methods respectively.

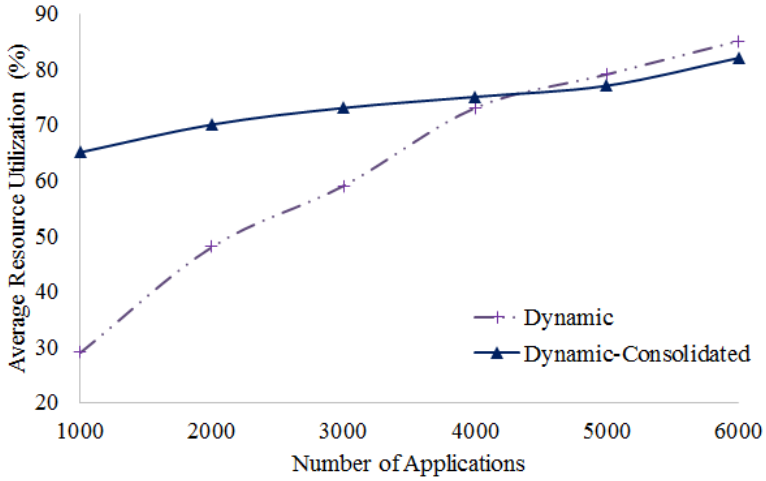


Figure 6.5: Average resource utilization of active VMs.

6.4 Summary of the Chapter

This Chapter is dedicated to the fourth and final research problem of the thesis: the Consolidation Problem. It makes the fourth contribution of our research by presenting a profile-based application assignment consolidation approach. Consolidation is implemented by a local search optimization (LSO) heuristic. The number of active VMs required is scaled down. This ensures VM resource-efficiency, further reduces server energy and maintains workload balance. The consolidation approach completes the final solution of energy-efficient profile-based application management in data centres of this thesis. Experimental analysis demonstrate that the consolidation process improves static and dynamic assignment energy-efficiency by 36.5% and 26.5% respectively. The number of active VMs is reduced such that the resource utilization is high and degree of imbalance is low. Therefore, consolidation of applications is a successful component of our profile-based application management framework.

The next Chapter 7 summarizes the complete profile-based application assignment framework and concludes this thesis.

Chapter 7

Conclusions and Recommendations

This Chapter summarizes the research work on profile-based application management for green (energy-efficient) data centres. It highlights the main contributions, discusses research limitations and outlines future research directions.

7.1 Summary of the Research

Data centres are indispensable in this Information Age. The proliferation of internet users and data have led to large-scale deployment of data centres around the world. However, they consume immense amounts of energy resulting in exorbitant energy costs and excessive carbon footprint. This demands green initiatives and energy-efficient strategies for greener data centres. Inefficiencies of resources provisioned to application workloads need to be well investigated and addressed. Improving application management of a data centre is an effective way for this purpose. Assignment of an application to different virtual machines affects both the energy consumption and resource utilization significantly. However, energy-efficiency and resource utilization are conflicting in general. Thus, it is imperative to develop a scalable application assignment strategy for a trade-off between energy-efficiency and Quality of Service (QoS) such as resource utilization and workload balance.

To address this problem, a profile-based application management framework has been presented in this thesis for greener data centres. Profiles provide prior knowledge of the runtime characteristics of the workload. The profile-based application management framework addresses the four research questions stated in Section 1.2: The Profiling Problem, The Static

Assignment Problem, The Dynamic Assignment Problem and The Consolidation Problem. The solutions for the research questions discussed in Chapters 3 through 6 form the four major contributions of this thesis. Chapter 3 explores profiles and profile building as an application assignment strategy. Chapters 4, 5 and 6 make use of profiles for static, dynamic and consolidated application assignment.

Firstly, the concept of profiles and profile building have been discussed (The Profiling Problem). Profiles are built from real data centre workload logs for applications, virtual machines (VMs) and servers. They are used to predict future workload and plan application assignment to VMs in advance. A profile-based application assignment problem have been formulated as a penalty-based constrained optimization model. This is solved by a Penalty-based Profile Matching Algorithm. Experimental studies have shown that the profile-based application assignment approach is feasible, scalable and 22% more energy-efficient in comparison with other existing/benchmark approaches. This implies that the profiling approach is successful in providing energy-efficient assignment solutions with good CPU utilization efficiency and application completion times within deadlines. Thus, the first contribution of the thesis presented in Chapter 3 solves the first research question: Profiling Problem.

Secondly, a Repairing Genetic Algorithm (RGA) have been designed to implement static profile-based application assignment (The Static Assignment Problem). RGA is incorporated with Longest Cloudlet Fastest Processor (LCFP) generated initial population and an Infeasible-solution Repairing Procedure (IRP). LCFP allows for faster convergence and minimized makespan. IRP converts infeasible solutions to feasible solutions by re-assigning applications to better VMs until constraint violations are null. Experiments have been conducted to demonstrate the effectiveness and efficiency of the profile-based RGA. It is implemented at the top application management layer in the three-layer energy management. In addition, for a complete energy management system, First Fit Decreasing (FFD) is implemented at the middle VM management layer in the three-layer energy management. For the investigated scenarios, RGA has shown 23% less energy consumption and 43% more resource utilization in comparison with the steady-state GA. The improved solutions are achieved with faster convergence and shorter computing time than GA. Therefore, the profile-based RGA has been established as a promising tool for energy-efficient application assignment to VMs in data centres. This solves the second research question: Static Assignment Problem and makes the second contribution of the thesis presented in Chapter 4

Thirdly, a profile-based dynamic application assignment framework have been developed to handle real-time applications and workload (The Dynamic Assignment Problem). It is implemented by the RGA. Profiles are used to estimate finishing time of applications, such that deadline constraints are satisfied and waiting time is minimized. The dynamic approach exhibits robustness by handling infrequent scenarios such as new/random applications or failed/deactivated VMs. The overhead of using profiles for real-time workload have been studied. Experiments are conducted to demonstrate the effectiveness and efficiency of the dynamic approach over the static approach and other benchmark approaches. Three-tiered energy management system is considered to derive actual energy savings. Profile-based dynamic application assignment is implemented at the application management layer. FFD VM placement is implemented at the VM management layer. For the investigated scenarios, the dynamic application assignment approach produces 48% more energy savings than existing benchmark application assignment approaches. It also performs better than the profile-based static assignment with 7% more energy savings, 10% more resource utilization efficiency and lower degree of VM load imbalance. Thus, the third contribution of the thesis presented in Chapter 5 solves the third research question: Dynamic Assignment Problem.

Finally, a methodology for profile-based application assignment consolidation approach have been developed (The Consolidation Problem). It is implemented by a Local Search Optimization (LSO) heuristic. The LSO is integrated into the RGA to develop the Improved-RGA. Consolidation of applications require low-workload VMs to be emptied and shut down. This scales down the number of active VMs in a data centre, thereby ensuring VM resource-efficiency. Consequently, the number of servers and power required to host the VMs also scales down. This further reduces server energy consumption and maintains workload balance. Experimental analysis of the three-tiered energy management system using profiles demonstrates significant energy savings. Consolidation improves static and dynamic assignment energy-efficiency by 36.5% and 26.5% respectively. Therefore, consolidation of applications is a successful final component of our profile-based application management framework. This solves the fourth research question: Consolidation Problem and makes the fourth contribution of our thesis presented in Chapter 6.

Thus, we have successfully addressed all the four research questions mentioned in Section 1.2. The four contributions forms our final profile-based application management framework for green (energy-efficient) data centres.

7.2 Limitations and Future Recommendations

Evolution of next-generation IT applications in science, business and academic fields necessitates energy-efficient application management of cloud data centres explored in this thesis. This will enable efficient provisioning of resources with lower energy consumptions and carbon footprint. The research presented in this thesis can be integrated into open source software, such as Open-Stack or proprietary software, such as VMware vShere. This will advance innovation in the development of next generation computing systems.

This research only focuses on small- to medium-data centres. Profiles have been built, tested and applied to application assignment using workload logs from data centres managed by universities, businesses and government agencies. This is because small- to medium-data centres claim ownership of 95% of the world's total data centre usage. Future recommendation include considering the following two issues:

1. Larger-scale data centres. This will require consideration of various factors in profile building such as communication, networks and storage.

2. Variable workload. This involves highly variable workload during short periods of time due to implementation of technologies such as cloud and high performance computing.

The framework presented in this thesis can be expanded to handle the above issues by modifying profile building.

Literature Cited

Ahmad, R. W., Gani, A., Hamid, S. H. A., Shiraz, M., Yousafzai, A., and Xia, F. (2015). A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications*, 52:11 – 25.

Apple (2016). Environmental responsibility report: Progress report, covering fy2015. Technical report, Apple Inc.

Arroba, P., Risco-Martn, J. L., Zapater, M., Moya, J. M., Ayala, J. L., and Olcoz, K. (2014). Server power modeling for run-time energy optimization of cloud computing facilities. *Energy Procedia*, 62:401 – 410.

Bahrpeyma, F., Zakerolhoseini, A., and Haghighi, H. (2015). Using IDS fitted Q to develop a real-time adaptive controller for dynamic resource provisioning in cloud's virtualized environment. *Applied Soft Computing*, 26:285 – 298.

Bajpai, P. and Kumar, M. (2010). Genetic algorithm an approach to solve global optimization problems. *Indian Journal of Computer Science and Engineering*, 1(3):199–206.

Bang, S.-Y., Bang, K., Yoon, S., and Chung, E.-Y. (2009). Run-time adaptive workload estimation for dynamic voltage scaling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(9):1334–1347.

Barroso, L. and Holzle, U. (2007). The case for energy-proportional computing. *Journal of Computer*, 40(12):33–37.

Barroso, L. A., Clidaras, J., and Hlzle, U. (2013). *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*. Morgan & Claypool Publishers.

Baruchi, A., Midorikawa, E. T., and Sato, L. M. (2015). Reducing virtual machine live migration overhead via workload analysis. *IEEE Latin America Transactions*, 13(4):1178–1186.

Bashroush, R., Woods, E., and Noureddine, A. (2016). Data center energy demand: What got us here won't get us there. *IEEE Software*, 33(2):18–21.

Baxter, J. and Patel, J. (1992). Profiling based task migration. In *Proceedings of the Sixth International Parallel Processing Symposium*, pages 192–195, Beverly Hills, California, USA. IEEE.

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

Beloglazov, A., Buyya, R., Lee, Y. C., and Zomaya, A. (2011). A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers*, 82:47–111.

Benbrahim, S. E., Quintero, A., and Bellaiche, M. (2014). New distributed approach for an autonomous dynamic management of interdependent virtual machines. In *Proceedings of the Eighth Asia Modelling Symposium*, pages 193–196, Taipei, Taiwan, China. IEEE.

Berral, J. L., Gavalda, R., and Torres, J. (2011). Adaptive scheduling on power-aware managed data-centers using machine learning. In *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing*, pages 66–73, Lyon, France. IEEE.

Bhoi, U. and Ramanuj, P. N. (2013). Enhanced max-min task scheduling algorithm in cloud computing. *International Journal of Application or Innovation in Engineering and Management (IJAIEM)*, 2(4):259–264.

Binkley, A. (2016). Renewable energy innovations in green data centers. Retrieved on 30/05/2016 from `http://datacenterfrontier.com/renewable-energy-innovations-driving-the-green-boom/`.

Blackburn, M. (2008). *Five ways to reduce data center power consumption (white paper)*. The Green Grid.

Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., and Rajamony, R. (2002). The case for power management in web servers. *Power aware computing*, pages 261–289.

Buyya, R., Beloglazov, A., and Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1–12, Las Vegas, Nevada, USA. CSREA Press.

Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering Cloud Computing: Foundations and Applications Programming*. Elsevier, Amsterdam, The Netherlands.

Calheiros, R. and Buyya, R. (2014). Meeting deadlines of scientific workflows in public clouds with tasks replication. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1787–1796.

Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A. F. D., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50.

Campbell, A., Wu, A. S., and Shumaker, R. (2008). Multi-agent task allocation: learning when to say no. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 201–208, Atlanta, Georgia, USA. ACM.

CarbonZone (2010). Carbon reduction commitment. Retrieved on 05/05/2016 from `http://www.carbonreductioncommitment.co.uk/`.

Caron, G., Hansen, P., and Jaumard, B. (1999). The assignment problem with seniority and job priority constraints. *Operations Research*, 47(3):449–453.

Chandio, A. A., Xu, C. Z., Tziritas, N., Bilal, K., and Khan, S. U. (2013). A comparative study of job scheduling strategies in large-scale parallel computational systems. In *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 949–957, Melbourne, VIC, Australia. IEEE.

Chaobo, Y. and Qianchuan, Z. (2008). Advances in assignment problem and comparison of algorithms. In *Proceedings of the 27th Chinese Control Conference*, pages 607–611, Kunming, Yunnan, China. IEEE.

Chen, F., Grundy, J., Schneider, J.-G., Yang, Y., and He, Q. (2014a). Automated analysis of performance and energy consumption for cloud applications. In *Proceedings of the Fifth ACM/SPEC International Conference on Performance Engineering*, ICPE '14, pages 39–50, New York, NY, USA. ACM.

Chen, F., Grundy, J., Yang, Y., Schneider, J.-G., and He, Q. (2013). Experimental analysis of task-based energy consumption in cloud computing systems. In *Proceedings of the Fourth ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 295–306, New York, NY, USA. ACM.

Chen, L. Y., Ansaloni, D., Smirni, E., Yokokawa, A., and Binder, W. (2012). Achieving application-centric performance targets via consolidation on multicores: Myth or reality? In *Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, pages 37–48, New York, NY, USA. ACM.

Chen, Q., Zheng, L., Guo, M., and Huang, Z. (2014b). EEWA: Energy-efficient workload-aware task scheduling in multi-core architectures. In *IEEE International Parallel Distributed Processing Symposium Workshops (IPDPSW)*, pages 642–651, Phoenix, AZ, USA. IEEE.

Chiang, R. and Huang, H. (2011). TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, Seatle, WA. IEEE.

Cook, G. and Pomerantz, D. (2015). Clicking clean: A guide to building the green internet. Technical report, Greenpeace.

Corcoran, P. M. and Andrae, A. S. (2013). Emerging trends in electricity consumption for consumer ict. *Electrical and Electronic Engineering Reports*, pages 1–56.

Costa-Requena, J., Peuhkuri, M., and Manner, J. (2014). Analysis of software and server hardware impact on energy efficiency. In *Proceedings of the IEEE International Energy Conference (ENERGYCON)*, pages 716–721, Cavtat, Croatia. IEEE.

Data Center Huddle (2016). Data center 101 the basics. Retrieved on 04/06/2016 from `http://www.dchuddle.com/data-center-101/`.

Dayarathna, M., Wen, Y., and Fan, R. (2016). Data center energy consumption modeling: A survey. *IEEE Communications Surveys Tutorials*, 18(1):732–794.

Delforge, P. (2014). *Americas Data Centers Are Wasting Huge Amounts of Energy (Issue Paper)*. Natural Resources Defense Council (NRDC).

den Bossche, R. V., Vanmechelen, K., and Broeckhove, J. (2013). Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. *Future Generation Computer Systems*, 29(4):973–985.

Desnoyers, P., Wood, T., Shenoy, P., Singh, R., Patil, S., and Vin, H. (2012). Modellus: Automated modeling of complex internet data center applications. *ACM Transactions on the Web*, 6(2):8:1–8:29.

Do, A. V., Chen, J., Wang, C., Lee, Y. C., Zomaya, A., and Zhou, B. B. (2011). Profiling applications for virtual machine placement in clouds. In *Proceedings of the Fourth IEEE International Conference on Cloud Computing*, pages 660–667, Washington, DC, USA. IEEE.

Doria, A., Carlino, G., Iengo, S., Merola, L., Ricciardi, S., and Staffa, M. (2010). Powerfarm: A power and emergency management thread-based software tool for the ATLAS Napoli Tier2. *Journal of Physics: Conference Series*, 219(5):1–10.

Doulamis, N., Doulamis, A., Panagakis, A., Dolkas, K., Varvarigou, T., and Varvarigos, E. (2004). A combined fuzzy-neural network model for non-linear prediction of 3-D rendering workload in grid computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(2):1235–1247.

Ergu, D., Kou, G., Peng, Y., Shi, Y., and Shi, Y. (2013). The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. *The Journal of Supercomputing*, 64(3):835–848.

Evans, D. (2011). The internet of things. *Cisco Blogs*.

Ewashko, T. A. and Dudding, R. C. (1971). Application of kuhn's hungarian assignment algorithm to posting servicemen. *Operations Research*, 19(4):991.

Facebook (2016). Carbon energy water footprint and report: Facebook sustainability. Retrieved on 04/06/2016 from `https://sustainability.fb.com/en/our-footprint/`.

Fahim, Y., Ben Lahmar, E., Labriji, E., and Eddaoui, A. (2014). The load balancing based on the estimated finish time of tasks in cloud computing. In *Second World Conference on Complex Systems (WCCS)*, pages 594–598, Agadir, Morocco. IEEE.

Fanti, M., Mangini, A., and Ukovich, W. (2012). A quantized consensus algorithm for distributed task assignment. In *Proceedings of the 51st IEEE Annual Conference on Decision and Control (CDC)*, pages 2040–2045, Maui, HI, USA. IEEE.

Gates, R. (2015). Top data center industry trends of 2016. Retrieved on 31/05/2016 from `http://searchdatacenter.techtarget.com/news/4500260725/Eight-emerging-data-center-trends-to-follow-in-2016`.

Gertphol, S., Yu, Y., Gundala, S., Prasanna, V., Ali, S., Kim, J.-K., Maciejewski, A., and Siegel, H. (2002). A metric and mixed-integer-programming-based approach for resource allocation in dynamic real-time systems. In *Proceedings of the International Parallel and Distributed Processing Symposium*, pages 1–10, Ft. Lauderdale, FL, USA. IEEE.

Ghorbannia Delavar, A. and Aryan, Y. (2014). HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems. *Cluster Computing*, 17(1):129–137.

Google (2016). Google data centers: Renewable energy. Retrieved on 04/06/2016 from `https://www.google.com.au/about/datacenters/renewable/`.

Grehant, X. and Demeure, I. (2009). Symmetric mapping: An architectural pattern for resource supply in grids and clouds. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, Rome, Italy. IEEE.

Guzek, M., Pecero, J. E., Dorronsoro, B., and Bouvry, P. (2014). Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems. *Applied Soft Computing*, 24:432–446.

Han, J., Jeon, S., Choi, Y.-r., and Huh, J. (2016). Interference management for distributed parallel applications in consolidated clusters. In *Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, pages 443–456, New York, NY, USA. ACM.

Han, K. and Cai, X. (2013). Speed-scaling-based job/tasks deployment for energy-efficient datacenters in cloud computing. In *Proceedings of the Second International Conference on Innovative Computing and Cloud Computing*, ICCC '13, pages 154–157, New York, NY, USA. ACM.

Hao, Y. and Liu, G. (2015). Evaluation of nine heuristic algorithms with data-intensive jobs and computing-intensive jobs in a dynamic environment. *Institution of Engineering and Technology (IET) Software*, 9(1):7–16.

Hatime, H., Pendse, R., and Watkins, J. (2013). Comparative study of task allocation strategies in multirobot systems. *IEEE Sensors Journal*, 13(1):253–262.

Hermenier, F., Lawall, J., and Muller, G. (2013). Btrplace: A flexible consolidation manager for highly available applications. *IEEE Transactions on Dependable and Secure Computing*, 10(5):273–286.

Hong, L. I., Ing, W. K., and Barsoum, N. N. (2012). Workload prediction of wireless sensor network based on linear prediction. In *Proceedings of the IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, pages 160–164, Bangkok, Thailand. IEEE.

Huang, L., Ye, R., and Xu, Q. (2011). Customer-aware task allocation and scheduling for multi-mode MPSoCs. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference*, pages 387–392, New York, NY, USA. IEEE.

Huang, R. and Masanet, E. (2015). *Chapter 20: Data Center IT Efficiency Measures, Technical Report*. USA.

Hwang, I. and Pedram, M. (2016). Hierarchical, portfolio theory-based virtual machine consolidation in a compute cloud. *IEEE Transactions on Services Computing*, PP(99):1–14.

Kessaci, Y., Mezmaz, M., Melab, N., Talbi, E.-G., and Tuyttens, D. (2011). *Intelligent Decision Systems in Large-Scale Distributed Environments*, chapter Parallel Evolutionary Algorithms for Energy Aware Scheduling, pages 75–100. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kim, D., Kim, B., and Cho, J. (2014). A partitioning scheme to guarantee minimum execution time for multiple applications in sensor network nodes. In *Proceedings of the International Conference on Information Networking*, pages 126–130, Phuket, Thailand. IEEE.

Klusacek, D., Toth, S., and Podolnikova, G. (2015). Real-life experience with major reconfiguration of job scheduling system. In *Proceedings of the 19th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–19, Hyderabad, India. IEEE.

Kong, X., Lin, C., Jiang, Y., Yan, W., and Chu, X. (2011). Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. *Journal of Network and Computer Applications*, 34(4):1068 – 1077.

Kontogiannis, T. (2005). Adaptable task modelling and its application to job design for safety and productivity in process control. In *Proceedings of the Annual Conference on European Association of Cognitive Ergonomics*, EACE '05, pages 27–34, Chania, Greece. University of Athens.

Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. Technical report, Analytics Press, Oakland, California, USA.

Korupolu, M., Singh, A., and Bamba, B. (2009). Coupled placement in modern data centers. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, pages 1–12, Rome, Italy. IEEE.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Le, K., Bianchini, R., Martonosi, M., and Nguyen, T. (2009). Cost- and energy-aware load distribution across data centers. In *Proceedings of HotPower*, pages 1–5, Montana, USA.

Lei, H., Zhang, T., Liu, Y., Zha, Y., and Zhu, X. (2015). SGEESS: Smart green energy-efficient scheduling strategy with dynamic electricity price for data center. *Journal of Systems and Software*, 108:23–38.

León, X. and Navarro, L. (2013). A stackelberg game to derive the limits of energy savings for the allocation of data center resources. *Future Generation Computer Systems*, 29:74–83.

Li, J., Shuang, K., Su, S., Huang, Q., Xu, P., Cheng, X., and Wang, J. (2012a). Reducing operational costs through consolidation with resource prediction in the cloud. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '12, pages 793–798, Washington, DC, USA. IEEE.

Li, W., Delicato, F. C., Pires, P. F., and Zomaya, A. Y. (2012b). Energy-efficient three-phase task scheduling heuristic for supporting distributed applications in cyber-physical systems. In *Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '12, pages 229–238, New York, NY, USA. ACM.

Li, Y., Han, J., and Zhou, W. (2014). Cress: Dynamic scheduling for resource constrained jobs. In *Proceedings of the 17th IEEE International Conference on Computational Science and Engineering (CSE)*, pages 1945–1952, Chengdu, Sichuan, China. IEEE.

Liu, N., Dong, Z., and Rojas-Cessa, R. (2012). Task and server assignment for reduction of energy consumption in datacenters. In *Proceedings of the 11th IEEE International Symposium on Network Computing and Applications*, pages 171–174, Cambridge, MA, USA. IEEE.

Liu, X., Wang, C., Zhou, B. B., Chen, J., Yang, T., and Zomaya, A. Y. (2013). Priority-based consolidation of parallel workloads in the cloud. *IEEE Transactions on Parallel and Distributed Systems*, 24(9):1874–1883.

Lo, V. M. (1988). Heuristic algorithms for task assignment in distributed systems. *IEEE Transactions on Computers*, 37(11):1384–1397.

Lublin, U. and Feitelson, D. G. (2001). The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63:2003.

Luis Bassa, C. and Gil-Lafuente, A. M. (2012). *Soft Computing in Management and Business Economics: Volume 1*, chapter The Hungarian Algorithm for Specific Customer Needs, pages 363–379. Springer Berlin Heidelberg, Berlin, Heidelberg.

Luo, L., Wu, W., Tsai, W., Di, D., and Zhang, F. (2013). Simulation of power consumption of cloud data centers. *Simulation Modelling Practice and Theory*, 39:152–171.

Machol, R. E. (1970). An application of the assignment problem. *Operations Research*, 18(4):745–746.

Markets and Markets (2015). Data center power market by solution type (power distribution and measurement, power back-up, and cabling infrastructure), by service type, by end user type, by vertical and by region - global forecast to 2020. Technical report, Research and Markets.

Mars, J., Tang, L., Hundt, R., Skadron, K., and Soffa, M. L. (2011). Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 248–259, New York, NY, USA. IEEE.

Mars, J., Tang, L., Skadron, K., Soffa, M., and Hundt, R. (2012). Increasing utilization in modern warehouse-scale computers using bubble-up. *IEEE Micro*, 32(3):88–99.

Mathew, T., Sekaran, K. C., and Jose, J. (2014). Study and analysis of various task scheduling algorithms in the cloud computing environment. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 658–664, New Delhi, India. IEEE.

Maza, C. (2016). How americas big data centers are going green. Retrieved on 04/06/2016 from `http://www.csmonitor.com/Environment/Energy/2016/0330/How-America-s-big-data-centers-are-going-green`.

McFarlane, R. (2015). Is energy-efficient software the next step to reduce operating costs? Retrieved on 31/05/2016 from `http://searchdatacenter.techtarget.com/`.

Melis, M. (2013). Practical methods for improving ict sustainability. Retrieved on 04/06/2016 from `http://www.sustainability-perspectives.com/article/practical-methods-for-improving-ict-sustainability/`.

Moens, H., Handekyn, K., and De Turck, F. (2013). Cost-aware scheduling of deadline-constrained task workflows in public cloud environments. In *Proceedings of the IFIP/IEEE*

International Symposium on Integrated Network Management, pages 68–75, Ghent, Belgium. IEEE.

Nagothu, K., Kelley, B., Prevost, J., and Jamshidi, M. (2010). Ultra low energy cloud computing using adaptive load prediction. In *Proceedings of the World Automation Congress*, pages 1–7, Kobe, Kansai, Japan. IEEE.

Nguyen, H., Shen, Z., Gu, X., Subbiah, S., and Wilkes, J. (2013). AGILE: Elastic distributed resource scaling for infrastructure-as-a-service. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, pages 69–82, San Jose, CA, USA. USENIX.

Nunez, C. (2014). "zombie" servers and inefficiency drive energy waste at data centers. *National Geographic*.

Orgerie, A.-C., Assuncao, M. D. d., and Lefevre, L. (2014). A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys*, 46(4):47:1–47:31.

Outlook, I. (2014). Industry outlook: Data center energy efficiency. *The Datacenter Journal*.

Oxley, M. A., Pasricha, S., Maciejewski, A. A., Siegel, H. J., Apodaca, J., Young, D., Briceo, L., Smith, J., Bahirat, S., Khemka, B., Ramirez, A., and Zou, Y. (2015). Makespan and energy robust stochastic static resource allocation of a bag-of-tasks to a heterogeneous computing system. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2791–2805.

Pahlavan, A., Momtazpour, M., and Goudarzi, M. (2012). Data center power reduction by heuristic variation-aware server placement and chassis consolidation. In *Proceedings of the 16th CSI International Symposium on Computer Architecture and Digital Systems*, pages 150–155, Shiraz, Fars, Iran. IEEE.

Panda, S. and Jana, P. (2015). Efficient task scheduling algorithms for heterogeneous multi-cloud environment. *The Journal of Supercomputing*, 71(4):1505–1533.

Pop, C. B., Anghel, I., Cioara, T., Salomie, I., and Vartic, I. (2012). A swarm-inspired data center consolidation methodology. In *Proceedings of the Second International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, pages 1–7, New York, NY, USA. ACM.

Pop, F., Dobre, C., Cristea, V., Bessis, N., Xhafa, F., and Barolli, L. (2015). Reputation-guided evolutionary scheduling algorithm for independent tasks in inter-clouds environments. *International Journal of Web and Grid Services*, 11(1):4–20.

Portaluri, G., Giordano, S., Kliazovich, D., and Dorronsoro, B. (2014). A power efficient genetic algorithm for resource allocation in cloud computing data centers. In *Proceedings of the Third IEEE International Conference on Cloud Networking (CloudNet)*, pages 58–63, Luxembourg City, Luxembourg. IEEE.

Prekas, G., Primorac, M., Belay, A., Kozyrakis, C., and Bugnion, E. (2015). Energy proportionality and workload consolidation for latency-critical applications. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, pages 342–355, New York, NY, USA. ACM.

Pretorius, M., Ghassemian, M., and Ierotheou, C. (2010). An investigation into energy efficiency of data centre virtualisation. In *Proceedings of the International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 157–163, Fukuoka, Kyushu, Japan. IEEE.

Rafiei, S. and Bakhshai, A. (2012). A review on energy efficiency optimization in smart grid. In *Proceedings of the 38th IEEE Annual Conference on Industrial Electronics Society*, pages 5916–5919, Montreal, Quebec, Canada. IEEE.

Rao, K. S. and Thilagam, P. S. (2015). Heuristics based server consolidation with residual resource defragmentation in cloud data centers. *Future Generation Computer Systems*, 50:87–98.

RE100 (2016). Bt puts sustainability at the heart of its business in line with the companys purpose to use the power of communications to make a better world. Retrieved on 05/05/2016 from `http://there100.org/btm`.

Rethinagiri, S. K., Palomar, O., Sobe, A., Yalcin, G., Knauth, T., Gil, R. T., Prieto, P., Schneega, M., Cristal, A., Unsal, O., Felber, P., Fetzer, C., and Milojevic, D. (2015). Paradime: Parallel distributed infrastructure for minimization of energy for data centers. *Microprocessors and Microsystems*, 39(8):1174–1189.

Ricciardi, S., Careglio, D., Santos-Boada, G., Sole-Pareta, J., Fiore, U., and Palmieri, F. (2011). Saving energy in data center infrastructures. In *Proceedings of the First International Conference on Data Compression, Communications and Processing*, pages 265–270, Palinuro, Italy. IEEE.

Sampaio, A. M., Barbosa, J. G., and Prodan, R. (2015). Piasa: A power and interference aware resource management strategy for heterogeneous workloads in cloud data centers. *Simulation Modelling Practice and Theory*, 57:142 – 160.

Santos, A. S., Madureira, A. M., and Varela, M. L. R. (2014). An ordered approach to minimum completion time in unrelated parallel-machines for the makespan optimization. In *Proceedings of the Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 72–77, Porto, Portugal. IEEE.

Sharma, N. and Reddy, G. (2015). A novel energy efficient resource allocation using hybrid approach of genetic dvfs with bin packing. In *Proceedings of the Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, pages 111–115, Gwalior, Madhya Pradesh, India. IEEE.

Sheikh, A. and Khan, S. (2005). Integer programming approach for optimal resource allocation in workflow automation design. In *Proceedings of the Ninth IEEE International Multitopic Conference*, pages 1–5, Karachi, Pakistan. IEEE.

Shi, X., Jiang, H., He, L., Jin, H., Wang, C., Yu, B., and Wang, F. (2011). EAPAC: An enhanced application placement framework for data centers. In *Proceedings of the 14th IEEE International Conference on Computational Science and Engineering*, pages 34–43, Dalian, China. IEEE.

Sindhu, S. and Mukherjee, S. (2013). A genetic algorithm based scheduler for cloud environment. In *Proceedings of the Fourth International Conference on Computer and Communication Technology (ICCCT)*, pages 23–27, Allahabad, Uttar Pradesh, India. IEEE.

Singh, G. and Kumar, P. (2014). Self-adaptive task distribution for load balancing using HABACO in cloud. In *Proceedings of the International Conference on Advanced Communication Control and Computing Technologies (ICACCCT*, pages 1563–1567, Ramanathapuram, Tamil Nadu, India. IEEE.

Singh, N. and Rao, S. (2012). Online ensemble learning approach for server workload prediction in large datacenters. In *Proceedings of the 11th International Conference on Machine Learning and Applications*, volume 2, pages 68–71, Boca Raton, FL, USA. IEEE.

Sinha, A. and Chandrakasan, A. (2001). Dynamic power management in wireless sensor networks. *IEEE Design Test of Computers*, 18(2):62–74.

Song, W., Xiao, Z., Chen, Q., and Luo, H. (2014a). Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 63(11):2647–2660.

Song, W., Xiao, Z., Chen, Q., and Luo, H. (2014b). Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 63(11):2647–2660.

Song, Y., Li, Y., Wang, H., Zhang, Y., Feng, B., Zang, H., and Sun, Y. (2008). A service-oriented priority-based resource scheduling scheme for virtualized utility computing. In *Proceedings of the 15th International Conference on High Performance Computing*, pages 220–231, Bangalore, Karnataka, India. Springer Berlin Heidelberg.

Song, Y., Wang, H., Li, Y., Feng, B., and Sun, Y. (2009). Multi-tiered on-demand resource scheduling for vm-based data center. In *Proceedings of the Ninth IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 148–155, Shanghai, China. IEEE.

Statista (2016). Internet of Things (IoT): number of connected devices worldwide from 2012 to 2020 (in billions). Retrieved on 04/06/2016 from `http://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`.

Stephens, R. (2013). *Essential Algorithms: A Practical Approach to Computer Algorithms*. Wiley Publishing, 1st edition.

Stoess, J., Lang, C., and Bellosa, F. (2007). Energy management for hypervisor-based virtual machines. In *Proceedings of the USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA. USENIX Association.

Sueur, E. L. and Heiser, G. (2010). Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the International Conference on Power Aware Computing and Systems*, pages 1–8, Berkeley, CA, USA. USENIX Association.

Sun, G., Liao, D., Zhao, D., Xu, Z., and Yu, H. (2015). Live migration for multiple correlated virtual machines in cloud-based data centers. *IEEE Transactions on Services Computing*, PP(99):1–14.

Tang, C., Steinder, M., Spreitzer, M., and Pacifici, G. (2007). A scalable application placement controller for enterprise data centers. In *Proceedings of the 16th International Conference on World Wide Web*, pages 331–340, Banff, Alberta, Canada. ACM.

Tang, M. and Pan, S. (2015). A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Processing Letters*, 41(2):211–221.

Tao, F., Feng, Y., Zhang, L., and Liao, T. (2014). Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing*, 19:264 – 279.

TechNavio (2015). Global data center outsourcing market 2015-2019. Technical report, Research and Markets.

Tembey, P., Gavrilovska, A., and Schwan, K. (2014). Merlin: Application- and platform-aware resource allocation in consolidated server systems. In *Proceedings of the ACM Symposium on Cloud Computing*, SOCC '14, pages 1–14, Seattle, WA, USA. ACM.

The Green Grid (2013). Awards industry recognition. Retrieved on 05/05/2016 from `http://www.thegreengrid.org/`.

Vaid, K. (2010). Invited talk: Datacenter power efficiency: Separating fact from fiction. In *Workshop on Power Aware Computing and Systems*, Vancouver, British Columbia, Canada.

Vasudevan, M., Tian, Y.-C., Tang, M., and Kozan, E. (2014). Profiling : an application assignment approach for green data centers. In *Proceedings of the 40th IEEE Annual Conference of the Industrial Electronics Society*, pages 5400–5406, Dallas, TX, USA. IEEE.

Vasudevan, M., Tian, Y.-C., Tang, M., Kozan, E., and Gao, J. (2015). Using genetic algorithm in profile-based assignment of applications to virtual machines for greener data centers. In *Proceedings of the 22nd International Conference on Neural Information Processing, Part II*, Lecture Notes in Computer Science, pages 182–189, Istanbul, Turkey. Springer International Publishing.

Verma, A., Ahuja, P., and Neogi, A. (2008). pMapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the Ninth ACM/IFIP/USENIX International Conference on Middleware*, pages 234–264, Leuven, Belgium. Springer-Verlag New York, Inc.

Votaw, D. and Orden, A. (1952). The personnel assignment problem. In *Proceedings of the Linear Inequalities and Programming Symposium*, pages 155–163, Washington, DC, USA. Elsevier Inc.

Wang, X., Wang, Y., and Cui, Y. (2014). A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing. *Future Generation Computer Systems*, 36:91–101.

Wang, Y. and Wang, X. (2014). Performance-controlled server consolidation for virtualized data centers with multi-tier applications. *Sustainable Computing: Informatics and Systems*, 4(1):52–65.

Wang, Z. and Su, X. (2015). Dynamically hierarchical resource-allocation algorithm in cloud computing environment. *The Journal of Supercomputing*, 71(7):2748–2766.

Whitehead, B., Andrews, D., Shah, A., and Maidment, G. (2014). Assessing the environmental impact of data centres part 1: Background, energy use and metrics. *Building and Environment*, 82:151–159.

Whitney, J. and Delforge, P. (2014). *Scaling Up Energy Efficiency Across the Data Center Industry: Evaluating Key Drivers and Barriers (Issue Paper)*. Natural Resources Defense Council (NRDC).

Williams, D., Jamjoom, H., Liu, Y.-H., and Weatherspoon, H. (2011). Overdriver: Handling memory overload in an oversubscribed cloud. *ACM SIGPLAN Notices*, 46(7):205–216.

Winter, J. A. and Albonesi, D. H. (2008). The scalability of scheduling algorithms for unpredictably heterogeneous CMP architectures. In *Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 42–51, Anchorage, Alaska, USA. IEEE.

Wu, G., Tang, M., Tian, Y.-C., and Li, W. (2012). Energy-efficient virtual machine placement in data centers by genetic algorithm. In *Proceedings of the 19th International Conference on*

Neural Information Processing, Part III, volume 7665 of *Lecture Notes in Computer Science*, pages 315–323, Doha, Qatar. Springer Berlin Heidelberg.

Xiao, W., Low, S. M., Tham, C. K., and Das, S. K. (2009). Prediction based energy-efficient task allocation for delay-constrained wireless sensor networks. In *Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, pages 1–3, Rome, Italy. IEEE.

Xu, Y. and Qu, W. (2011). A trust model-based task scheduling algorithm for data-intensive application. In *Proceedings of the Sixth Annual ChinaGrid Conference*, pages 227–233, Liaoning, China. IEEE.

Yang, H., Breslow, A., Mars, J., and Tang, L. (2013). Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers. *SIGARCH Compututer Architecture News*, 41(3):607–618.

Yang, Y., Cao, X., Ju, J., and Chen, Y. (2005). Research on prediction methods for load balancing based on self-adaptive and confirmation mechanism. In *Proceedings of the International Conference on Control and Automation*, volume 1, pages 533–536, Budapest, Hungary. IEEE.

Ye, K., Wu, Z., Wang, C., Zhou, B. B., Si, W., Jiang, X., and Zomaya, A. Y. (2015). Profiling-based workload consolidation and migration in virtualized data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(3):878–890.

Yuan, J., Jiang, X., Zhong, L., and Yu, H. (2012). Energy aware resource scheduling algorithm for data center using reinforcement learning. In *Proceedings of the Fifth International Conference on Intelligent Computation Technology and Automation*, pages 435–438, Zhangjiajie, Hunan, China. IEEE.

Zapater, M., Ayala, J., and Moya, J. (2012). Leveraging heterogeneity for energy minimization in data centers. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 752–757, Ottawa, ON, USA. IEEE.

Zeng, H. and Di Natale, M. (2013). An efficient formulation of the real-time feasibility region for design optimization. *IEEE Transactions on Computers*, 62(4):644–661.

Zhang, Y. and Guo, R. (2014). Power-aware fixed priority scheduling for sporadic tasks in hard real-time systems. *Journal of Systems and Software*, 90:128–137.

Zhao, Q., Xiong, C., and Wang, P. (2016). Heuristic data placement for data-intensive applications in heterogeneous cloud. *Journal of Electrical and Computer Engineering*, 2016:1–8.

Zhu, K., Song, H., Liu, L., Gao, J., and Cheng, G. (2011). Hybrid genetic algorithm for cloud computing applications. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC)*, pages 182–187, Jeju Island, South Korea. IEEE.