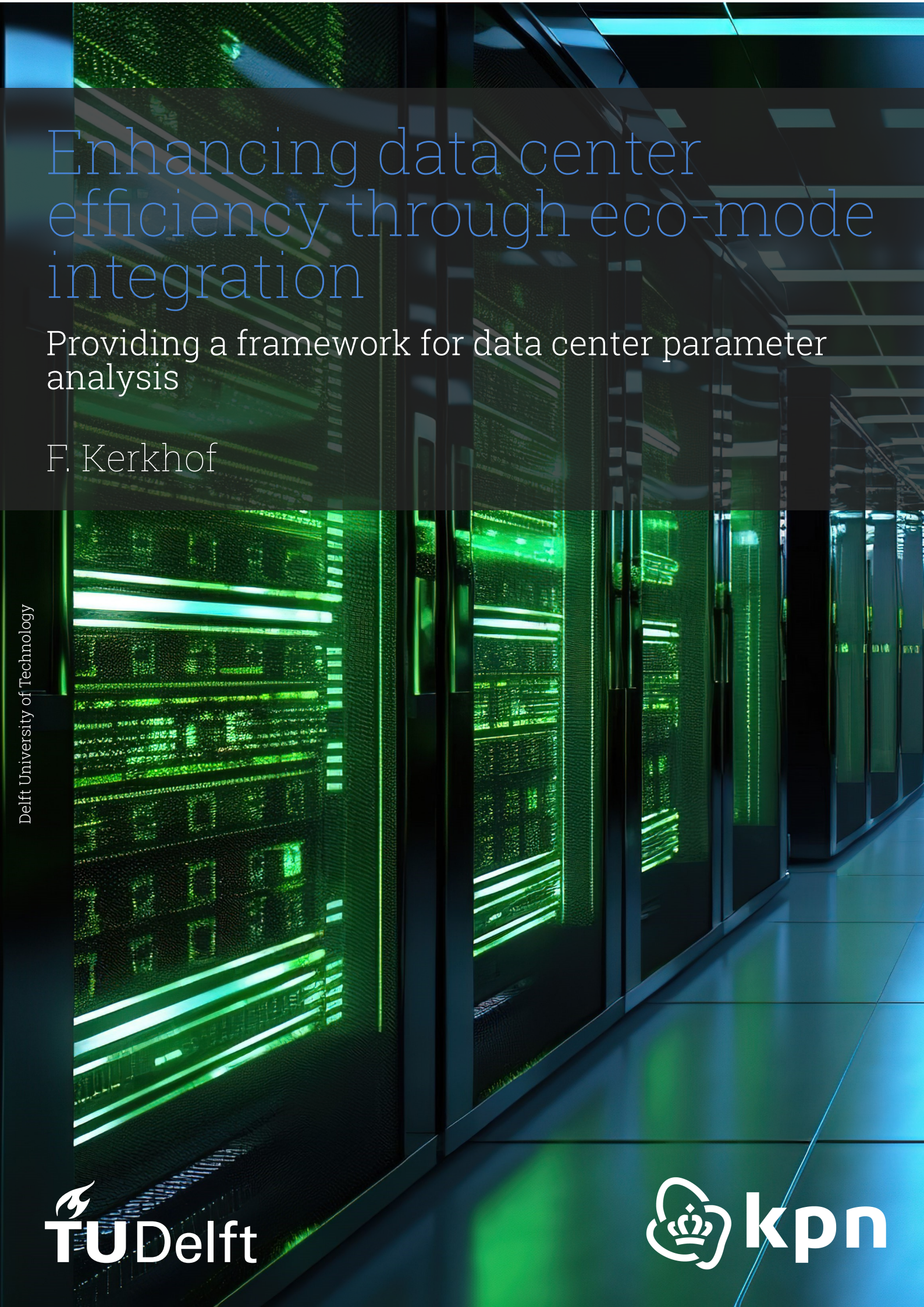

# Enhancing data center efficiency through eco-mode integration

## Providing a framework for data center parameter analysis

F. Kerkhof

Delft University of Technology

TU Delft

kpn

# Enhancing data center efficiency through eco-mode integration

## Providing a framework for data center parameter analysis

by

F. Kerkhof

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on Wednesday, March 6, 2024 at 14:00.

| | | |
|---|---|---|
| Student number: | 4601785 | |
| Project Duration: | February, 2023 - March, 2024 | |
| Faculty: | Electrical Engineering, Mathematics and Computer Science, Delft | |
| Daily Supervisors: | Dr. Ing. E.F.M. van Boven | TU Delft |
| | MSc. D.J. Tuinhof | KPN |
| Thesis Committee | Dr. Ir. E. Smeitink | TU Delft, supervisor |
| | Prof. Dr. Ir. Z. Al-Ars | TU Delft |
| | Drs. J.S. Cox RC | KPN, supervisor |
| | Dr. Ing. E.F.M. van Boven | TU Delft |

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft kpn

# Abstract

The demand for computational resources is increasing exponentially due to an increasing amount of digital services. Cloud computing is becoming the standard for enterprises to provide these resources. This resulted in hyperscalers which consist of a large number of servers. Data centers consume more than 1% of the world's electrical energy. Therefore, many techniques are developed that both help to fulfil the needs of digital services and reduce the energy consumption of data center services. Modern-day servers can switch between different operating states which are often integrated in a power configuration mode of servers known as eco-mode. One of these techniques throttles the clock frequency of a central processing unit (CPU) which enables the possibility to lower the power needed for that CPU. This technique is known as dynamic frequency and voltage scaling (DFVS) and the states it switches between are known as performance states (P-states). A different technique that is integrated with eco-mode is the ability to switch between different idle states of the CPU. These states define whether certain caches of the CPU are flushed or not to conserve energy. These states are known as core states (C-states). A different approach that is focused on conserving energy is virtualisation within data centers. Virtualisation enables one physical server to host multiple virtual instances of servers (virtual machines). This reduces resource wastage and energy consumption of a data center. However, this creates the need for a strategic placement that ensures that the demands of the virtual machines are met and that minimises energy consumption and resource wastage. This thesis analyses four of these techniques: the best fit decreasing (BFD) algorithm, the integer linear programming (ILP) algorithm, the particle swarm optimisation (PSO) algorithm and the genetic algorithm (GA). This thesis provides a framework that uses a holistic approach to provide insights into the effects of using eco-mode of servers within the dynamics of virtual machine placement in data centers. This framework serves as a first step in parameterising the dynamics of a data center regarding its energy consumption and performance. The results show a potential energy reduction of up to approximately 20% with negligible impact on a data center's performance. This result occurs when applying the best fit decreasing algorithm and having a server with an energy-efficient eco-mode. However, this thesis does not cover all parameters that play a role in the data center's performance and energy consumption, so more research on this area is recommended.

# Preface

You are reading the master thesis 'Enhancing data center efficiency through eco-mode integration — Providing a framework for data center parameter analysis', which I wrote to finish my master's program in Electrical Engineering at the Delft University of Technology. I was privileged to be part both of the Network Architectures and Services (NAS) Group at the faculty of Electrical Engineering, Mathematics and Computer Sciences and of KPN.

Firstly, I would like to thank Dr. Ing. Edgar van Boven, who not only helped me set up this research proposal at the TU Delft and KPN but also guided me throughout this project. I really appreciated the guidance throughout the project.

Moreover, I would like to thank MSc. Denice Tuinhof. You helped me change the course when needed and supported me during my whole thesis. I really appreciated our weekly meetings in which you both challenged me and helped me interpret and discuss the results. You ensured that I really enjoyed my time at KPN during my thesis and you have aided me in lifting my thesis to a higher level.

Next to that, I would like to express my gratitude towards Dr. Ir. Eric Smeitink for the critical analysis of my work as a supervisor. You helped me a lot during our meetings and you thought along with my approach pushing me towards a higher level.

Furthermore, I would like to thank Drs. Jeroen Cox, who provided me the opportunity to research this very interesting subject and the opportunity to do this in cooperation with KPN. Thank you for helping me look at this research from a different angle and helping me set up this research.

Moreover, I would like to thank Prof. Dr. Ir. Zaid Al-Ars for completing my thesis committee and for taking the time to evaluate this work and be part of my thesis defence adding his expertise to my evaluation.

Lastly, I would like to thank all my family, friends and colleagues at KPN for always being there when I needed support and giving me the distractions I sometimes needed. Especially Alexandra as she was doing her master's thesis alongside me at KPN making the last year very enjoyable.

*F. Kerkhof*
*Delft, February 2024*

# Contents

# Acronyms

**ACPI** advanced configuration and power interface. 4, 6

**BFD** best fit decreasing. i, 8, 9, 11, 25, 26, 28–30, 34, 35, 37, 39

**C-states** core states. i, 4–8, 19, 30, 36, 39
**CPU** central processing unit. i, 4, 6, 7, 20, 22, 24, 38

**DFVS** dynamic frequency and voltage scaling. i, 4, 7, 8, 17, 39, 40

**GA** genetic algorithm. i, v–viii, 8, 10, 12, 14–16, 26, 29, 30, 34, 35, 37, 39, 52

**HVAC** heating, ventilation and air conditioning. 4

**ILP** integer linear programming. i, 8–10, 15, 21, 26–30, 34, 35, 37, 39, 49
**IT** information technology. vii, 4, 8

**MIPS** millions instructions per second. vii, 20, 21, 23–25, 38

**P-states** performance states. i, 4–8, 17, 19, 30, 35, 36, 39
**PSO** particle swarm optimisation. i, vi–viii, 8, 10–16, 26–30, 34, 35, 37, 39, 52
**PUE** power usage effectiveness. 4

**RAM** remote access memory. vii, 18, 19, 38

**VM** virtual machine. i, v–viii, 1–3, 5, 8–15, 17–41
**VMM** virtual machine manager. v, 3, 18–20

# List of Figures

# List of Tables

# List of Symbols

| Symbol | Definition | Unit |
|---|---|---|
| $b_i$ | Current network bandwidth request of VM $i$ | [Mb/s] |
| $B_j$ | Network bandwidth capacity of host $j$ | [Mb/s] |
| $c$ | Static power dissipation of server | [W] |
| $c_c$ | Cognitive coefficient | |
| $c_i$ | Current computational power request of VM $i$ | [MIPS] or [Hz] |
| $c^i_{t,a}$ | Allocated computational power for VM $i$ at time $t$ | [MIPS] or [Hz] |
| $c^i_{t,r}$ | Requested computational power for VM $i$ at time $t$ | [MIPS] or [Hz] |
| $C_j$ | Computational power capacity of host $j$ | [MIPS] or [Hz] |
| $c_s$ | Social coefficient | |
| $C$ | Switched load capacitance | [F] |
| $E$ | Total energy consumed by servers during simulation | kWh |
| $E_{DC}$ | Total energy consumed by data center | [kWh] |
| $E_{gain}$ | Relative decrease of energy consumption (due to eco-mode) | |
| $E_{IT}$ | Total energy consumed by IT equipment of data center | [kWh] |
| $f$ | Operating frequency | [Hz] |
| $f_i$ | Fitness score of particle $i$ (PSO) OR parent $i$ (GA) | |
| $\mathcal{H}$ | Set (or list) of hosts | |
| $I_{handled}$ | Number of handled instructions | [MI] |
| $m_i$ | Current RAM request of VM $i$ | [MB] |
| $M_j$ | RAM capacity of host $j$ | [MB] |
| $N_p$ | Number of particles (PSO) OR parents (GA) | |
| $N_t$ | Number of iterations (PSO) OR generations (GA) | |
| $p$ | Exponent of dynamic power dissipation of server | |
| $p_i$ | Probability of selecting parent $i$ | |
| $\mathcal{P}$ | Set (or list) of particles (PSO) OR parents (GA) | |
| $P(u)$ | Power dissipation of server at utilisation $u$ | [W] |
| $P^j$ | Current power dissipation of host $j$ | [W] |
| $P^{CPU}$ | Power dissipation by processor of a server | [W] |
| $P^{CPU}_{static}$ | Static component of power dissipation by processor of a server | [W] |
| $P^{CPU}_{dynamic}$ | Static component of power dissipation by processor of a server | [W] |
| PUE | Power usage effectiveness | |
| $r_c$ | Random factor for cognitive value | |
| $r_s$ | Random factor for social value | |
| $s_i$ | Current storage request of VM $i$ | [MB] |
| $S_j$ | Storage capacity of host $j$ | [MB] |
| $SLAV$ | Fraction of missed instructions | |
| $T$ | Time frame of the simulation | [minutes] |
| $T_{sim}$ | Total simulation time | [minutes] |
| $u$ | Utilisation level of server | |

| Symbol | Definition | Unit |
|---|---|---|
| $V$ | Gate voltage | [V] |
| $\mathcal{V}$ | Set (or list) of VM's | |
| $V_i^p$ | Velocity vector of particle $p$ at iteration $i$ | |
| $w$ | Inertia coefficient (PSO) OR survival coefficient (GA) | |
| $x_{ij}$ | Binary variable that checks whether VM $i$ is placed on host $j$ | |
| $X_i^p$ | Position vector of particle $p$ at iteration $i$ | |
| $\alpha$ | Activity factor | |
| $\gamma$ | Dynamic power dissipation constant of server | [W] |
| $\phi_h$ | Binary variable that checks whether host $h$ is active | |
| $\mu$ | Energy-performance efficiency | [Wh/Ml] |

# 1 Introduction

In the contemporary digital era, data centers serve as the foundational infrastructure supporting our interconnected world. These centralized facilities manage, process, and store the vast volumes of data generated by modern technologies, playing a crucial role in enabling applications like cloud computing, artificial intelligence, and connectivity. As the nerve centers of the global digital ecosystem, data centers ensure the seamless functioning of diverse services while also facing scrutiny for their environmental impact.

## 1.1. Trends in data center services

The current era can be marked by the proliferation of digital services and the technological advancements made herein. This is due to the high demand for data storage and computing processes. These storage and computing elements are most of the time placed in data centers, and thus the data center service sector is growing.

This also creates demand for larger data centers such as hyperscalers. Hyperscalers are large data centers provided by companies such as Google, Amazon and Microsoft. They provide servers not only for themselves but also for other companies (and consumers) of which they can rent their resources partly. This type of data center service is known as cloud computing.

Renting only a part of the resources of a server is possible due to virtualisation. Virtualisation is the process of creating a virtual alternative. Within the context of data centers, servers can be virtualised by creating so-called virtual machines (VMs). Virtual machines emulate an operating system while running on the software of a different operating system. This technology allows us to have multiple operating systems on one physical server. To a further extent, it allows us to have multiple servers on one physical server.

Cloud computing is not only provided by large companies such as Google, as some sectors still operate data centers themselves. Telecom operators, such as KPN, often have their own data centers to have their service more in line with their own needs. This still uses cloud computing principles. These operators manage their data centers for themselves, and also some of their clients. Although such companies manage their own data centers, some of them are shifting more towards hyperscalers as the complexities of managing a data center keep increasing and are becoming very costly.

One of the drivers for the operational costs of a data center is the electrical energy consumption by data centers. In 2022 the energy consumed by data centers (excluding data transmission networks and cryptocurrency mining) was 240-340 TWh, which is 1-1.3% of the global electrical energy demand [1]. This issue is addressed within the Netherlands too, since 2.3% of the electricity consumption is due to its data centers [2].

The aforementioned virtualisation techniques are used to reduce the electrical energy consumption by data centers. Virtualisation results in a better carbon footprint and a reduction of both operational costs and investment costs. Operational costs can be reduced since it will reduce the energy consumption of a data center. The investment costs can be reduced as fewer servers are needed since one physical server can handle the workload of multiple applications.

Section 1.3 explains more about the contributions to the reduction of energy consumption of data centers and what components are responsible in what manner for this energy consumption. Before that, Section 1.2 provides an overview of the architecture of a data center and its functionalities.

## 1.2. High-level data center architecture

This section expounds on the physical building along with the challenges faced by a data center on different levels. Figure 1.1 shows a schematic overview of a data center.

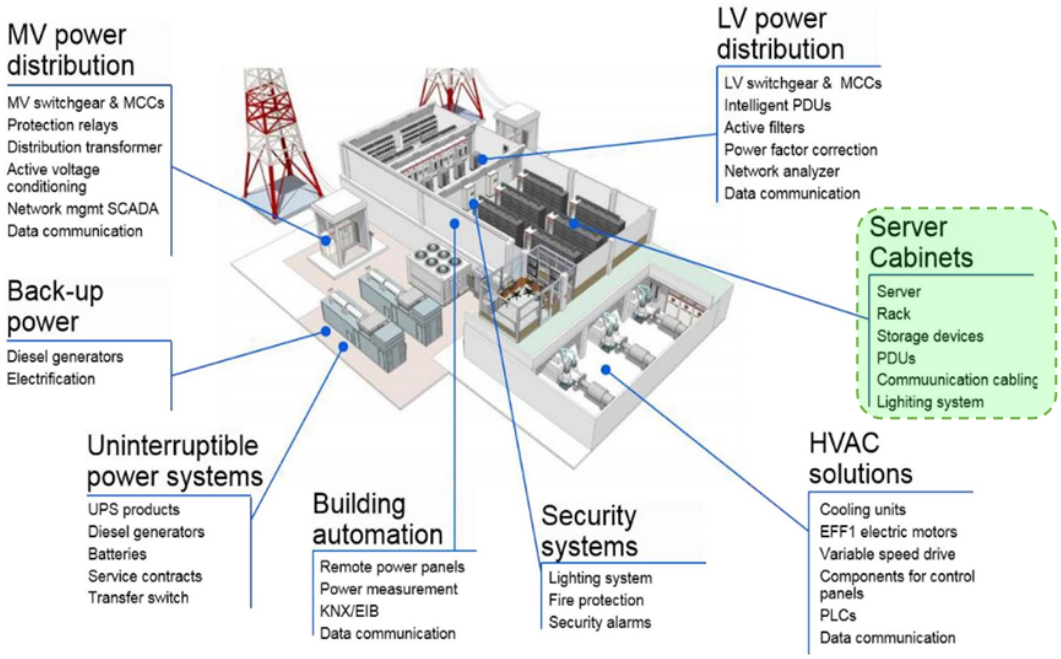


**Figure 1.1:** Overview of a data center [3]

This figure shows the many components needed to keep a data center running. The highlighted section, the server cabinets, is part of a data center where the core functionality is placed. All the other components are auxiliary components to keep a data center running and secure from any threats.
As you can see, a data center needs to run its power distribution on multiple levels. Once the power is distributed, it needs some systems that provide stable power to all the systems in the form of uninterruptible power systems. Moreover, some backup power is supplied, which often are diesel generators.
The core functionality is provided by the servers, storage and network elements. This is the highlighted part in Figure 1.1. Of these core functionality components, this research focuses on the servers. Servers can host many user applications. An application can consist of multiple virtual machines. A virtual machine is a software emulation of a physical computer (or server). A virtual machine has, just like a physical server, an operating system and specific applications that run on that virtual machine. A user application can often be divided into sub-applications that could function better on different operating systems. Virtual machines are a suitable solution for this.
One physical server can host multiple VMs. The amount of VMs it can host depends on the resources needed by the VMs and the resource capacity that the server has. Resources that define both a host and a VM are: computational power, memory, network bandwidth and internal storage.
In a data center, the VMs created by all its users need to be placed on the servers that the data center has. These servers have often been clustered for performance purposes. Figure 1.2 shows a functional schematic of a data center, including the VM placement. The highlighted part illustrates the VM placement, whereas the rest of the figure also shows the power supply and other supportive systems. Note that this figure does not explicitly include storage devices.
The data center has rooms which are full of racks and within these racks, servers are placed. These racks are therefore often referred to as server racks, whilst they also contain network elements and possibly storage devices. These racks are connected through the network elements and often one rack is full of network elements to connect that cluster of racks to other clusters.

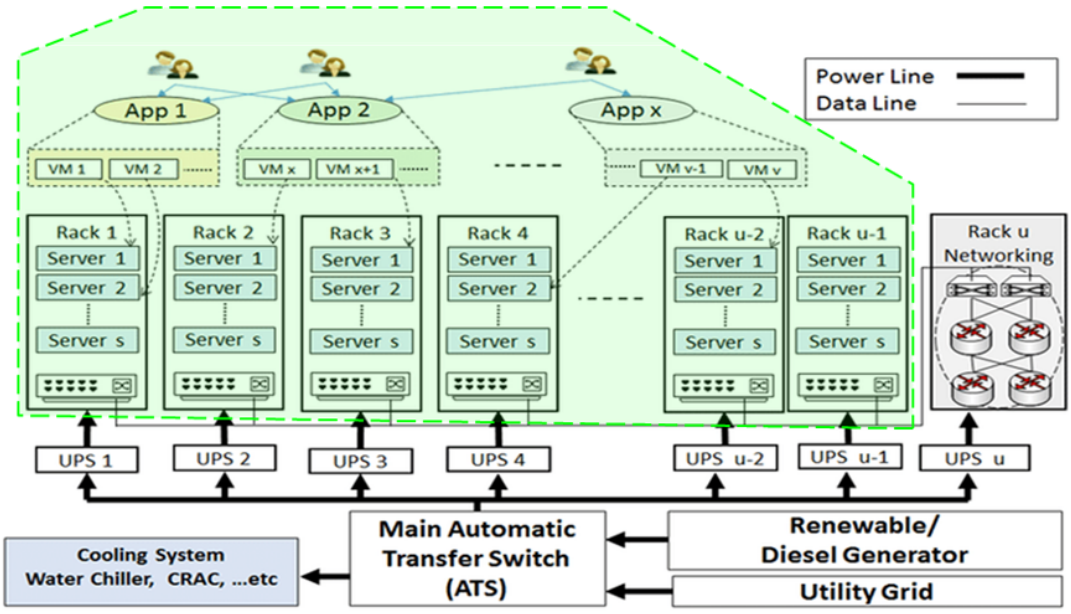


**Figure 1.2:** Functional schematic of data center [4]

In this way, each server is connected with any of the other servers, meaning that the VMs can be placed on any of the servers within a data center. These VMs can not only be placed on any server, but they could also migrate to any other server when needed, for example when a server is shut down to conserve energy.
The process of placing and migrating VMs on servers is handled by the virtual machine manager (VMM). A VMM is provided by a hypervisor. A hypervisor is similar to an operating system that ensures the possibility to run multiple VMs on one physical server. A simplified version of the functionality of a VMM is displayed in Figure 1.3

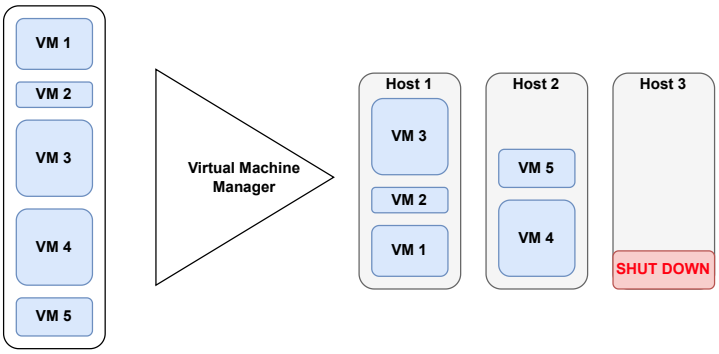


**Figure 1.3:** Simplified functionality of virtual machine manager

The VMM ensures that the resource demands of all VMs are fulfilled by the servers these VMs are placed on. Often, there are multiple different placements possible to fulfil these needs.
Providing the resources to the VMs is the core functionality of the VMM, but other objectives play a role in the placement too. One of the objectives of the VMM is to minimise energy consumption. Figure 1.3 shows an example of an efficient placement so that fewer hosts are needed. This would result in a better carbon footprint of a data center and less operational costs for the data center operator.
Another is to minimise resource wastage. This objective in practice is closely related to the objective of minimising energy consumption. Having less resource wastage results in needing fewer servers in total, which is also shown by Figure 1.3. Needing fewer servers leads to lower investment costs for the data center operator.

## 1.3. Data center energy consumption

As aforementioned, data centers consume a large portion of the world's electrical energy. One way to have a green(er) data center is to use sustainable energy sources. This mainly reduces the carbon footprint of data centers but does not influence the energy consumption within data centers. However, a much larger impact can be made by reducing the electrical energy consumption of a data center.
To give a measure to which portion of the electrical energy is consumed by the core functionality of a data center and which portion is consumed by supportive systems, such as heating, ventilation and air conditioning (HVAC), it is common to calculate the power usage effectiveness (PUE) of a data center. The PUE of a data center can be calculated using the following equation [5]:

$$\text{PUE} = \frac{E_{DC}}{E_{IT}} \tag{1.1}$$

Here $E_{DC}$ is the total energy consumed by the data center as a whole and $E_{IT}$ is the energy consumed by all the IT equipment within the data center. The IT equipment consists of the servers, storage devices and networking devices. Examples of other components that contribute to the energy consumption of a data center which is not part of the IT equipment are HVAC systems, security systems and lighting.
The PUE values range between one and infinity, with one meaning the data center is marked as energy-efficient since all of the energy consumed is used for the functional components of a data center.
To decrease the PUE, much research has been conducted on decreasing the energy consumption by HVAC systems, since these systems were the highest contributors to the high energy consumption from the supportive systems. This resulted in a quick decline of the worldwide average PUE between 2007 and 2018 from 2.5 to 1.58 [6].
After this period, the PUE value has stagnated. One of the reasons is that the focus of research has shifted towards reducing the energy consumption of IT equipment. By reducing the energy consumption of the IT equipment whilst the energy consumption of the supportive systems remains constant, the PUE value will increase. The stagnation thus means that the improvement regarding the energy consumption of the IT equipment and HVAC systems produces similar results.
Marcacci [7] categorised the energy consumption of a data center into four main categories: Servers, network elements, storage devices and supportive systems. The first three are part of the IT equipment and form the core functionality of a data center. The research by Marcacci shows that the greatest share of electrical energy consumed is by the servers and by the support systems.
Thereafter, more research was condoned on reducing the energy consumption of servers, as reducing the energy of HVAC systems had already been researched extensively. Another reason to shift the research focus to servers specifically is that Zoie et al. [5] have shown that increasing the workload to minimise the number of servers needed reduces the energy consumption of the data center as a whole more than trying to decrease the PUE value of a data center by improving the supportive systems.
Reducing the energy consumption of servers can be achieved by configuring these servers different power states. These states are defined by advanced configuration and power interface (ACPI) [8]. Energy-saving techniques are mostly based on switching between these different power states. However, these power states also affect the performance of a server.
The advanced configuration and power interface defines different types of states which affect the server on different hardware levels. One of the most commonly known states are the core states (C-states). These states enable to have one (or more) core(s) of a central processing unit (CPU) on hold, meaning that it does not execute any instructions. This technique helps to conserve energy for idle servers.
On a deeper hardware level, there are so-called performance states (P-states). These states affect the clock frequency of an individual (or multiple) core(s) in a CPU. A higher clock frequency requires a higher gate voltage and thus draws more power. Thereafter, specific frequency-voltage pairs are created to tune the computational power and the power dissipation by a core. These pairs are the so-called P-states. The process of switching between these pairs is known as dynamic frequency and voltage scaling (DFVS).
Modern-day servers are all able to configure these states. Since the workload is very dynamic, the functionality of these states can be exploited when they can be switched dynamically to match the workload. Thereafter, modern-day servers developed a so-called eco-mode which dynamically switches between both the C-states and P-states.

## 1.4. Research objective

This research combines two commonly researched areas within this scope: virtual machine (VM) placement and server power management techniques. The latter is commonly known as the eco-mode of servers. This research delves deeper into the techniques used for this and uses a holistic approach to integrate this technique within the higher data center levels, meaning that eco-mode is applied on a cluster level.
This research is done in cooperation with KPN as a follow-up on the LEAP [9] project by the Dutch government. Eco-mode is often unused within data centers since the operators have no insights into the impact on the data center's performance. The fear of performance degradation results in data center operators keeping all servers configured with high-performance mode. However, with an ambition to become climate-neutral, the energy consumption needs to be reduced.

## 1.5. Research questions

This research provides a framework that provides insights into the influences of these techniques within the context of data centers when combined and it illustrates how their design choices affect each other. This framework is provided along with the answering of the following research questions:

**RQ1** What key aspects should be considered within eco-mode techniques to conserve energy without experiencing unacceptable performance degradation on a cluster level?

a. How will the C-states of individual servers affect the energy consumption on cluster level?
b. How will the C-states of individual servers affect the performance on cluster level?
c. How will the P-states of individual servers affect the energy consumption on cluster level?
d. How will the P-states of individual servers affect the performance on cluster level?

**RQ2** How will the design choices for virtual machine placement strategies influence the impact due to the eco-mode techniques?

a. What correlation is there between eco-mode techniques and virtual machine placement strategies regarding the energy consumption of a cluster?
b. What correlation is there between eco-mode techniques and virtual machine placement strategies regarding the performance of a cluster?

**RQ3** How will the design choices for virtual machine resource dimensions influence the impact due to the eco-mode techniques?

a. What correlation is there between eco-mode techniques and virtual machine resource dimensions regarding the energy consumption of a cluster?
b. What correlation is there between eco-mode techniques and virtual machine resource dimensions regarding the performance of a cluster?

## 1.6. Thesis synopsis

The structure of this thesis is as follows. Chapter 2 provides an overview of the literature on the subject. Moreover, it will give the theory behind the techniques that are used. Then, Chapter 3 shows the methodology for the experiments that have been executed to answer the posed research questions. After that, Chapter 4 displays the results of the conducted experiments of which the interpretations are discussed in Chapter 5. The latter also discusses the unexplored parameters of this research. After that, Chapter 6 gives an explicit answer to the posed research questions and addresses any recommendations for further research. This chapter provides recommended practices for data center operators too.

# 2

# Literature Overview

This Chapter provides an overview of the current advancements of research performed on this topic. The structure is as follows. Firstly, in Section 2.1, a detailed analysis of the functionalities that support so-called eco-mode for servers is given. Eco-mode is a power management configuration for servers. Subsequently, the current virtual machine policies and related goals are explained. Lastly, an overview of research regarding combining these research fields is provided.

## 2.1. Functionality of eco-mode in servers

Configuring a server with eco-mode means enabling the capability of changing the core states (C-states) and/or the performance states (P-states) as defined by advanced configuration and power interface (ACPI) [8]. The influence of both of these different types of states is elaborated upon within this research. The states are defined for a trade-off between performance and power of the central processing unit (CPU). Eco-mode is most often designed to reduce the energy consumption of the CPU since this is the component with the largest energy consumption for a server [10]. To have a better understanding of the impact that can be made by these states, it should be known that the power consumed by a processor can be divided into a static power and a dynamic power range [11].

$$P^{CPU} = P^{CPU}_{static} + P^{CPU}_{dynamic} \tag{2.1}$$

The C-states can influence the static power whereas the P-states can mainly influence the dynamic power consumed by the CPU.

### 2.1.1. Core states (C-states)

The C-states influence the power dissipated by the processor when it is (nearly idle) by configuring the processor (or individual cores of a processor) in idle mode. When a processor, or an individual core, is in idle mode, it is not executing any instructions. The operating C-state is referred to as C0, whereas C1 and higher are the actual idle modes. The higher the C-state, the longer the wake-up time from that state to the active state (C0). Moreover, for a higher C-state less energy is consumed by the processor. This means it will mainly affect the static power as described in Equation 2.1.

The aforementioned wake-up time, or latency, is the cause for the notion of the use of eco-mode on a server. Therefore it is important to know the impact of this latency. The latency of switching from higher C-states, such as C6, is less than a millisecond [12]. However, in application services, which often are constructed tree-like, the effect of the latency degradation could have 10 times the impact in the root of this application.

Decreasing the latency degradation as a result of the use of C-states has been analysed a lot. Zhan et al. propose a technique which optimises the number of active cores and has only a subset of the cores that reside in high C-states [13]. Traditionally, cores have the same residency distribution for C-states. The residency distribution defines the fraction of time that a core is in a certain C-state. Their research resulted in less switching between C-states and thus less latency degradation.

Moreover, Yahya et al. [14] provide a C-state architecture that tries to mitigate the effects of latency

for specific latency-sensitive applications. This would pave the way for high-performance servers to benefit from using its C-states as well, whereas currently, this is not the case since the performance degradation as a result of these latencies is too large.
The energy gains as an effect of the C-states are very clear when a server is completely idle. The difference in power can be up to a third compared to when C-states are disabled within a server. Due to the ability to configure the C-state for each core individually, an energy reduction is also seen when the workload on the server increases, as was shown by the research of Beckett [15]. The power dissipated by the server is equal for a server with C-states enabled to that of a server with C-states disabled when the server is operating at full capacity. Beckett [15] has shown that the performance of a server operating at full capacity only decreases by 1%, whilst the energy, even at full capacity, is decreased by 3% (P-states were disabled).
It is noticed that this research only involves tests on one specific server, whereas Rteil et al. [16] have researched the impacts for multiple servers and for different generations of the same server model. Their research emphasises that the energy savings for the average enterprise server can be very significant since the energy consumed in servers with an average CPU utilisation level of lower than 75% is much lower. However, when the average utilisation level is above 75%, which is often the case for high-performance cluster servers, the degradation as a result of the eco-mode on that server becomes too high and it is not worth the small energy savings.
All in all, enabling C-states shows promising opportunities for energy savings, but especially when used in high-performance clusters, which are often configured as tree-like architectures, the degradation in performance as a result of increasing latency may get overhand. The benefits are largest when the CPU utilisation level is relatively low, which for many enterprise solutions in data centers is dominantly the case for their server clusters.

### 2.1.2. Performance states (P-states)

As aforementioned, the C-states mostly affect the static power dissipated by the CPU, as described in Equation 2.1. The performance states (P-states) however, mostly affect the dynamic power dissipated by the CPU. The dynamic power can be approximated by the following equation [17]:

$$P_{dynamic}^{CPU} = \alpha C V^2 f \tag{2.2}$$

Here, $C$ is the load capacitance for the switches on the CPU, $V$ is the voltage applied to the CPU and $f$ is the operating frequency of the CPU. Lastly, $\alpha$ is the activity factor, which is linearly dependent on the workload on that CPU.
Due to the quadratic relationship between the dynamic power and the voltage applied to the CPU, it is interesting to decrease the applied voltage to mitigate the power dissipation. However, when the voltage is lowered, the clock frequency should also be decreased for a stable operation [18]. Thereafter, the performance of a CPU will degrade as execution will take longer for lower clock frequencies.
The exploitation of adapting both the voltage and frequency to the needs of the workload that needs to be handled by CPU is known as dynamic frequency and voltage scaling (DFVS). This technique was first proposed by Weiser et al. [19]. Their results were solely based on simulations but presented potential energy savings of up to 20%, which later was demonstrated to be potentially even higher when considering the energy consumed in idle states (C-states) too by Miyoshi et al. [20]
The ability to throttle the voltage and frequency to conserve energy has been integrated into modern-day CPUs by manufacturers such as AMD and Intel. They have created discrete voltage-frequency pairs for their CPUs, which have been defined as P-states [8].
As aforementioned, Rteil et al. [16] measured possible gains for modern-day servers for different types of workloads when enabling both the C-states and P-states. With their research, they demonstrate that servers can have great benefits from the addition of P-states when the utilisation level of a server does not become too high. The exact value for this utilisation level depends on the server that is used and the workload that is applied on that server. This thesis focuses on CPU-bound workload since this will be most restrictive on potential gains due to enabling eco-mode.
Much research has been conducted on the positive impact that DFVS could achieve on single servers and in some cases for server clusters. Chang et al. [21] demonstrated that the energy efficiency of a server cluster will be enhanced significantly under light workload conditions. Additionally, Wang et al. [22] showed that performance will not degrade significantly for non-critical tasks. Moreover, Huang et al. [23] showed that in a mixed-critical system, energy efficiency can be improved whilst meeting all

the performance requirements regarding delay. Rizvandi et al. [24] even showed substantial energy savings when applying DFVS in a high-performance cluster.
An essential point to consider in these investigations is that, in every case, the workload types and sizes were well-defined. In contrast, DFVS has not been extensively utilised in cloud environments where workloads are highly dynamic and difficult to predict. The dynamic nature of these environments, attributed to hosting applications for numerous users, raises questions about the repercussions of enabling both the C-states and P-states states. This study proposes a framework designed to offer a more comprehensive understanding of the potential advantages and disadvantages associated with these states enabled in a cloud environment.

## 2.2. Virtual machine placement

Virtualisation is the key technology for a cloud environment [25]. Virtualisation enables the possibility to divide the service and resources provided by the physical fundamentals of a server. A virtual machine requests a certain amount of resources of a server and thus makes use of the ability to partition its resources. The number of virtual machines is larger than the number of servers, evidently, and thus a mapping for VMs to these servers is needed. Servers in these environments are thereafter often called hosts. This mapping of VMs is an important process which has a great impact on the energy-efficiency and resource-efficiency of the IT equipment in data centers.
To minimise the power consumption of a data center, a lot of research towards consolidation of VMs has been done to increase the resource utilisation of servers. Consolidation is the process of placing VMs on as few servers as possible. Consolidation of VMs is only functional when the demand for resources by these VMs varies over time.
The process of consolidating VMs to minimise the number of active hosts is often referred to as virtual machine placement. The virtual machine placement problem is a well-known variant of the bin-packing problem [26]. The bin packing problem has been proven to be NP-hard, which also holds for the VM placement problem [27].
Extensive inquiries have been conducted regarding the VM placement problem [28, 29, 30]. Due to the significant diversity in solution approaches, this work selectively emphasises only a subset of them.
The methods that are explained are all mono-objective, as they have been subject to more extensive research [28]. This generalisation extends to the approaches that are covered. Furthermore, the implementation of these methodologies is intended for this research. All methods are designed with a power-aware objective, in line with the objectives of this research.
The upcoming sections of this research unfold a series of optimisation strategies, each with its strengths and applications. Commencing with a best fit decreasing (BFD) algorithm, a greedy approach inspired by a classical bin-packing problem solution. Subsequently, the only deterministic method in the form integer linear programming (ILP) algorithm is discussed. This is a mathematical approach convenient for tackling intricate decision problems. Furthermore, particle swarm optimisation (PSO) is highlighted, leveraging principles of collective behaviour for problem-solving. The final section is dedicated to the genetic algorithm (GA), a heuristic approach inspired by natural selection.

### 2.2.1. Best Fit Decreasing

The best fit decreasing algorithm represents a modification of the first fit decreasing algorithm, which, in turn, is a refinement of the original first fit algorithm. The functioning of the first fit algorithm speaks for itself since it will find the first possible bin in which the item fits. To place this in the context of the VM placement problem, the algorithm will search through the already active hosts for a VM that can host that VM. When it finds a suitable host, the VM is placed on that host. When no suitable host is found, it will power on a new host. This is a rather fast method, but in many cases leads to a sub-optimal solution.
Dósa and György [31] have proven that the first fit algorithm can be improved by ordering the items according to their size in decreasing order. To put this in the context of the VM placement problem, it sorts the VMs that need to be placed on a (new) host according to their computational power request. Then it will first find a placement for the VM with the largest request in computational power.
Additionally, Dósa et al. [32] proposed another alternative which involves finding the best fit. This best fit can be different types of criteria. In their proposed method, the algorithm searches for the fit in the bin that is already fullest. However, the criteria can be anything else, depending on the problem.

The best fit decreasing algorithm is a combination of these two algorithms. This algorithm thus searches a placement for the largest item according to the best-fit criteria. Beloglazov et al. [33] proposed a best fit decreasing solution within the context of the VM placement problem. Firstly, their algorithm sorts the VMs according to their request in computational power in decreasing order. Then it finds the best fit according to the estimated increase in power dissipation due to placing that VM in that host. The best fit is thus the fit with the least expected increase in power. The heuristic for this algorithm can be described as follows:

1. Sort all VMs in decreasing order according to their request in computational power;
2. Keep a list of active hosts;
3. Find the host with the least expected increase in power;
   - If it fits at no host, power on a new host to place the VM on.
   - If a suitable host is found, place the VM on that host.
4. Repeat step 3 for the next VM, until all VMs have been placed.

#### Implementation of BFD algorithm

The BFD algorithm as implemented by Beloglazov et al. [33] has been used for this research. This greedy algorithm first sorts the VMs that need to be migrated according to their current request in computational power descendingly. When the VMs are sorted, it thus searches a new host for the VM with the largest computational request.
Searching for a new host is done according to only one criterion: the increase in power when placing the VM on that host. The algorithm checks this increase for each host and it will place the VM on the host with the smallest increase in power.
Then the BFD algorithm does the same for the next VM (from the sorted VM list of VMs that needs to be migrated) until the algorithm has found a new host for each of the VMs that need to migrate. The execution of this algorithm is captured in Algorithm 1.

**Algorithm 1** Best Fit Decreasing Algorithm

**Input**: $\mathcal{V}_{migrate}, \mathcal{H}_{excluded}$
**Output**: **Map**(host,vm)

1: **sort** $\mathcal{V}_{migrate}$ ▷ Descendingly on computational power
2: **for** $vm \in \mathcal{V}_{migrate}$ **do**
3: $minPower =$ ***MAX VALUE***
4: $newHost =$ ***null***
5: **for** $host \in \mathcal{H}$ **do**
6: **if** $host \in \mathcal{H}_{excluded}$ **then continue**
7: **if** $vm$ fits on $host$ **then**
8: $powerDifference = host.getPowerAfterAllocation(vm) - host.getCurrentPower()$
9: **if** $powerDifference < minPower$ **then**
10: $power = powerDifference$
11: $newHost = host$
12: **else**
13: **continue**
14: $migrationMap.add(newHost, vm)$
**return** $migrationMap$

Notice that for the input, a set of excluded hosts is added ($\mathcal{H}_{excluded}$). This set consists of the over-utilised hosts. This is done to decrease the computational complexity of the algorithm and is therefore also implemented for all of the other VM placement algorithms.
Moreover, a set of VMs that need to be migrated is used ($\mathcal{V}_{migrate}$) since only placement for these VMs is searched for, instead of searching a new optimal placement for all active VMs.

### 2.2.2. Integer Linear Programming

Integer linear programming (ILP) is an optimisation technique for which some or all of the variables are strictly integers and where both the objective function and the constraints are linear equations. When

some of the variables used are not strictly integers, this method is often referred to as mixed-integer linear programming. For the extent of the VM placement problem, a stochastic nature is added to the problem as the utilisation levels of the VMs can be modelled as a stochastic process.
ILP is a classical approach to solving the bin-packing problem and it has been applied to minimise the active number of hosts used in the context of the VM placement problem [34]. A very important drawback of this method is the computational complexity of this algorithm.
The run-time of ILP grows double exponentially with an increasing number of constraints and decision variables [35]. Over the years, new algorithms have been presented showing improved run-time [36, 37]. However, the run-time still grows exponentially as a result of the constraints.
Since the use of a integer linear programming algorithm in the context of VM placement problem, the objective often was minimising the resource wastage, as was also done by Gupta et al. [34]. Their research illustrates that the run-time can be reduced by restructuring the problem into two stages. This caused researchers to alternate the formulation of the problem into smaller sub-problems which are often presented as multi-objective ILP solutions.
Regaieg et al. [38] propose a two objective model. This model first poses an ILP formulation that solves the maximum number of requests and uses the solution of that as an input for the ILP formulation that minimises the number of active hosts. In this way they decrease the number of constraints for the second stage, the decrease their run time.
Later, Regaieg et al. extend their research with an additional stage in between the aforementioned stages [39]. In this stage, they use the input of the first stage to minimise the resource wastage, which then is used as input for the stage that minimises the number of active hosts.
Even though the run-time becomes more competitive with many of the (meta-)heuristic algorithms, they still have a very large run-time. To decrease the run-time, meta-heuristic solutions have been proposed. Two of these are particle swarm optimisation (PSO) and the genetic algorithm (GA), which is elaborated upon in the forthcoming sections.

### Implementation of ILP algorithm

The objective function used in this research aims to minimise the number of active hosts. The objective function is formulated as follows:

$$\min \sum_{j \in \mathcal{H}} \phi_j, \qquad \phi_j = 1 \text{ iff host } j \text{ is active, else } 0 \tag{2.3}$$

The constraints consist of multiple components. Firstly, the Equations 2.4 hold, since these describe if the VMs resource requests are possible with the capacity limit of each host.

$$\sum_{i \in \mathcal{V}} x_{ij} c_i \leq C_j, \qquad j \in \mathcal{H}; \tag{2.4a}$$

$$\sum_{i \in \mathcal{V}} x_{ij} m_i \leq M_j, \qquad j \in \mathcal{H}; \tag{2.4b}$$

$$\sum_{i \in \mathcal{V}} x_{ij} b_i \leq B_j, \qquad j \in \mathcal{H}; \tag{2.4c}$$

$$\sum_{i \in \mathcal{V}} x_{ij} s_i \leq S_j, \qquad j \in \mathcal{H}. \tag{2.4d}$$

Here, $x_{ij}$ is one and only one if VM $i$ is placed on host $j$, $c_i$ is the current computational request of VM $i$, $m_i$ is its current memory request, $b_i$ its current network bandwidth request and $s_i$ its storage request. Similarly, $C_j$ is the computational power capacity of host $j$, $M_j$ its memory capacity, $B_j$ its network bandwidth capacity and $S_j$ its storage capacity. When one of these conditions is not met, the new VM can not and will not be placed on that host. When no suitable host was found for that VM, it will remain on its current host, even if this would lead to an over-utilised host and thus a performance degradation. Note that for this research, only condition 3.1a is relevant.
Moreover, some additional constraints are added, to ensure the functioning of the algorithm. Firstly the following set of constraints is added:

$$\sum_{j \in \mathcal{H}} x_{ij} = 1, \qquad i \in \mathcal{V} \tag{2.5}$$

This equation ensures that each VM is placed on one host exactly. Thus each VM can not be placed on more than one host and it is always placed on at least one host. The latter part of this statement is possible in the scope of this research since it will always be possible to place all of the VMs. This means that the total capacity of all hosts always suffices to place all of the VMs. This is shown in Section 3.2.3. This is done since this research does not focus on data centers or server clusters that are close to their capacity limit.
Furthermore, the following set of constraints is added:

$$1 \geq \phi_j \geq x_{ij}, \qquad i \in \mathcal{V}, j \in \mathcal{H} \tag{2.6}$$

These constraints ensures that $\phi_j$ is one if at least one VM is placed on host $j$ (for all hosts).
These form all the constraints. The total number of constraints account for: $|\mathcal{H}|$ constraints due to Equation 3.1a, $|\mathcal{V}|$ constraints due to Equation 2.5 and $|\mathcal{H}||\mathcal{V}|$ constraints due to Equation 2.6. The total number of constraints is dominated by Equation 2.6.
To decrease the number of constraints, the subsets $\mathcal{V}_{migrate}$ and $\mathcal{H}_{available} = \mathcal{H} \backslash \mathcal{H}_{excluded}$ replace the sets $\mathcal{V}$ and $\mathcal{H}$ respectively. However, to do this, some adaptations to the aforementioned constraints should be made.
Firstly, the capacity of the hosts that already host one or more VMs should be decreased to the available capacity. This ensures that the algorithm does not force hosts to become over-utilised.
Secondly, for all hosts that already are active due to the hosting of at least one VM, which thus are not to be migrated, the respective variable $\phi_j$ becomes constant and set to 1. This ensures that hosts with no VMs, and thus a larger available capacity, are not preferred over hosts that already host at least one VM.
The algorithm thus minimises the active number of hosts and does not take into consideration the power curve of the hosts it uses. For the implementation of this algorithm, the OR-Tools suite provided by Google has been used [40].

### 2.2.3. Particle Swarm Optimisation

A popular meta-heuristic solution is the use of particle swarm optimisation (PSO). This method was first proposed by Kennedy and Eberhart [41] and was introduced to the VM placement problem by Dashti and Ramani [42]. Their results show energy savings of at least 14% when compared to the BFD algorithm as proposed by Beloglazov et al. [33]
The original intention was to simulate social behaviour regarding the movement of organisms such as a fish school [43]. In later studies, it has proven to be a solution for a wide variety of applications [44].
The basic principle of the PSO algorithms consists of having candidate solutions, called particles, that move around in the search space to find an overall optimal solution. The movement of these particles is described in simple equations which are based on the particle's best-known solution as well as the overall best-known solution [45]. Moving to new positions is done iteratively for a fixed number of iterations, or when a tolerable solution has been found [46].
The movement of the particles can be described using the following equations:

$$X_{i+1} = X_i + V_i \tag{2.7a}$$

$$V_{i+1} = wV_i + c_c r_c(b - x_i) + c_s r_s(g - x_i) \tag{2.7b}$$

Here $X_i$ and $V_i$ are the particle's position and velocity, respectively, at the $i$th iteration. Furthermore, $b$ represents the best-known solution by that individual particle, whereas $g$ represents the overall best-known solution. Moreover, $w$ is the inertia coefficient which is set by the practitioner, which also holds for the so-called cognitive coefficient ($c_c$) and the social coefficient ($c_s$). These hyper-parameters have a great impact on the performance and thus have to be researched for their use case. Lastly, $r_c$ and $r_s$ are two random picked values from a uniform distribution between 0 and 1.
The inertia coefficient describes the influence of the particle's current velocity and should always be valued between 0 and 1 to prevent divergence [47]. For values closer to 1, the focus of the algorithm is on exploring the search space whereas for values closer to 0, the algorithm converges faster to its local and/or global optimal solution. Thereafter, a common practice is to decrease the inertia coefficient over time.
The cognitive and social coefficients have a similar role as the inertia coefficient. However, especially

the ratio between the values is important. When the cognitive coefficient is larger than the social coefficient, the algorithm explores the search more, whereas it converges faster when the social coefficient is larger than the social coefficient. When both values are very high, the effect of the inertia diminishes. Research has shown it is better to have the values of these coefficients between 1 and 3 [47].
To implement the PSO algorithm, the hyper-parameters should be carefully set, depending on the objective of the algorithm. Multiple researchers have tried to find optimal configurations for the use of PSO algorithm to minimise the power consumption within data centers [48, 49]. This means that the local- and global best-known solutions are based on the estimated power consumption of a data center. This objective function is often referred to as a fitness function. Both researches show promising results and therefore, an implementation of the PSO algorithm is used for the cause of this research.

### Implementation of PSO algorithm

The PSO algorithm is one of the two meta-heuristic algorithms that have been implemented for this research. The other one is the genetic algorithm (GA) and they share some similarities in configuration to keep the comparison as fair as possible.
Firstly, a set of particles needs to be created. This set consists of 30 particles in this implementation. Each particle consists of two characteristics: a position and a velocity. The position is defined as the proposed mapping by that particle. The position thus could be the final solution for the mapping of each VM to a new host. This mapping for the particle's position is shown in Figure 2.1. The velocity is defined as a vector that represents the change of position and thus implicitly refers to a different mapping, see Figure 2.2.

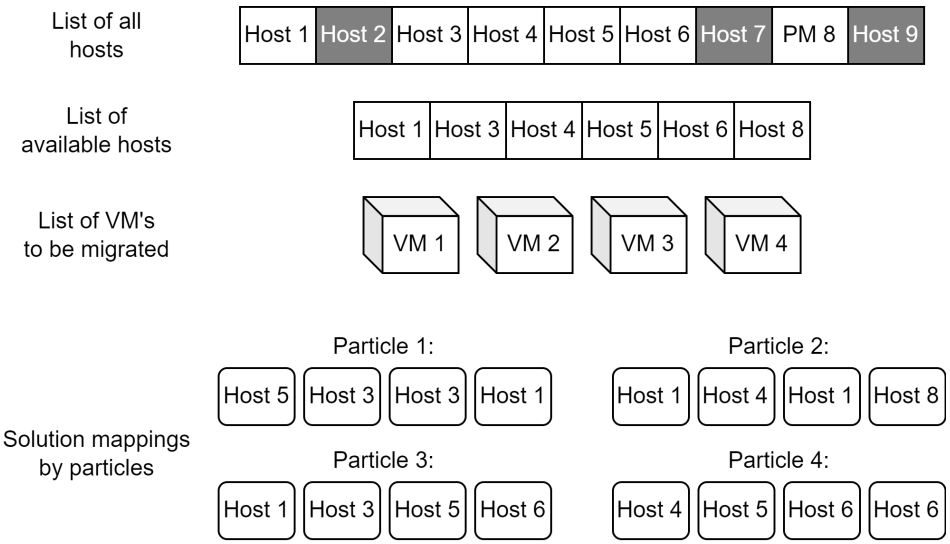


**Figure 2.1:** Visualisation of VM placement mapping

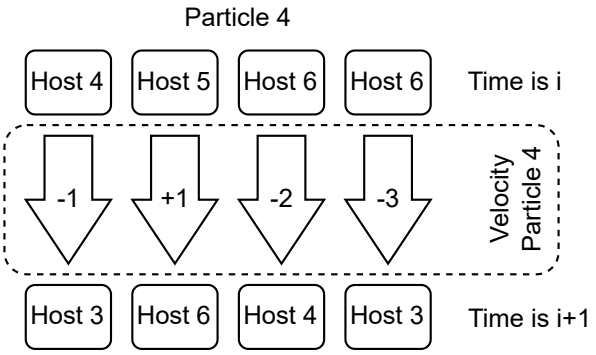


**Figure 2.2:** Visualisation of the velocity of particle 4 (Figure 2.1)

As can be seen in Figure 2.1, the dimensions of each particle's solution are equal to the number of VMs that need to be migrated. The boundaries of the particle's position are determined by the number of available hosts. As for the example shown in Figure 2.1, each dimension can have 6 discrete values. This thus creates a solution space consisting of $|\mathcal{H}_{available}|^{|\mathcal{V}_{migrate}|}$ solutions, where $\mathcal{H}_{available} = \mathcal{H} \backslash \mathcal{H}_{excluded}$ and $\mathcal{V}_{migrate}$ is the set of VMs that need to be migrated.
Each particle is initialised with a position and velocity randomly, uniformly distributed over the whole solution space. Note that the position of each dimension is bounded between 0 and $|\mathcal{H}_{available}|$, whereas the velocity of each dimension is bounded between $-|\mathcal{H}_{available}|$ and $|\mathcal{H}_{available}|$.
Then the fitness for each of these particles is calculated. The fitness of particle $i$ is calculated as follows:

$$f_i = \sum_{j \in \mathcal{H}} P_j \tag{2.8}$$

Here $P_j$ refers to the power dissipated by host $j$ when the proposed placement of that position would hold. However, when a host is expected to be over-utilised as a result of a particle's proposed placement, the fitness score $f_i$ is set to be infinitely large.
The aim is thus to minimise the fitness score as described in Equation 2.8. Each particle keeps track of the best fitness score and the respective position for that score. So when the fitness score for the current position is calculated, it then immediately checks if it is better than its best fitness score. If this is the case, the value for the personal best fitness score and personal best position are updated.
Once every particle has updated its personal best, if needed, the personal best of each particle is compared to the global best. The global best is also stored with its respective position. When the best personal best of all particles is better than the current global best, the fitness score of the global best is replaced by the respective personal best and the global best position is updated by the respective personal best position.
Once the global best values are updated, if needed, each particle determines its new position. This is done by adding the velocity of that particle to its current position. After that, the new velocity for that particle is determined. This is calculated according to the following equation [48]:

$$V_{t+1} = w_t \cdot V_t + c_c \cdot r_c \cdot (X_{best} - X_t) + c_s \cdot r_s \cdot (X_{globalBest} - X_t) \tag{2.9}$$

Here, $V_{t+1}$ refers to the particle's new velocity, $V_t$ refers to its current velocity, $X_t$ refers to its current position, $X_{best}$ refers to its personal best position and $X_{globalBest}$ refers to the global best position.
Besides, $w_t$ refers to an inertia coefficient which determines the influence of the current velocity. This inertia coefficient changes over the number of iterations by the following equation:

$$w_t = w^{max} - (w^{max} - w^{min}) \cdot \frac{t+1}{N_t} \tag{2.10}$$

Here $w^{max}$ refers to the maximum inertia coefficient and $w^{min}$ to the minimum inertia coefficient. Moreover, $N_t$ is the maximum number of iterations that are performed, which is 50 in this case. The variable $t$ refers to the current iteration number. The process is repeated 50 times, which means $t$ ranges from 0 to 49.
Furthermore, the constants $c_1$ and $c_2$ refer to the cognitive and social constant respectively. The cognitive constant ($c_1$) determines how much the particle's velocity is determined by its history and the social constant ($c_2$) determines how much the particle's velocity is influenced by the history of all the other particles. For this research, they both have been set equal to 2.05, which has been chosen based on the research provided by Ibrahim et al. [48]
Moreover, $r_1$ and $r_2$ represent two different random values between 0 and 1, which are uniformly distributed. The velocity and position are both only defined in discrete integer values. However, Equation 2.9 could lead to non-integer values. Thereafter, the result of that equation is always rounded to the nearest integer value, which implicitly ensures that the position always consists of integer values only. The process of the particle swarm optimisation algorithm is captured in Algorithm 2 in Appendix A. Note that this algorithm needs some sort of mapping between integer values and the corresponding VM or host. The index of $gBestX$ corresponds to the VM that has the same index in $\mathcal{V}_{migrate}$ and the value of $gBestX$ corresponds to the index of $\mathcal{H}_{available}$.
The values that are used for the variables that need to be initialised are shown in Table 2.1.

| Symbol | Value | Description |
|---|---|---|
| $N_t$ | 50 | Number of iterations |
| $N_p$ | 30 | Number of particles |
| $w^{max}$ | 0.9 | Maximum value for the inertia coefficient |
| $w^{min}$ | 0.4 | Minimum value for the inertia coefficient |
| $c_c$ | 2.05 | Cognitive constant |
| $c_s$ | 2.05 | Social constant |
| $\mathcal{P}$ | | Set of particles |
| $X^p$ | | Position of particle $p$ |
| $V^p$ | | Velocity of particle $p$ |

**Table 2.1:** Variables for the PSO algorithm

### 2.2.4. Genetic Algorithm

Another meta-heuristic solution is the genetic algorithm (GA), which is inspired by the natural process of evolution. It consists of many genetic operators which are often inspired by biological processes, such as crossover, selection and mutation [50]. The GA can be used for a great variety of optimisation problems too [51, 52].
Similar to the PSO algorithm, the GA starts with a set of candidate solutions. This set is often referred to as a population. The candidate solutions can be referred to as parents. After initialising the population, a repetitive process of genetic operations is performed. Each iteration is referred to as a generation.
The first stage of a generation is selection. This describes the process of selecting two parents that are going to reproduce. A lot of different methods have been researched [53]. One of the most common selection methods is the roulette-wheel selection, also known as fitness proportionate selection [54]. The probability of selecting each parent is determined according to their fitness. When a parent has a high fitness score compared to the other parents, the probability of being selected is also high. The selection probability is commonly defined using the following equation:

$$p_i = \frac{f_i}{\sum_{j=1}^{N_p} f_j} \tag{2.11}$$

Here $p_i$ is the probability of the selection of parent $i$ and $f_i$ is its fitness score. This is calculated for a population of size $N_p$. Other selection processes are not elaborated upon in this work, since there is no implementation of the GA in the context of the VM placement problem that addresses a different selection process.
The second stage of a generation is called a crossover. Crossover consists of cloning a part of the solution from its first parent and cloning the other part of its solution from the other parent. Just like for the selection process, many different crossover methods have been researched [55]. One of the most common crossover methods is the one-point crossover. This method picks a random crossover point. This process is visualised in the Figure 2.3.

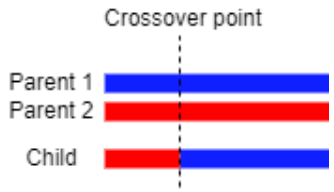


**Figure 2.3:** Visualisation of crossover process

This process randomly selects a point in between the dimensions of the parents. This point indicates that the solution mapping of parent 1 before this crossover point is taken, and the solution mapping of parent 2 after this crossover point is taken to construct the solution mapping for the child.
After the crossover stage, a child can have some changes to its candidate solution as a result of mutation. The mutation stage was the first process that later evolved to the GA [56]. The mutation stage

alters the solution that results from the crossover stage according to a probability distribution. Just like the other stages, many different alternatives have been proposed in the literature [57]. They mostly vary in the distribution of picking a random number, which impacts the probability for each of the mappings provided by the candidate solution to be altered to any other possible mapping. One of the commonly used distributions is a uniform distribution with a very small probability of mutation, such as 1% or even smaller.
Moreover, a process that is often combined with the GA is the process of survival of the fittest, which originates from evolutionary algorithms [58]. Genetic algorithms are a subclass of evolutionary algorithms. The survival of the fittest stage ensures that the fittest parents of the population survive and remain in the next generation, whilst all the others are removed from the population.
There are a lot of hyper-parameters to be set and each should be chosen carefully for the problem it is trying to solve. The GA has been used for bin-packing problems which thus is easily extended to the virtual machine placement problem [59].
Since then, extensive research has been conducted on fine-tuning hyper-parameters [60, 61, 62]. The foremost challenge to be tackled by designing a GA is keeping the run-time as short as possible. Even though it is better than for implementations of ILP algorithm, GA too has a very long run-time. This, for instance, resulted in propositions of having more efficient fitness functions to evaluate the candidate solutions [63].

### Implementation of GA

The genetic algorithm (GA) has been implemented, which is, just like PSO algorithm, a meta-heuristic algorithm. The genetic algorithm shows some similarities with the PSO algorithm.
Just like for the PSO algorithm, a set of random solutions is created with the same dimensions. The same mapping, as explained for the PSO algorithm, is used for the genetic algorithm. The set of solutions in this case is called the population. Each solution of the population only consists of a mapping.
A solution is called a parent. From the population, two parents are selected randomly. The parents with a better fitness value (thus a smaller value), have a higher probability of being selected. The same fitness function as for the PSO algorithm is used for the GA (see Equation 2.8). The probability of a parent to be selected is determined as follows:

$$p_i = \frac{\frac{1}{f_i}}{\sum_{j \in \mathcal{P}} \frac{1}{f_j}}, \qquad \text{with } \mathcal{P} \text{ as the set of parents} \tag{2.12}$$

Once two parents are selected, a new solution is generated as a result of these parents. This new solution is called the child and the process of generating a child is called crossover. The crossover consists of taking the mapping of the first $c$ VMs of one parent and the mapping of the other VMs from the other parent. This value $c$ is a random integer number from a uniform distribution between 0 and $|\mathcal{V}_{migrate}|$.
When the child has been generated, it might undergo some mutations. Each mapping of the solution of the child has a 1% chance to mutate, which means that the mapping of that VM changes to a different host. The processes of parent selection, crossover and mutation within the context of this research are visualised in Figure 2.4.

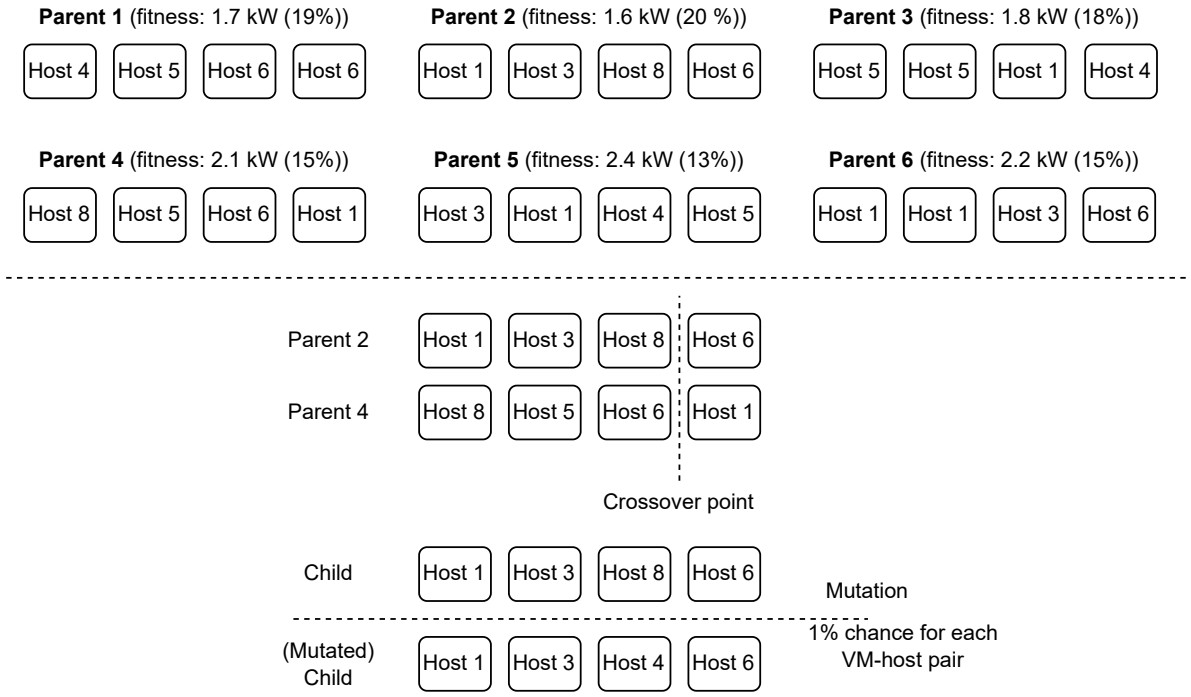


**Figure 2.4:** Visualisation of the parent selection, crossover and mutation process of the genetic algorithm

For each parent in Figure 2.4, its fitness score and its chance of selection are shown. The fitness score is calculated the same as for the PSO algorithm, which is described in Equation 2.8. The calculation for the chance of selection is described in Equation 2.12. Note that the parents with the best fitness are not automatically selected. They do have the highest probability of being selected. Figure 2.4 shows an example where not the two parents with the best fitness are selected.
Like the PSO algorithm, the genetic algorithm consists of multiple iterations, which are called generations. The number of children that are generated differs per generation. This is because the population size should remain the same, but the number of parents that survive a generation differs. A survival of the fittest principle has been applied to this algorithm, which means the fittest portion of parents survives. The number of parents that survive does increase over time. This ensures that in the earlier stages of the algorithm, it is focused on exploring the solution space and in the later stages it is focused on converging to an (local) optimum. The following equation is used to determine the fraction of parents that survive:

$$w_t = w^{max} - (w^{max} - w^{min}) \cdot \frac{1}{t+1} \tag{2.13}$$

Note that there is a small difference between Equation 2.10 and Equation 2.13. In Equation 2.10, $w_t$ changes over the iterations from $w^{max}$ to $w^{min}$, whereas in Equation 2.13 $w_t$ changes from $w^{min}$ to $w^{max}$. This differs for these algorithms to ensure that both algorithms focus on exploring the search space in the early iterations and converge to an optimum in the later iterations.
In the end, the solution with the best fitness score of the last population is selected as the final solution for the mapping. The functioning of the genetic algorithm is captured in Algorithm 3 in Appendix A. For this algorithm, the mapping is done similarly to the mapping for the PSO algorithm. The values that are used for the variables that need to be initialised are shown in Table 2.2.

| Symbol | Value | Description |
|---|---|---|
| $N_t$ | 50 | Number of generations |
| $N_p$ | 30 | Number of parents |
| $w^{max}$ | 0.9 | Maximum value of survival coefficient |
| $w^{min}$ | 0.4 | Minimum value of survival coefficient |
| $\mathcal{P}$ | | Population, set of parents |

**Table 2.2:** Variables for the genetic algorithm

## 2.3. Integrating eco-mode into virtual machine placement

Virtual machine placement algorithms mostly do not consider eco-mode techniques that servers are capable of. Of course, in many of the algorithms, the power dissipated by each of the servers is considered, but it does not account for the possible switching between P-states.

A few DFVS-aware algorithms have been proposed in the literature. Wang and Wang [64] proposed a DFVS-aware algorithm, but they have not considered the impact of the frequency. Effectively, this means that the proposed VM placement algorithm and the eco-mode of servers operate in parallel.

Petrucci et al. [65] proposed a frequency-aware VM placement algorithm for small data centers. They did not cover any impact on the performance and only focused on the conservation of energy. Arroba et al. [66] also proposed a frequency-aware model which also presented an improved (very slightly) improved performance. However, they lack in their elaboration on the actual VM placement policy.

There is not yet research that delves into the decomposition of the influence of different key parameters in data center environments. The problem of getting a data center more energy efficient consists of very many parameters, which makes it extremely hard to tune a data center, or even a cluster, to its most power-efficient form [67]. This research provides a framework for a subset of these parameters and explicates the relationships between these parameters.

# 3

# Methodology

This chapter outlines the methodology employed in this research. Firstly, an explanation of the functioning of the simulation tool, CloudSim [68] is provided, as it forms the foundation of this study. Subsequently, Section 3.2 elaborates upon the parameters that are varied over in the experimental setups. Finally, Section 3.3 provides details on the experimental setup itself.

## 3.1. CloudSim

CloudSim is a simulation tool that has been used very widely within this research area. CloudSim is a toolkit for modelling and simulating the behaviour of cloud environments, such as VMs, data centers and resource provisioning services, which includes the VM placement algorithm [69]. CloudSim is an event-driven simulation tool, which means that all components maintain a received message queue and all components generate messages to pass along to other components. The components that form the core of CloudSim are:

- *Cloud Information Service Registry*: A registry that maintains data center characteristics information.
- *Broker*: The broker resembles the user query, which consists of VMs and acts as a scheduler of this query among the data centers.
- *Data Centers*: The data center entity resembles a data center, including a virtual machine manager (VMM).

### 3.1.1. CloudSim Core Components

For the scope of this research, only the *data center* entity is of relevance, since only one data center is simulated. The other two components only have a significant role when more than one data center is simulated. The *data center* entity consists of multiple components.

**Virtual Machine Manager (VMM)** This describes the VM managing policies, which mainly consists of the VM placement algorithm. This is the most important component of the VMM for this research. Furthermore, it consists of the host over-utilisation detection algorithm and the VM selection policy. The exact functionality of each component of the VMM is explained later.

**Host List** This is a list of all hosts, ergo servers, that are placed within the data center. Each host contains two main components: technical specifications and its own VM list.
The technical specifications consist of multiple common server characteristics, such as computational power, remote access memory (RAM), network bandwidth and storage. However, for this research, only the computational power is of relevance. Furthermore, the energy consumption for all utilisation levels of the computational power is described for each host by its power curve, as is explained in more detail in Section 3.2.1. Within this characteristic of the host, the difference between eco-mode and high-performance mode is defined. In this research, the delays as a result of switching between

C-states [70] and P-states [9] are neglected, since these are negligible for the performance of a data center as a whole.
The VM list of the host only contains the VMs that are provided resources by that host. The host only knows the current computational power request of the VMs, and it does not know the dimensions of the VMs. More details about the dimension of the VMs are given in Section 3.2.3.

**VM List** This list contains all VMs that should be processed within this datacenter. For this research, this means all VMs. Each VM consists of similar characteristics as hosts do. However, it describes the maximum request for each resource. Thus, each VM is described by its maximum computational power, maximum memory (RAM), maximum network bandwidth and maximum storage. Furthermore, each VM keeps track of its current utilisation for each resource. Again, this research only keeps track of the computational power. It is important to note that this VM list has insight both into the dimensions of the VM and the current request of computational power. Note that a VM is defined by a maximum computational request (and so for all other resources), but the computational request during simulation varies over time. The configuration thus defines the maximum computational request by a VM, which is referred to as the dimensions of a VM.

The information known to each entity is like a tree. Thus the virtual machine manager has insight into all information, both of the host list and the VM list. The individual hosts only have insight into its technical specifications and the current utilisation of its VMs. Each individual VM has insight into its dimensions and current utilisation but has no insight into the technical specifications of the host it is placed on or of any of the other virtual machines. This is visualised in Figure 3.1.

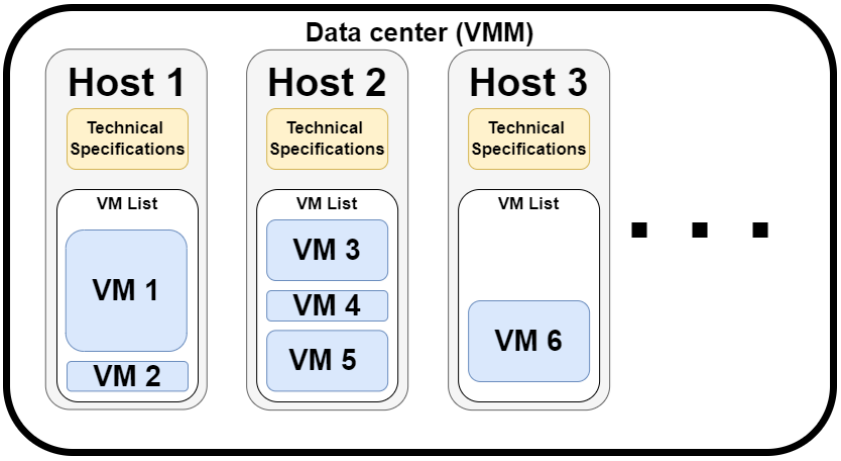


**Figure 3.1:** Visualisation of the *data center* entity

As mentioned before, the VM knows its current utilisation. This is based on the tasks of the applications that are running on this VM. Since the purpose of this research is to investigate the dynamic between VMs and the VMM in combination with the eco-mode of hosts, each VM has one application running with a task that is produced stochastically according to a uniform distribution. This effectively means that the utilisation level of each VM is at random according to a uniform distribution between zero and the maximum request of that VM. These applications are referred to in CloudSim as *cloudlets*. However, these are not of importance in this research. In this research, workload thus means the VMs and not the actual set of instructions that should run on these VMs.

### 3.1.2. Data Center Configuration

A simulation first needs to be set up before it can start. This is done by the following steps.

1. A *Cloud Information Service* registry is created;
2. A *broker* is created
3. A *data center* with all its hosts is created;
4. The *broker* submits all VMs to the, in this use case only one, *data center*;
5. Now the simulation can start.

The important part of these steps is to a have correct host list for the *data center* (step 3) and a correct VM list for the simulation (step 4).
For this experiment, a homogeneous data center is created, which means that all the hosts have the same technical specifications. In this way, the impact of the eco-mode on a cluster level can be extracted in a consistent environment. For further research, it would be very interesting to investigate the impact within a heterogeneous data center, since this is a much more realistic scenario [71].
For this research, the homogenous data center consists of 50 Dell PowerEdge R630 servers with an Intel Xeon E5-2699 v4 2.2 GHz processor [72]. This server with this processor has been chosen since research has been done on the eco-mode provided by the Intel processor on this server for one server [16]. Of this research, this server has been chosen since it is most common in commercial use and is not a server designed for high performance.
The Intel Xeon E5-2699 v4 2.2 GHz processor can functionally have 2 CPUs [72], which has been simulated as 2 cores in CloudSim. This differs from the number of cores as specified in the sheet Intel provides, but in practice, this would be the same. This is because the way cores are simulated in CloudSim is similar to the way the number of functional CPUs Intel refers to in its datasheet. Therefore, the computational power capacity is 2.2 GHz per core, so a total capacity of 4.4 GHz this host has. The computational power in CloudSim is specified in millions instructions per second (MIPS), which is not known. However, the 2.2 GHz is converted as 2200 MIPS assuming the processor to have one instruction per clock cycle. This assumption holds since the same conversion is used for the virtual machines. The dimensions of these virtual machines are discussed in Section 3.2.3, since this is one of the aspects that is analysed in this research.
Not only does the host list need to be set up, but also the configuration for the functioning of the virtual machine manager needs to be set up. This consists of the components as described in Section 3.1.1. The VM placement algorithms are described in Section 3.2.2. Moreover, the host over-utilisation detection algorithm and the VM selection algorithm, which decides which VMs migrate from an over-utilised host, are configured along default configurations of CloudSim. The effect on and from these algorithms is not analysed within the scope of this research and thus is the same for all simulations.
Since the over-utilised detection algorithm is not analysed within this research, one of the default algorithms provided by CloudSim has been chosen. Therefore, the host over-utilisation detection algorithm is based on a linear regression algorithm that predicts the host utilisation based on the last 10 utilisation samples. This prediction is multiplied by a safety parameter. When the result of this multiplication is larger than 100%, the host is marked as over-utilised. The safety parameter used in this research is 1.2, which effectively means that the predicted utilisation level can not be higher than 83%.
For the same reason as is used for the choice of an over-utilised detection algorithm, the choice for the VM selection algorithm has been chosen. This is for the VM with the smallest migration time, which effectively means the VM with the smallest memory. This is based on the current utilisation level of the VMs, not on the dimensions of the VMs. However, the memory specification of VMs is not used within the scope of this research. Therefore, it is assumed that VMs with a larger request of computational power, have a larger memory, which ensures that the VM with the largest request in computational power is selected to migrate to a different host. This assumption holds for many VM instances, for example when using Amazon EC2 instances [73]. This too, is a default configuration of the CloudSim tooling.

### 3.1.3. Simulation Flow

Once the data center is configured as described in Section 3.1.2, the simulation can start. Firstly, all VMs are submitted to the *data center* by the *broker*. Since there is only one *data center* entity for this research, they are all submitted to that *data center*. When this is done, the *data center* provides a first placement. There is no optimisation process here, which means that for the first time frame, a different VM placement policy is used. This is done using a first fit algorithm based on the maximum computational power of a VM. This means that the initial placement is the same for each simulation run. After that, on an interval basis, the placement is optimised. This is done by the following steps:

1. Find all over-utilised hosts;
2. Select VMs that need to be migrated from these hosts;
3. Find new placement for VMs that need to be migrated;
4. Find under-utilised hosts after this placement;
5. Find placement for VMs of under-utilised hosts, if possible.

This is done once every five minutes for this simulation. Steps 4 and 5 enforce that all VMs of hosts that have a very low utilisation will migrate to hosts that can host these VMs besides the VMs they already host. This process is necessary to be able to shut down hosts, which is done to conserve energy. The utilisation threshold is 1% of the host's capacity, which is a CloudSim default value. This value might seem extremely low, but in practice is often surpassed. One percent of the capacity of the host in this case means 440 MHz (or MIPS). For example, when only one small VM with a maximum computational request of 500 MHz is placed on this host, the chance of being below this threshold is 88%.
The utilisation level of each VM is not updated during the timeframe of five minutes. At the end of the timeframe, the utilisation level of each VM is updated. The utilisation level (before it is updated) can be regarded as the average utilisation level for that VM for the last five minutes. Then the utilisation levels of all the VMs are updated. After this update, the steps described above are performed. In this way, the algorithms operate with the current utilisation levels.
Once the new VM placement is determined in step 3 (and possibly steps 4 and 5), CloudSim tries to migrate the VMs accordingly. When migrating a VM from one host to another, it checks if it is possible. This is done by checking the following criteria when placing a VM on a host:

$$\sum_{i \in \mathcal{V}} x_{ij} c_i \leq C_j, \qquad j \in \mathcal{H}; \tag{3.1a}$$

$$\sum_{i \in \mathcal{V}} x_{ij} m_i \leq M_j, \qquad j \in \mathcal{H}; \tag{3.1b}$$

$$\sum_{i \in \mathcal{V}} x_{ij} b_i \leq B_j, \qquad j \in \mathcal{H}; \tag{3.1c}$$

$$\sum_{i \in \mathcal{V}} x_{ij} s_i \leq S_j, \qquad j \in \mathcal{H}. \tag{3.1d}$$

Here, $x_{ij}$ is one and only one if VM $i$ is placed on host $j$, $c_i$ is the current computational request of VM $i$, $m_i$ is its current memory request, $b_i$ its current network bandwidth request and $s_i$ its storage request. Similarly, $C_j$ is the computational power capacity of host $j$, $M_j$ its memory capacity, $B_j$ its network bandwidth capacity and $S_j$ its storage capacity. When one of these conditions is not met, the new VM can not and will not be placed on that host. When no suitable host was found for that VM, it will remain on its current host, even if this would lead to an over-utilised host and thus a performance degradation. Note that this set of equations forms the basis for the ILP algorithm as described in Section 2.2.2. Figure 3.2 summarises the workflow of a CloudSim simulation.

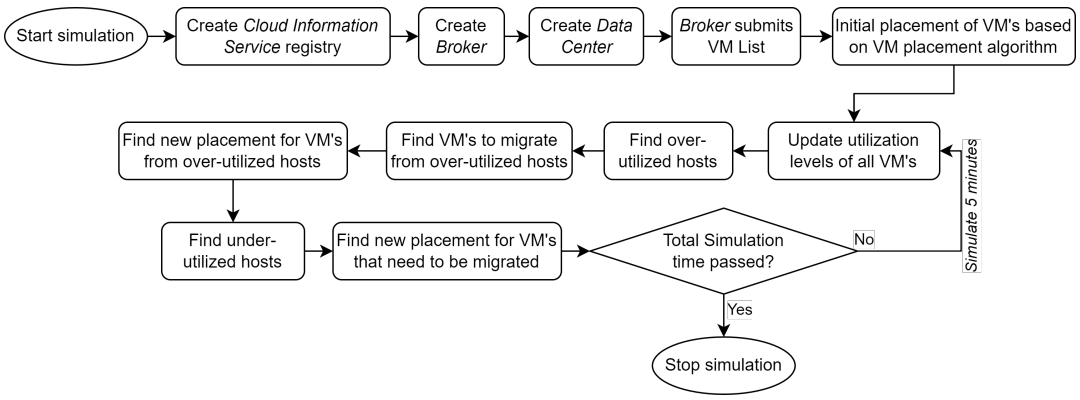


**Figure 3.2:** CloudSim simulation flow

The steps described above are repeated every five minutes until the total simulation time has passed, which is a simulation time of 1 day exactly. This is chosen since it is the default value of the CloudSim tooling. Since this research does not look at workload patterns or workload prediction, this simulation time suffices for this research. This means that the steps described above are repeated 288 times per simulation. The workload that will be used for simulation is randomly generated but seeded. Thus it is the same for each simulation (that has the same VM type distribution). This is chosen to obtain a fair

comparison of the different power dissipation curves and the different VM placement algorithms.
To illustrate the confidence of using a seeded workload, the first experiment also has been conducted 100 times. Each run randomly generates a workload. This results in confidence in the calculated averages and a spread of each of the results. These are shown in Appendix B.

## 3.2. Analysed Parameters

To give insights on the impact of the eco-mode on cluster level within data centers, three main parameters that could influence its impact are analysed. These are the shape of the eco-mode power curve (Section 3.2.1), the VM placement algorithms (Section 3.2.2) and the workload dimensions of the virtual machines (Section 3.2.3).

### 3.2.1. Shape of Eco-Mode Power Dissipation Curves

The shape of the eco-mode power dissipation curve is dependent on the server hardware and mostly the CPU. As explained in Section 2.1, processors can switch between so-called power-performance states (P-states) and CPU power states (C-states).
The C-states switch between different levels of *sleep modi* of the cores. These states can conserve the energy of servers when their workload is relatively low. These benefits will thus be largest when a server is mostly idle. For a more detailed description of this, read Section 2.1.1
The P-states throttle the computational power to conserve energy. They switch between voltage-frequency pairs to conserve energy. The processors can lower the voltage, to decrease the power dissipated by the server. However, the maximum operating frequency should also be lowered, which means a smaller computational power.
Furthermore, the eco-mode power dissipation curve can be different when a different set of instructions is performed on the processor, as illustrated by Rteil et al. [16]. This research does not analyse the relationship between the type of instructions that need to be performed on a processor and the shape of the eco-mode power curve.
However, to include this aspect within the impact of the eco-mode of servers on a cluster level, different eco-mode power dissipation curves have been implemented. As a reference, the eco-mode power curve shape of the Dell R630 PowerEdge server with an Intel Xeon E5-2669 v4 processor has been used. The reference curve shape is taken from the experiments performed by Rteil et al. [16] with a workload fully based on stochastical simulation in Java (ssj) operations. This type of instruction set has been chosen since it is the most common and general use case for many commercial applications. Other instruction sets have often a very specific use case.
This reference eco-mode power curve shape has been assumed to be the most optimistic power curve shape. This power curve shape has been parameterised. This is also done for the high-performance power curve from the same experiment.
Moreover, 8 other power curves have been parameterised. They differ in the power dissipated when there is no workload, which can be referred to as the impact of the C-states, and the shape of the curve towards the maximum power dissipated, exponential, linear or root-like. This can be referred to as the impact of the P-states.
The eco-mode power dissipation curves have been parameterised by the following simple equation:

$$P(u) = c + \gamma u^p \tag{3.2}$$

with

$$\gamma = (380 - c)100^{-p}$$

Here $c$ represents the power dissipated when there is no workload (in watts), thus the impact made by the C-states, and $p$ represents the curve of the power profile, thus the impact made by the P-states. Furthermore, $u$ represents the utilisation level in percentages, which explains the value 100 in the equation for $\gamma$. The values that were used for $c$ and $p$ are listed in the table below.
Moreover, the high-performance power curve has been parameterised similarly, which is a linear line starting from 220 watts and going linearly to 380 watts at a 100% workload.
These power curves are shown in Figure 3.3. The reference curve is visualised by Figure 3.3g. This reference curve has been assumed to be the most optimal curve due to the eco-mode of a server, which means this power dissipation curve has the lowest value for any utilisation level.

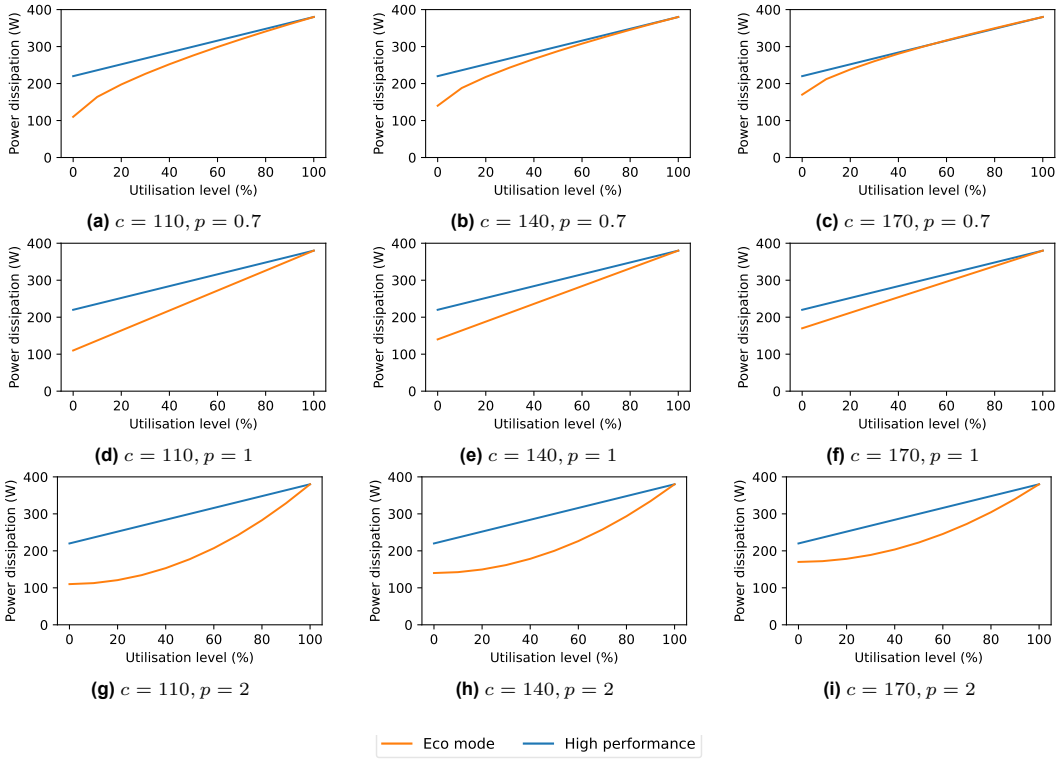


**Figure 3.3:** Parameterised power curve shapes

| $c$ (Watt) | $p$ |
|---|---|
| 110 | 0.7 |
| 140 | 1 |
| 170 | 2 |

**Table 3.1:** Values of $c$ and $p$

### 3.2.2. Virtual Machine Placement Algorithms

For the scope of this research, four different VM placement algorithms have been implemented which all have the objective to minimise the energy consumption of the data center. To have a fair comparison, all algorithms are single objective and thus only try to minimise energy consumption and do not use any technique to preserve the performance of the data center. In this way, the impact of eco-mode is the dominant aspect to be analysed and not the impact of the different VM placement algorithms. However, it is analysed for multiple VM placement algorithms to analyse if there is a dependency between the impact of the eco-mode and the VM placement algorithms.
The algorithms that have been implemented are described in Section 2.2. The implementations of these VM placement algorithms have been created as separate Java files in the CloudSim tooling and have been integrated within the flow as shown in Figure 3.2.

### 3.2.3. Virtual Machine Dimensions

The last parameter that is varied over in some of the experiments is the VM dimensions. With the VM dimensions, the maximum request for each of the resources is meant. Since the scope of this research is only focused on the computational power resource, this is the only resource that varies.
The requested computational power of a VM is noted in Megahertz (MHz), which is also the common practice for the capacity of the hosts. However, since CloudSim only operates in millions instructions per second (MIPS), the simple one-on-one conversion for both the capacity of hosts and the requests of VMs is made from MHz to MIPS.
For this research, three different VMs are used: 500 MHz, 1000 MHz and 2000 MHz. In CloudSim, these would have the values: 500 MIPS, 1000 MIPS and 2000 MIPS.

There are four different distributions of VMs that are processed. For each different type, there is a distribution that only consists of that type of VM and a fourth distribution that consists of a uniform distribution of all VM types.
To have a fair comparison, the expected workload of each of these distributions is matched. The utilisation levels during each simulated time frame are the same for each of these VMs and they all have a uniform distribution. This means that for the distribution that only consists of VMs that request 500 MIPS more VMs are processed than for the distribution that only consists of VMs that request 2000 MIPS. The actual numbers that are used are shown in Table 3.2.

| VM size(s) | #VMs |
|---|---|
| 500 MHz | 112 |
| 1000 MHz | 56 |
| 2000 MHz | 28 |
| 500 MHz, 1000 MHz, 2000 MHz | 16, 16, 16 |

**Table 3.2:** Distribution of number of virtual machines

They all lead to an expected workload of 28000 MIPS at any point in time of the simulation, since the expected request MIPS is half of its maximum request due to a uniform distribution. All of these VMs request one virtual CPU, thus each VM can be fully placed on one physical CPU of a host.
The hosts, each with a capacity of 2200 MIPS and 2 CPUs have a combined capacity of 50 x 2200 x 2 = 220000 MIPS. The capacity thus more than suffices, since the expected workload only accounts for 12.7% of the capacity. The maximum workload for each distribution would account for 25.4%. This illustrates that the total capacity limit is not encountered in this research. In this way, the focus is really on the functioning of the VM placement algorithms and the impact of the eco-mode when it can fully operate.

## 3.3. Experimental Setup

To test the impact of eco-mode and answer the research questions mentioned in Section 1.5, three experimental setups have been created. These are elaborated upon in this section. Moreover, Table 3.3 shows the variables that have been kept constant in each of the experiments, but are needed for the setup in CloudSim. This is mainly the configuration for the data center consisting of the hosts.

| | |
|---|---|
| number of hosts | 50 |
| MIPS capacity per core | 2200 |
| number of cores per host | 2 |

**Table 3.3:** Environment variables for setup in CloudSim

### 3.3.1. Experiment 1: Power curve shapes versus VM placement algorithms

In the first experimental setup, the impact of the eco-mode as a result of different power dissipation curves is compared for the different implemented VM placement algorithms. The workload that should be processed in all of these simulations consists of the uniformly distributed VM types.
For each different power dissipation curve, as can be seen in Figure 3.3, a simulation run is done. This is also done for the linear high-performance power dissipation curve. This would result in 10 different results for each VM placement algorithm.
These runs are performed for each of the different implemented VM placement algorithms, resulting in 4 different results per power curve. This would result in a total of 40 different results, where each result is linked to one combination of a power curve and a VM placement algorithm.
There are two different kinds of results measured. Firstly, the total energy consumption for the simulation time (of 1 day) is measured for each simulation run. This is calculated as follows:

$$E = \sum_{j \in \mathcal{H}} \sum_{t}^{T_{sim}} P^j(u) \cdot T \cdot \delta(t \mod T) \tag{3.3}$$

Here, $E$ is the energy in kWh, $T_{sim}$ is the total simulation time (24 hours), $P_t^j(u)$ is the power dissipated by host $j$ at time $t$ for the utilisation $u$ at that time and $T$ is the time frame of the simulation (5 minutes). The $\delta$ function forces that the energy consumed during the last time frame is calculated at the end of the time frame.
Since the simulation environment does not change during a time frame, this method suffices to calculate the total energy consumed during the simulation. Moreover, when the utilisation $u$ of a host is equal to zero, the power dissipated is assumed to be zero, since the host can be shut down.
Moreover, the research the impact mode on the performance, the percentage of missed instructions is calculated as follows:

$$\begin{aligned} SLAV &= \frac{\sum_{i \in \mathcal{V}} \sum_t^{T_{sim}} c_{t,r}^i \cdot T \cdot \delta(t \mod T) - \sum_{i \in \mathcal{V}} \sum_t^{T_{sim}} c_{t,a}^i \cdot T \cdot \delta(t \mod T)}{\sum_{i \in \mathcal{V}} \sum_t^{T_{sim}} c_{t,r}^i \cdot T \cdot \delta(t \mod T)} \\ &= 1 - \frac{\sum_{i \in \mathcal{V}} \sum_t^{T_{sim}} c_{t,a}^i \cdot T \cdot \delta(t \mod T)}{\sum_{i \in \mathcal{V}} \sum_t^{T_{sim}} c_{t,r}^i \cdot T \cdot \delta(t \mod T)} \end{aligned} \tag{3.4}$$

Here $c_{t,a}^i$ is the allocated computational power in MIPS for VM $i$ at time $t$, whereas $c_{t,r}^i$ refers to the requested computational power.
This does calculate the percentage of instructions that are missed due to an under-allocation by the host, which is caused by hosts that are over-utilised since then its capacity can not fulfil the needs of all the VMs it hosts.
This performance metric is noted as the percentage of missed instructions. However, this does not mean that actual instructions are missed when these scenarios happen. In practice, this percentage of missed instructions is experienced as a delay. The percentage of missed instructions is namely a result of an under-allocation and thus the computational power that is requested by the VMs is not provided by the host's capacity.
CloudSim can provide more performance metrics, which are all focused on the allocation of computational power. All of these performance metrics are some form of an average of the whole simulation, thus some effects are not able to be obtained from this simulation tool. This metric is used since it covers most accurately the impact on the server cluster's performance as a whole compared to the other performance metrics.
With the metrics as described in Equations 3.3 and 3.4, an analysis can be made of the impact of the eco-mode as a result of different power curve shapes for that eco-mode. Moreover, it can be concluded if there is a dependency of the impact of the eco-mode based on the VM placement algorithm.

### 3.3.2. Experiment 2: Power curve shapes versus VM dimensions

In the second experimental setup, the power dissipation curves are analysed again, but this time compared for the different VM dimensions. In this setup, the virtual machine (VM) placement algorithm that is used is the same for each simulation run. This is the BFD algorithm. The explanation for this is given in Section 4.2.
Thus a simulation, again, is run for each of the power dissipation curves as shown in Figure 3.3, including the high-performance power dissipation curve. For each of these curves, 4 simulation runs are done for each of the different workload distributions, as described in Table 3.2. This results in a total of 40 different results for this experimental setup.
For the results, the same metrics as in experiment 2 are used, which are described by Equations 3.3 and 3.4. With these metrics, an analysis can be made of whether the impact of the eco-mode is influenced by the VM dimensions (regarding computational power).

### 3.3.3. Experiment 3: VM placement algorithms versus VM dimensions

In the third and last experimental setup, the VM placement algorithms are compared for the different VM dimensions. This is tested for one eco-mode power dissipation curve and the high-performance power dissipation curve. The eco-mode power dissipation curve that is used is explained in Section 4.3.
A simulation is run for each of the VM placement algorithms for each of the workload distributions as described in Table 3.2. This would lead to a total of 32 results since the simulation is run for one eco-mode and the high-performance power dissipation curve.
Again, the same metrics, that are described by Equations 3.3 and 3.4 are used. With these metrics, it can be analysed whether the design choices that can be made are dependent on the VM placement algorithm that is used (and vice versa).

# 4

# Results

This chapter reveals the findings from the experiments discussed in Section 3.3. This chapter breaks down the results of each experiment. Lastly, an overall conclusion is drawn based on these findings.

## 4.1. Exp. 1: Power curve shapes versus VM placement algorithms

In this first experimental setup, the different power dissipation curves, as described in Section 3.2.1, are compared against the different implemented VM placement algorithms: the best fit decreasing (BFD) algorithm, the integer linear programming (ILP) algorithm, the particle swarm optimisation (PSO) algorithm and the genetic algorithm (GA). The VMs that are processed in the simulations for this experimental setup are all constant. The uniformly distributed VM types are used, as described in Section 3.2.3.
Each of the aforementioned algorithms has been applied for each of the applied power dissipation curves. The simulations utilising either the particle swarm optimisation (PSO) algorithm or genetic algorithm (GA) have been executed 10 times each. Then the average of these results is taken for all metrics. This repetition is essential to average out the variation in the produced results due to the random features in their algorithms.
It is sufficient to only repeat the simulations for the PSO algorithm and the GA since the BFD algorithm and the ILP algorithm will reproduce the same results always for their simulations. This is because the random generator for the utilisation levels of the VMs is seeded and thus will always be the same for each simulation. The reason to seed the random generator is to compare the different simulations for the same workload which shifts the focus of the analysis towards the VM placement algorithms and the power dissipation curves.
To compare the different simulation results, two metrics are used: energy consumption and percentage of missed instructions. The energy consumption is the total energy consumption of the server cluster after simulating one day. The percentage of missed instructions is also based on simulating one day. The calculations for these metrics are shown in Equations 3.3 and 3.4. The reason to use these metrics is described in Section 3.3.
Figure 4.1 shows the result for the case where each host has a power curve with $c = 110$ and $p = 2$ (as is depicted in Figure 3.3g) for the energy consumption metric. Each colour thus represents a different VM placement algorithm (BFD algorithms is blue, ILP algorithm is orange, PSO algorithm is green and GA is red). The hosts all have the same power dissipation curve with the aforementioned parameters. Additionally, the high-performance reference for each of the different VM placement algorithms is shown with the box surrounded by black lines. These reference boxes differ since they are dependent on the VM placement strategy.
Figure 4.1 shows that the ILP algorithm has the highest energy consumption both when the hosts are configured with eco-mode and when the hosts are configured with high-performance mode. The PSO algorithm has in both cases the lowest energy consumption.

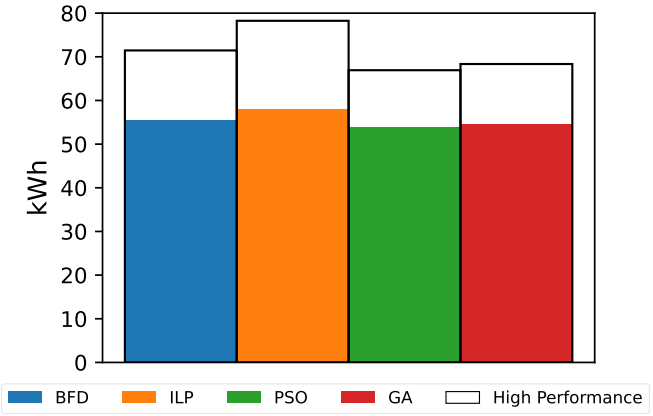


**Figure 4.1:** Energy consumption of a server cluster (with $c = 110$ and $p = 2$) for different VM placement algorithms

Figure 4.2 shows the results for all power dissipation curves for this experiment regarding the energy consumption metric. Each of these bar graphs is structured the same as for Figure 4.1, but each bar graph represents a different power dissipation curve. The parameters that characterise these power dissipation curves are shown in the captions.

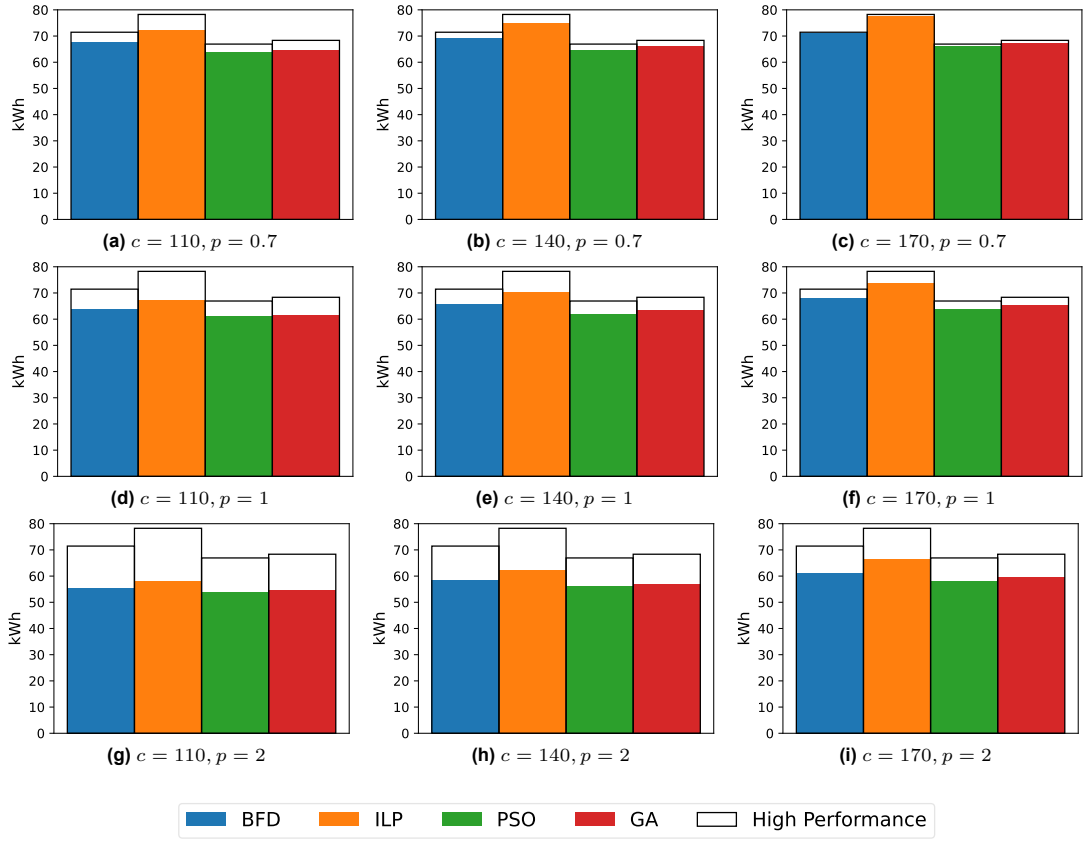


**Figure 4.2:** Energy consumption for different power curves and different VM placement algorithms

Figure 4.2 shows that the PSO algorithm (green bars) always has the lowest energy consumption compared to the algorithms with hosts that have the same power dissipation curve. The ILP algorithm (orange bars) always has the largest energy consumption.

Figure 4.3 shows the percentage of missed instructions when all hosts have enabled eco-mode characterised by $c = 110$ and $p = 2$ for the different VM placement algorithms. The same colour scheme as is used for Figures 4.1 and 4.2 is used. Again, a white box with black edges represents the simulation with the same VM placement algorithm, but with all hosts in high-performance mode.

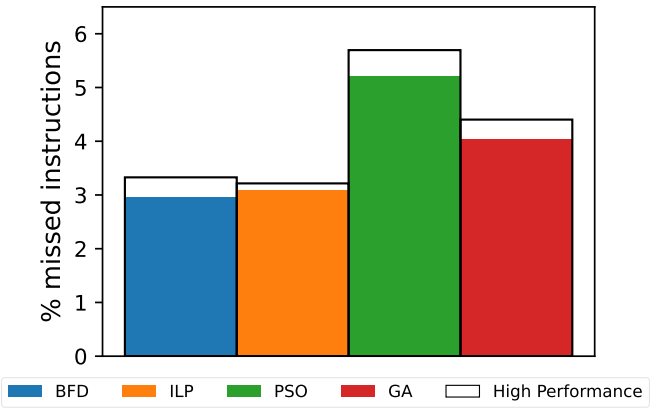


**Figure 4.3:** Percentage of missed instructions of a server cluster (with $c = 110$ and $p = 2$) for different VM placement algorithms

When looking at the height of the bars, in both cases of the eco-mode and high-performance mode, the PSO algorithm (green bar) has the highest percentage of missed instructions and thus performs the worst. Moreover, the ILP algorithm (orange bar) performs the best. The BFD algorithm (blue bar) performs almost as well as the ILP algorithm.
When the energy consumption is lower, it can be expected that the percentage of missed instructions is larger since the same workload is probably placed on fewer hosts. Thereafter, the probability of under-allocation increases. However, whereas the energy consumption metric is relatively close to each other, the variance within the performance metric for the different simulation results varies much more.

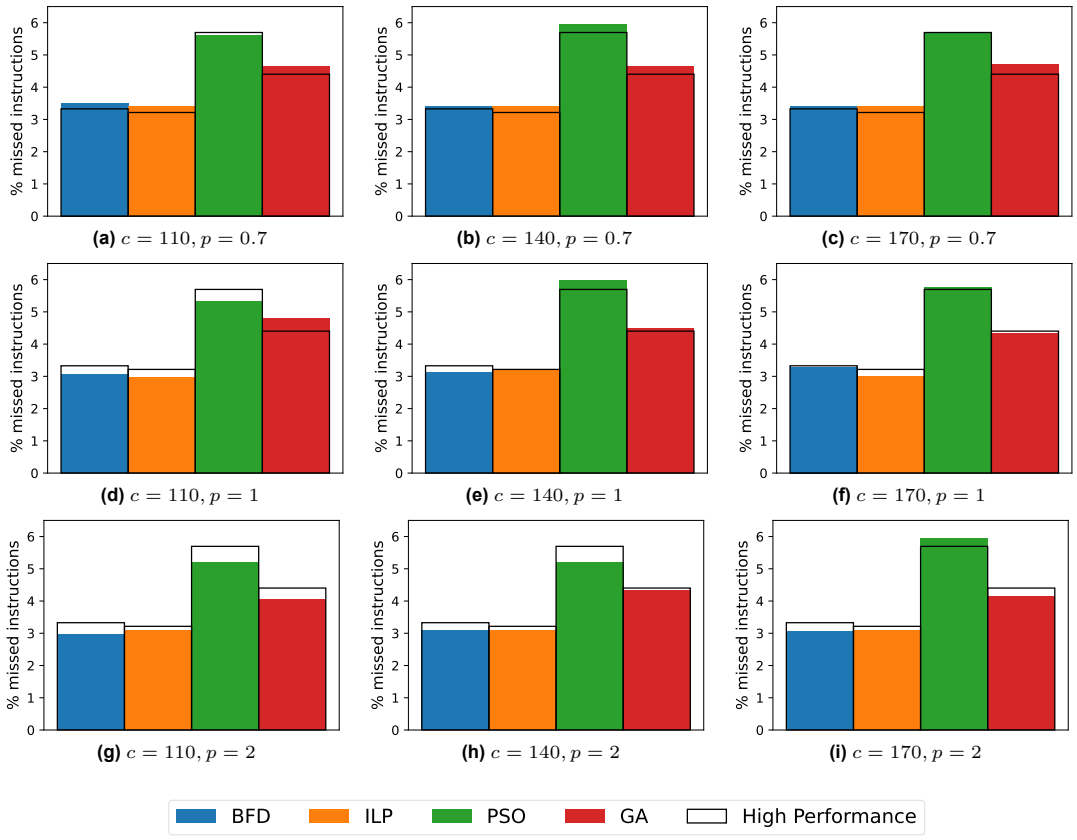


**Figure 4.4:** Performance for different power curves and different VM placement algorithms

Figure 4.4 shows similar graphs as for Figure 4.3, but for all the other power dissipation curves too. The parameters that describe these curves are shown in the caption of each of the graphs.
The PSO algorithm has the highest percentage of missed instructions for all of the power dissipation curves. The GA also performs significantly worse than the BFD algorithm and the ILP algorithm. The BFD algorithm and ILP algorithm have a comparable performance.
Furthermore, the effect of different power dissipation curves due to eco-mode seems not to correlate with any of the VM placement policies. The percentage of missed instructions when using the BFD algorithm decreases in six of the nine different eco-mode power dissipation curves. For all the other VM placement algorithms, the percentage of missed instructions decreases for only four of the nine eco-mode power dissipation curves. Thus there seems to be no clear correlation at first glance regarding the performance.

To gain a full insight into the impact resulting from different power dissipation curves and different VM placement algorithms, both metrics should be considered. This is visualised in Figure 4.5. Each graph represents the impact of a different power dissipation curve for each of the implemented VM placement algorithms. The vertical axis shows the values for the performance metric (percentage of missed instructions) and the horizontal axis shows the energy consumption. Each VM placement algorithm is represented by a different colour, which corresponds with the colour schemes of Figures 4.2 and 4.4. The respective high-performance reference is shown with a full marker, whereas the result due to eco-mode is represented by an open marker. A line is drawn between these points to emphasise the impact due to configuring the hosts with eco-mode.

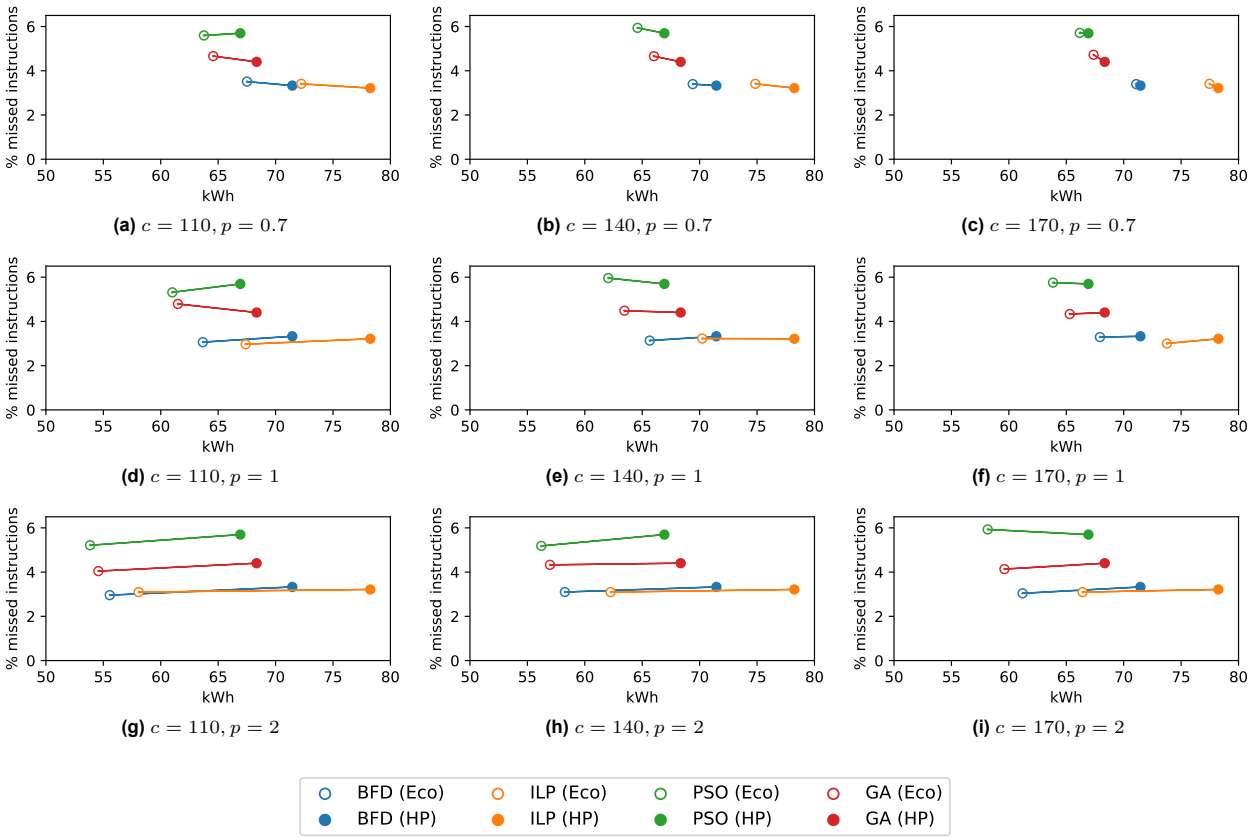


**Figure 4.5:** Comparison on performance and energy metric resulting from experiment 1 for different power curves

When examining the performance metric in Figure 4.5, the impact due to eco-mode is relatively small, whereas the energy consumption is much more affected when configuring all hosts with eco-mode. This can be seen since almost all lines in Figure 4.5 are nearly horizontally flat. This illustrates that the risk of configuring all hosts with eco-mode is minimal since the performance (as is described for this research) remains at a similar level.

The reduction in energy consumption, however, can become very high. The savings could potentially be around 15 kWh (or 20 kWh for the ILP algorithm) when considering the power dissipation curve characterised by $c = 110$ and $p = 2$ (Figure 4.5g). In other cases, such as $c = 170$ and $p = 0.7$, the reduction in energy consumption is almost negligible.
Based on Figure 4.5, enabling the eco-mode of all servers in a cluster is almost always beneficial, even when considering the possible impact on the performance. This is especially true when $p = 1$ or $p = 2$ (middle and bottom row in Figure 4.5). These values indicate that the power dissipation curve is reduced more as a result of the P-states, as is described in Section 2.1. Moreover, for decreasing values of $c$, the energy reduction becomes more (the energy reduction in the left column is larger than in the middle column, which is larger than the reduction in the right column). This shows that C-states that are capable of reducing the idle power more result in a larger reduction in energy. The increase in energy consumption reduction as a result of decreasing values of $c$ is smaller than that of increasing values of $p$.
Besides analysing the effects which result from the different power dissipation curves, the different VM placement strategies can be analysed too based on Figure 4.5. Firstly, it is clear that the performance of the BFD and ILP algorithms is better since the percentage of missed instructions is lowest for these algorithms for any power dissipation curve. Furthermore, the PSO algorithm always performs worst.
Regarding energy consumption, the BFD algorithm, the PSO algorithm and the GA have similar results. The ILP algorithm always has a higher consumption than any of the other algorithms.
The most desired position in the graphs of Figure 4.5 is the left bottom as this region indicates the lowest energy consumption and the lowest percentage of missed instructions. In almost all cases the BFD algorithm is closest to this most desirable region. However, this depends on how strict the requirements for energy consumption are. When the energy consumption should be lowest, the PSO algorithm outperforms the BFD algorithm.
Currently, data center operators often value the performance metric higher than the energy consumption metric [9]. In that case, the BFD algorithm is most preferred, based on Figure 4.5.

Experiment 1 shows that energy consumption could be significantly reduced by configuring all hosts with eco-mode. The order of reduction depends mostly on the power dissipation curves of the hosts in the server cluster and slightly on the VM placement algorithm that is used. Furthermore, the performance of the cluster could even improve under the right circumstances, but there is a risk that it degrades. This is mostly dependent on the influence the P-states have on the power dissipation curve. Based on the results of this experiment, a server that has a power dissipation curve similar to that depicted in Figure 3.3g is desired. The preferred algorithm for the VM placement on that server cluster is the BFD algorithm.

## 4.2. Exp. 2: Power curve shapes versus VM dimensions

Experiment 2 has been set up to get more insight into the relationship between the VM dimensions and the impact of eco-mode for different power curves. Besides that, the results can be used to verify the observations of experiment 1 regarding the impact of different power dissipation curves. The VM placement algorithm is the same for each simulation in this experiment. The best fit decreasing (BFD) algorithm is used since its computational complexity is the lowest, so the run time of these simulations is the shortest and it shows the most promising results in experiment 1.
Just as for experiment 1, the energy consumption and percentage of missed instructions after simulating one day are measured. Figure 4.6 shows the energy consumption with parameters $c = 110$ and $p = 2$. The vertical axis shows the energy consumption for simulating one day. Each different pattern on the blue bar represents a different VM type distribution. The bars are all blue, since for all the simulations the BFD algorithm is used and the blue bars are also used for the BFD algorithm in experiment 1. The results due to configuring all hosts with high-performance mode for that respective distribution is shown by the white box with black edges.

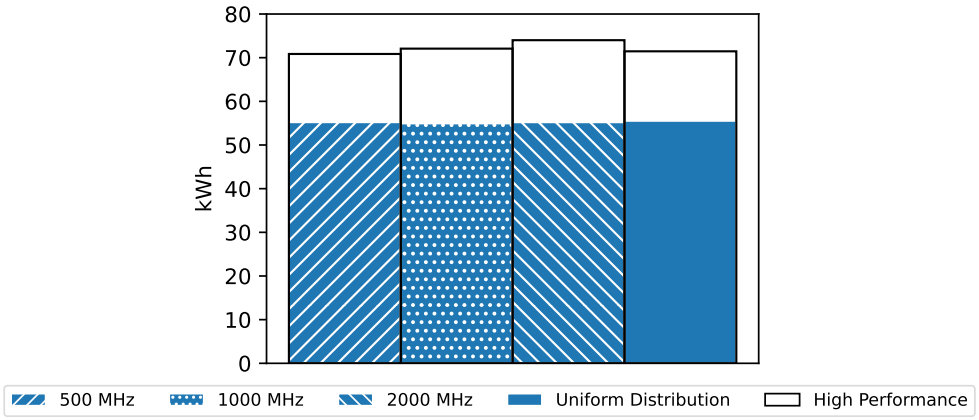


**Figure 4.6:** Energy consumption of a server cluster (with $c = 110$ and $p = 2$) for different VM types

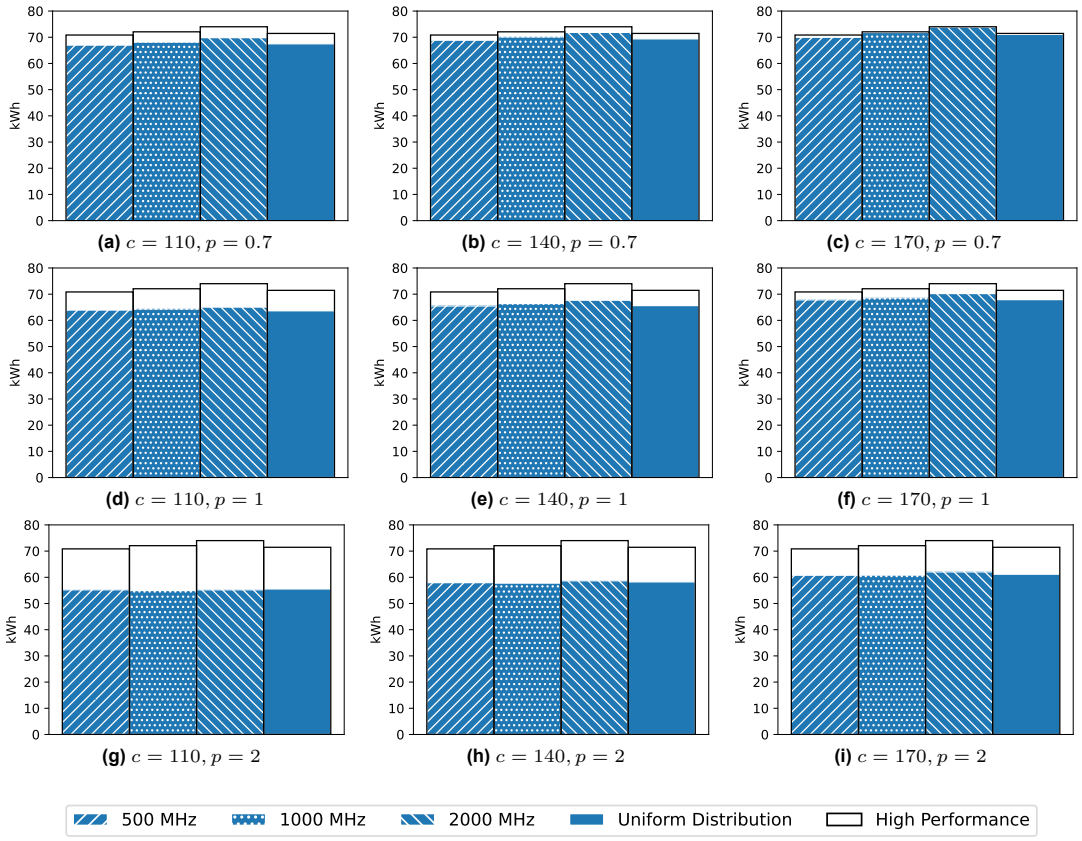


**Figure 4.7:** Energy consumption for different power curves and VM dimensions

The results for all the other power dissipation curves are shown in Figure 4.7. Note that the expected requested computational power (both overall and at any given time in the simulation) is the same for each simulation run. Thus it is expected to have an (approximately) equal energy consumption for each of the simulations in this experiment. Figure 4.7 shows that the energy consumption when having VMs that can maximally request 2000 MHz (for any given power dissipation curve), instead of 1000 MHz or 500 MHz, the energy consumption is slightly higher.

Besides calculating the energy consumption for each simulation, the percentage of missed instructions is also calculated. The results for a power dissipation curve characterised by $c = 110$ and $p = 2$ are shown in Figure 4.8. Each bar represents a different VM type distribution. The black edges around each of these bars, with sometimes a white box on top of the blue bar are the high-performance reference for the respective VM type distribution. The vertical axis shows the percentage of missed instructions.

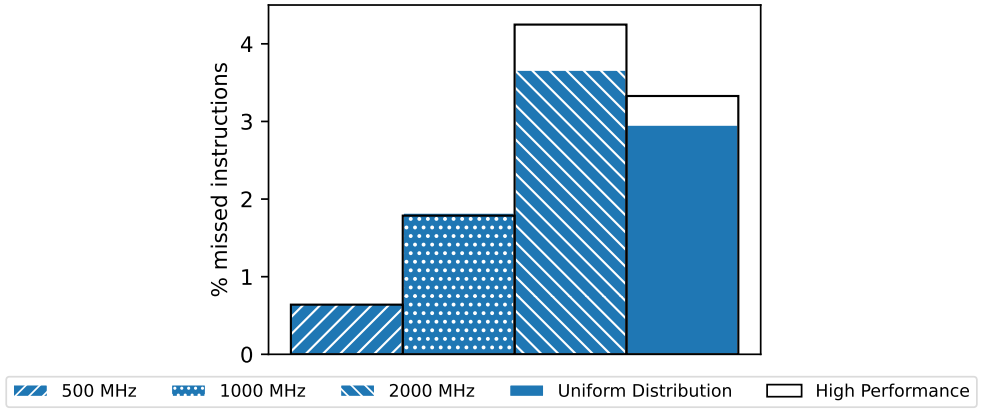


**Figure 4.8:** Percentage of missed instructions of a server cluster (with $c = 110$ and $p = 2$) for different VM types

Figure 4.9 shows the result regarding the performance metric for all different power dissipation curves. The characteristics that describe these curves, the values for $c$ and $p$, are shown in the caption for each bar graph.

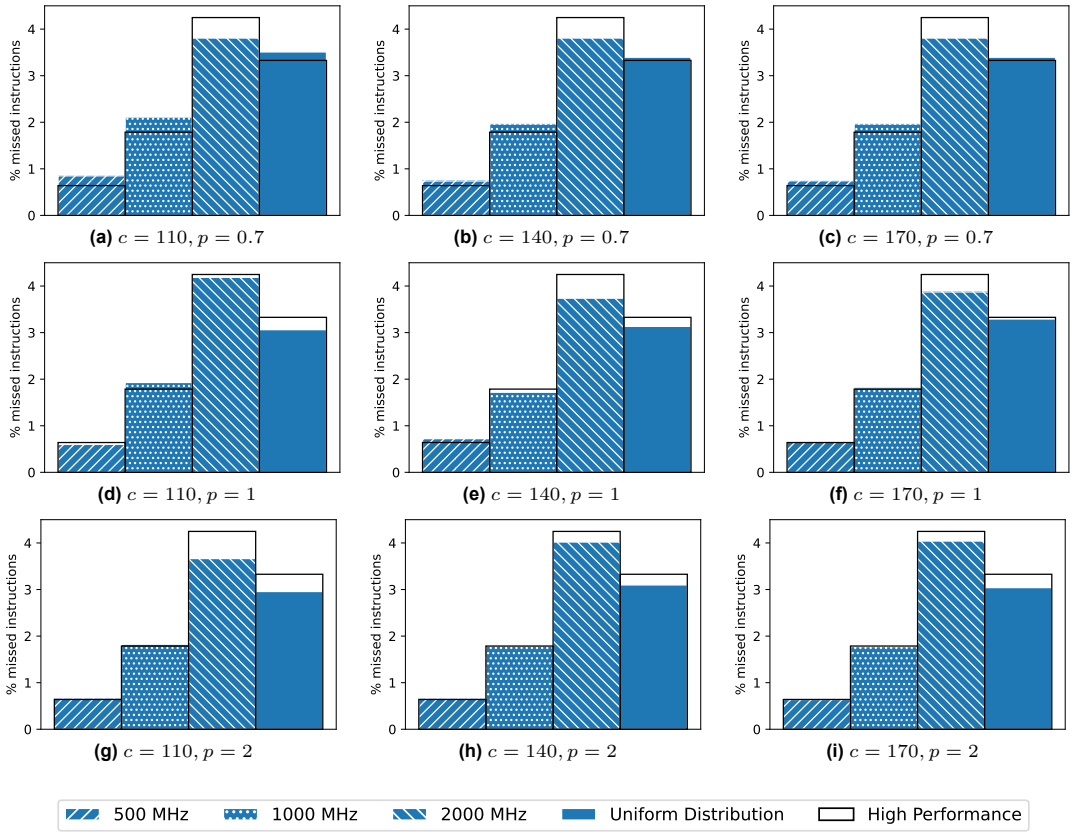


**Figure 4.9:** Percentage of missed instructions for different power curves and VM dimensions

It is immediately clear that the (average) computational request correlates with the percentage of missed instructions. When the workload distribution consists only of VMs with a maximum computational request of 2000 MHz, the percentage of missed instructions is the highest, whereas the percentage of missed instructions is very small for distributions that consist only of VMs with a maximum computational request of 500 MHz.
This result is expected due to the stochastic nature of the utilisation level of the VMs. The variance for VMs that can have a high maximum request is larger and thus the probability of under-allocation increases. This effect will be discussed more elaborately in Chapter 5.

Besides analysing the performance and energy metric individually, an analysis that combines these metrics gives more insight into whether to use eco-mode or not and which VM type distribution is preferred regarding the energy consumption and performance delivered to these VMs. Figure 4.10 shows the performance metric (percentage of missed instructions) on the vertical axis and the energy consumption on the horizontal axis. Each different marker shape represents a different VM type distribution. The high-performance reference marker is a full marker whereas the eco-mode marker is shown by an open marker. Each different graph in Figure 4.10 represents a different power dissipation curve. The characteristics of these curves are shown in the caption of each graph. The lines emphasise the impact due to eco-mode for that specific VM type distribution for the power dissipation curve of that graph.

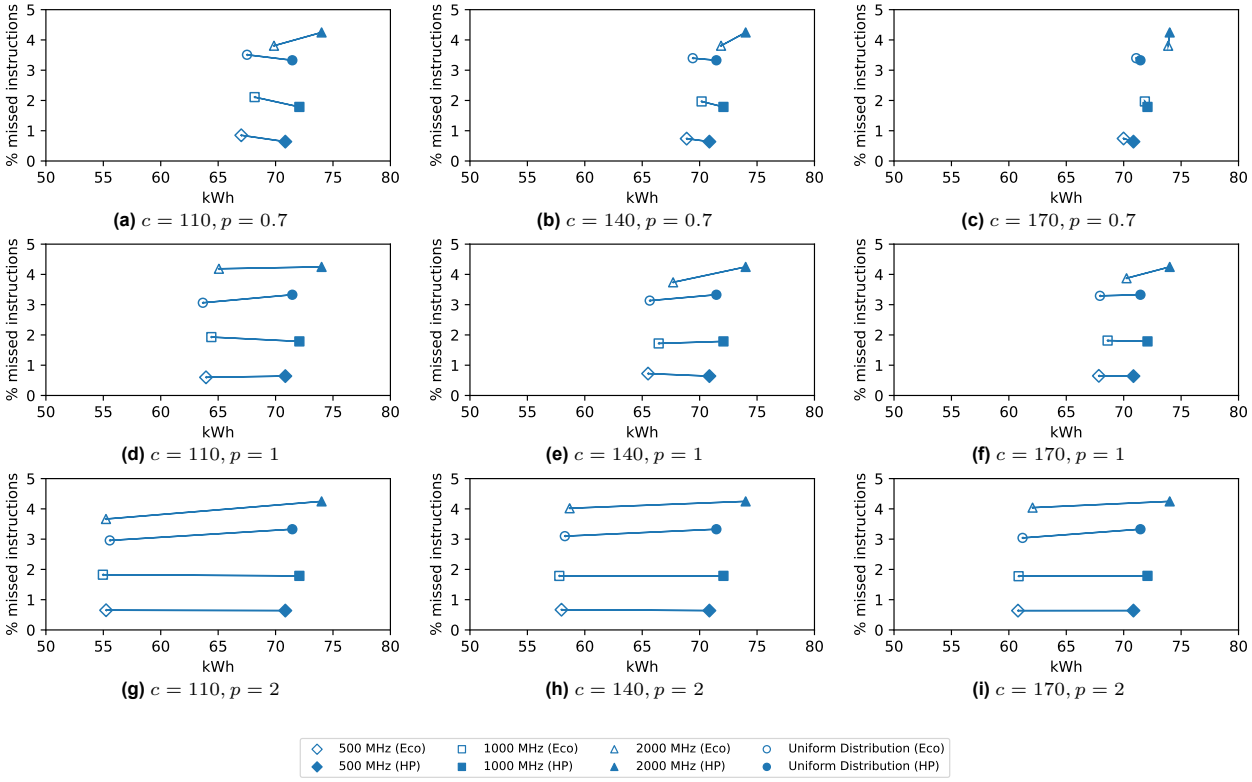


**Figure 4.10:** Comparison on performance and energy metric resulting from experiment 2 for different power curves

Since in almost all the graphs of Figure 4.10 all the results are nearly flat horizontal lines. Thus the impact on the performance is negligible. The change on the horizontal axis, which represents the impact on energy consumption, is for certain power dissipation curves much larger. The reduction in energy consumption for this experiment follows a similar pattern as for experiment 1. Thus, for decreasing values of $c$ (columns from right to left), the reduction in energy consumption increases. For increasing values of $p$ (rows from top to bottom), the reduction in energy consumption becomes larger. The reduction due to increasing values of $p$ is larger than that for decreasing values of $c$.
When examining the different VM type distributions, each graph has a similar pattern. The percentage of missed instructions is nearly linear with the average computational request by its VMs. Thus, the percentage of missed instructions when using only VMs of size 2000 MHz is approximately four times as much compared to the simulation where each VM maximally requests 500 MHz.
Moreover, the energy consumption for each distribution is the nearly same for all of the graphs. The apparent differences become smaller when the energy reduction due to configuring all hosts with eco-mode is larger.

Experiment 2 confirms the findings of experiment 1 regarding energy consumption. Thus the energy consumption will always decrease when configuring all hosts with eco-mode. The potential depends on the power dissipation curve of the hosts. The VM type distribution has little to no influence on energy consumption. However, the used VM type distribution influences both the impact due to eco-mode regarding the server cluster's performance as well as the performance in general. A larger VM type distribution has worse performance in general, but configuring all hosts with eco-mode has the most impact on this distribution.

## 4.3. Exp. 3: VM placement algorithms versus VM dimensions

This last experiment is mostly to verify the results of experiment 1 regarding the VM placement algorithms and the results of experiment 2 regarding the workload distributions. This experiment thus varies the different VM placement algorithms and the different workload distributions. These simulations are done for the high-performance power curve and the eco-mode power curve with $c = 110$ and $p = 2$ (depicted in Figure 3.3g). This specific eco-mode power curve is chosen since it performs best both in reducing energy consumption and reducing the percentage of missed instructions.
The same metrics as for the other experiments are used. Each bar represents a combination of one algorithm with one workload distribution. The colours that are used represent different algorithms and match that of experiment 1 and the patterns that are used represent a different workload distribution match that of experiment 2.
The energy consumption for one day for these simulations is shown in Figure 4.11. The energy consumption for the different workload distributions differs only slightly for all different VM placement algorithms.
The PSO algorithm and the ILP algorithm show a different pattern for the different VM type distributions than the BFD algorithm and the GA. The PSO algorithm and ILP algorithm achieve lower energy consumption when the workload consists of only large (2000 MHz) VM instances, whereas the inverse can be seen for the BFD algorithm and the GA. Thus there is a dependency between the VM type distribution and the VM placement strategy. This research does not cover the details of this dependency.

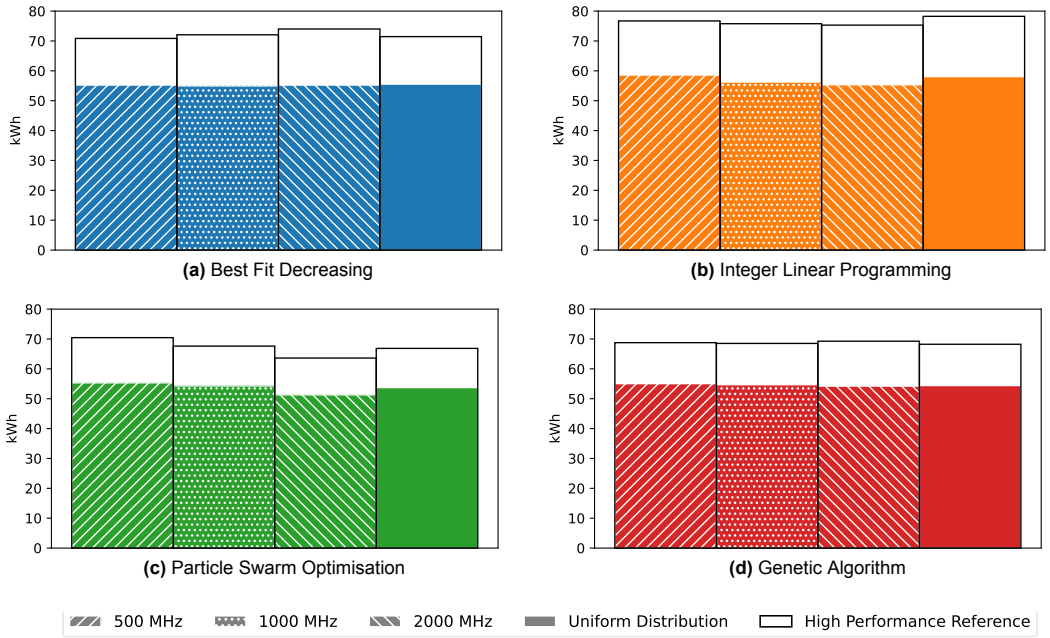


**Figure 4.11:** Energy consumption for different VM placement algorithms and VM dimensions

To compare the energy consumption as a result for the different VM placement algorithms, the average value is taken over all the different VM type distributions. These values are shown in Table 4.1. The PSO algorithm has the lowest energy consumption when the hosts are in high-performance mode as well as when they are in eco-mode. The ILP algorithm always has the highest energy consumption.

| VM placement algorithm | High-Performance Mode | Eco-mode |
|---|---|---|
| BFD algorithm | 72.1 kWh | 55.2 kWh |
| ILP algorithm | 76.5 kWh | 57.1 kWh |
| PSO algorithm | 67.1 kWh | 53.7 kWh |
| Genetic algorithm | 68.7 kWh | 54.6 kWh |

**Table 4.1:** Average energy consumption per VM placement strategy

However, the energy reduction due to enabling eco-mode is not equal for all the different VM placement algorithms. The impact of eco-mode on the reduction of energy consumption is larger for the algorithms that have a worse absolute performance. Thus, the impact is largest on the ILP algorithm and smallest on the PSO algorithm, as can be seen by the averages shown in Table 4.1. This can also be seen in the results of experiment 1.
All in all, the results regarding the energy consumption of this experiment are in line with the results of experiments 1 and 2. This thus confirms the findings of these experiments.

The percentage of missed instructions for the simulations of this experiment is shown in Figure 4.12. When examining these results regarding the different workload distributions, the distribution consisting of only VMs with a high maximum computational request has the worst performance. To a further extent, the higher the (average) maximum computational request, the higher the percentage of missed instructions. This is in line with the results of experiment 2.

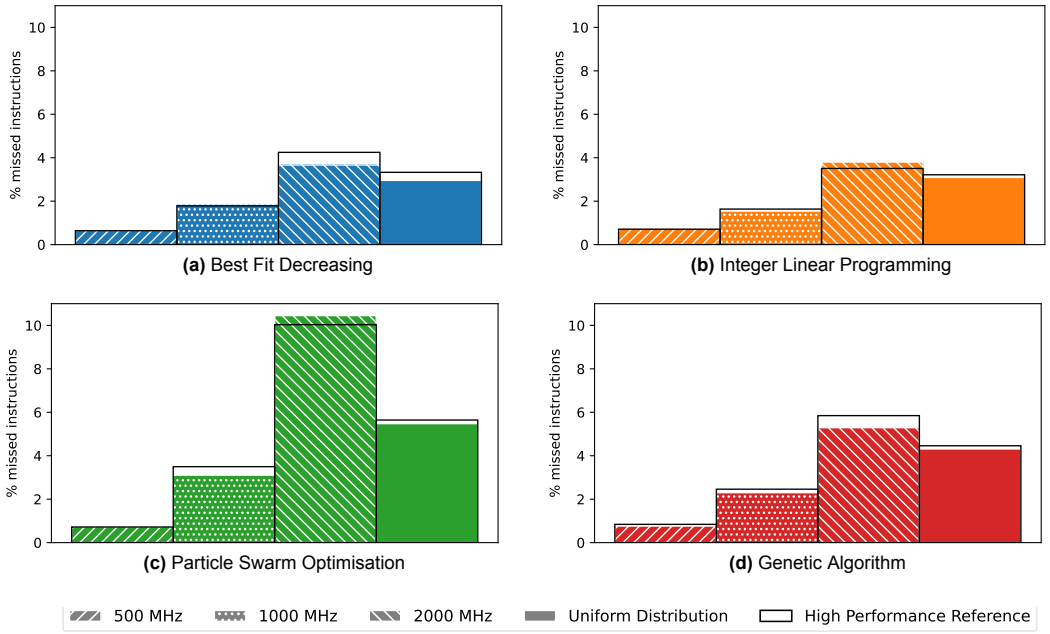


**Figure 4.12:** Percentage of missed instructions for different VM placement algorithms and VM dimensions

Moreover, the PSO algorithm has the worst performance. The percentage of missed instructions is significantly higher than for the other algorithms, especially when only large VM instances (2000 MHz) are applied. The other three algorithms' performances are relatively similar. The GA performs slightly worse than the BFD algorithm and ILP algorithm. These results are similar to those of experiment 1. Furthermore, the percentage of missed instructions decreases when applying the GA. This is the only algorithm that has performance enhancement for any of the workload distributions.

## 4.4. Summary of results

All in all, the different power curves due to enabling the eco-mode always reduce the energy consumption of a data center. Having energy-efficient P-states could potentially lead to an improved performance. Moreover, when considering both the performance and energy consumption, the BFD algorithm seems to perform the best.
The degree of impact eco-mode can have on the server cluster's energy consumption and performance is dependent on the VM placement algorithm and VM type distribution. However, the general impact is always the same, which is that the energy consumption is reduced and the percentage of missed instructions could potentially decrease too.

# 5

# Discussion

This chapter discusses how to interpret the results as presented in Chapter 4. Furthermore, this chapter identifies the gaps and limitations of this research, recognising the opportunities to enhance the robustness of this research by refinement. Recommendations for further research is provided in Section 6.2

Configuring a host with the eco-mode effectively means enabling the capability to switch between core states (C-states) and performance states (P-states). Experiment 1 (Section 4.1) and experiment 2 (Section 4.2) have illustrated that these states allow for reducing the energy consumption of a data center significantly. Moreover, when the P-states are cause for an energy-efficient power dissipation curve, the percentage of missed instructions could improve rather than degrade.
For this simulation, an energy reduction of 15 kWh in a day could be achieved for 50 virtual machines (VMs) on 50 servers. Many medium-sized data centers have over a thousand servers. This is 20 times as much as the environment simulates in this research. When extrapolating these results, this would mean an energy reduction of 15x20 = 300 kWh within one day for one data center. This is equal to approximately 44 households on average [74]. The potential could be even higher since none of the algorithms ever use all 50 of the servers. However note that the simulations did not accurately model this situation, so this potential is not exact.
When considering the performance metric, the change due to configuring all hosts with eco-mode is almost negligible. Some slight changes might occur, as the percentage of missed instructions, experienced as delay or a slower application, increases when $p = 0.7$ whereas it decreases when $p = 2$. This means that having energy-efficient P-states might help the performance of a server cluster.
These results that might improve a cluster's performance are experienced when $p = 2$ when using Equation 2.1. However, this is the only value larger than one, meaning the only exponentially shaped power dissipation curve. This research has no analysis regarding different values of $p$ larger than one, which could give more insight into the importance of the exponential order of the power curve.
An exponential power dissipation curve shows a smaller performance degradation or even slight performance enhancement since the increase in power dissipation of placing a VM on a new host that was shut down before is similar to placing a VM on a host that becomes highly utilised as a result of that placement. Thereafter, exploring different exponential power dissipation curves would provide more insight into the importance of the order of exponentiality.
To a further extent, the power curves have all been parameterised and not deducted from an analysis of actual servers. However, realistic values have been used based on the research of Rteit et al. [16] The reference taken from this research is under a workload consisting of stochastic simulation in Java (ssj). This is most suitable for the CloudSim tooling and it resembles enterprise applications the most. Using measurements of the actual servers to model the power dissipation curves would result in a more realistic simulation.
To a further extent, the simulation environment would be more realistic in a heterogeneous environment [71], implying hosts that have different power curves and different resource capacities. Analysing a heterogeneous environment could lead to significantly different results.
Even though this research does not perfectly simulate a realistic data center scenario, it does indicate

that there is a potential reduction in energy consumption whilst maintaining, or even improving the data center's performance.

Moreover, this research gives insight into a small set of virtual machine (VM) placement policies that all aim to minimise a server cluster's energy consumption. The results indicate that they are all very competitive regarding energy consumption, as can be seen in experiment 1 (Section 4.1) and experiment 3 (Section 4.3. The particle swarm optimisation (PSO) algorithm outperforms the others with a slight difference. On the contrary, the PSO algorithm has a significantly higher percentage of missed instructions compared to the other algorithms. The best fit decreasing (BFD) algorithm and ILP algorithm perform the best regarding this metric.
Since this research does not propose a new VM placement algorithm, the VM placement policies that are used are deducted from the literature. After that, the parameters used have not been adapted for the scope of this research which could potentially lead to different results.
Moreover, the objective functions for all these algorithms are (or have been slightly altered to be) mono-objective. Therefore, the objective of all these algorithms is to minimise a server cluster's energy consumption, whereas having multi-objectivity could enhance the performance metric of some of these algorithms.
To a further extent, the objective function slightly differs for the used VM placement algorithms. This could explain why the ILP algorithm results in a higher energy consumption than the other algorithms. The ILP algorithm's objective is to minimise the active number of hosts, implying that this would result in a minimised energy consumption. This is not necessarily true, but this design is chosen to reduce the run time of this algorithm. Even with this implementation, it takes very long.
On further inspection of the functioning of the ILP with the implementation for this research, it does not function accordingly. It cannot minimise the number of active hosts compared to the algorithms, as can be seen in Appendix C. This seems to cause the unexpected results of the ILP algorithm. This also explains the good score on the performance metric since the VMs are spread over more hosts.
Moreover, the BFD algorithm does not consider the energy consumption of the data center (or server cluster) as a whole, but only for the VM that it is placing. This is done to reduce computational complexity and thereafter run time compared to algorithms such as the PSO algorithm and GA.
Besides, the cause for the PSO algorithm and the GA to have a higher percentage of missed instructions than the BFD algorithm is that the PSO algorithm and the GA can find a more efficient placement of the VMs. This results in lower energy consumption but a higher probability of under-allocation since the placement is based on the utilisation levels of the previous time frame.
As explained based on Figure 4.5, the BFD algorithm scores best overall since it has (almost) the best performance of all algorithms regarding the performance metric and is very competitive with the PSO algorithm and the GA regarding the energy consumption metric. It is also the least complex algorithm and thus very quick. Thus, based on this research, the BFD algorithm seems most promising within the context of VM placement.

Furthermore, the results of experiment 2 (Section 4.2) and experiment 3 (Section 4.3) show that the workload distributions as described in this research do not influence the energy consumption of a data center significantly. On the contrary, a data center's performance is affected by the workload distribution. The higher the maximum computational request of the VMs, the higher the percentage of missed instructions.
The latter is probably due to the stochastic nature of the set of instructions that needs to be performed on each of the VMs. The utilisation level of a VM is based on a uniform distribution, meaning that the variance is higher when the maximum computational request is higher. When the variance is higher, the mismatch by the VM placement algorithm is often higher regarding the number of missed instructions resulting in a larger percentage of missed instructions, since the total number of instructions remains the same.
Furthermore, the simulations do not consider the creating of new or ending of VMs within this simulation. In practice, VMs have an end of life when they have served their purpose or when the application is changed in such a way that the configuration changes. Moreover, over time, new VMs are added due to various reasons such as the launch of a new application. This aspect could influence the results but has not been considered due to time limitations.
Moreover, the stochastic nature of the utilisation levels of the VMs means that there is no real-life-based

workload regarding the utilisation level of the VMs. A uniform distribution of utilisation levels has been chosen to ensure that VMs are sometimes very highly or lowly utilised. Moreover, it constrains the utilisation levels between 0 and 1 (0 and 100%).
Even though the utilisation levels of the VMs vary, the sum of the expected and/or the sum of the maximum utilisation of all VMs does not approach the data center's full capacity. In this way, the restriction of not being able to place all VMs is not considered and affects the data center's performance and energy consumption. Since this most probably will diminish the impact due to configuring all hosts with eco-mode and since many data center do not have a workload near their full capacity, this situation is not considered for this research.
The utilisation levels have only been described for the computational power since the other resources have not been considered. Especially the remote access memory (RAM) could alter the results according to Rteit et al. [16] Their research finds that the benefits of enabling eco-mode are larger for a memory-bound workload. The workload used in this research is defined as CPU-bound. This is chosen due to the limitations and complexity of CloudSim. However, the results are still promising, even though eco-mode is thus suggested to be less effective in a CPU-bound environment.
Furthermore, some data centers have more specific use cases, such as telecom operators like KPN. These operators provide a stable and secure network, which is much more bandwidth-bound. The requested bandwidth thus varies much over time and should be considered in research to provide a more accurate prediction on utilising eco-mode in its server clusters.
Based on the results of experiment 1 and experiment 2, neither the VM placement algorithms nor the VM workload distribution correlates with the eco-mode. This implies that a decision on whether or not to configure hosts with eco-mode can be made independently of the used VM placement policy or VM workload distribution.
However, since this research has not investigated the hyper-parameters for the different VM placement algorithms, no statement could be given whether one of these parameters correlates with a parameter from eco-mode. When analysing or proposing a new VM placement algorithm, this should be considered.
The results of experiment 3 indicate that there is a certain correlation between the VM placement policy and the maximum request in computational power of the VMs. The exact relationship between these can not be deducted from the results of this experiment. Rather, some evident statements can be made based on the results as is done in Section 4.3.
Based on experiments 2 and 3, it is clear that the percentage of missed instructions for the VM type distribution consisting of only VMs that maximally request 2000 MHz (or MIPS) is significantly higher. A possible explanation for this is that the algorithms predict the utilisation levels of all VMs at time $t+1$ to be equal to the workload of that VM at time $t$. Assuming that nothing is known about the history of the utilisation and nothing is known about the distribution, this is the most accurate predictor. This is known as the martingale distribution.
Since the distribution is known, due to the configuration for this simulation, one could argue that using the expected value, which is equal to half of the maximum computational request in this case, would be a better predictor. It is to be expected that this decreases the number of missed instructions, especially for the workload distributions that consist of large VM instances.

On a more general note, the performance metric used in this research does not cover the overall performance of a data center (or a server cluster). The reason to only choose this metric is due to the limitations of the CloudSim tooling. Other performance metrics are measured in CloudSim too, which are all similar alternatives or derivations of the performance metric used in this research and thus do not provide different insights.
Other metrics such as the delay due to eco-mode are not considered. Configuring hosts with eco-mode will (almost) certainly lead to delays in executed workload since it will lower frequency to save power. The order of these delays on a cluster level within a cloud (or dynamic) environment is not known, but is assumed to be negligible for this research based on the research of Huang et al. [23].

All in all, the simulations provided in this research show a promising potential when enabling the eco-mode for all servers in a server cluster. However, the environment used in this research is abstracted too much from a realistic scenario and no measurements on an actual server cluster have been done to give sufficient insights to implement these results in an operating data center.

# 6

# Conclusions

This chapter answers the research questions posed in Section 1.5. Furthermore, recommendations are given for further research. Moreover, some recommended practices for data center operators are provided.

## 6.1. Conclusions

This research provides a framework that combines the energy-saving technologies that are fundamental to eco-mode within the complex system of virtual machine (VM) placement. Eco-mode is defined as the capability of a server to switch dynamically between C-states and P-states. The impact as a result of configuring all servers to eco-mode in a server cluster is analysed for multiple VM placement policies. The VM placement policies that have been used for comparison are the best fit decreasing (BFD) algorithm, the integer linear programming (ILP) algorithm, the particle swarm optimisation (PSO) algorithm and the genetic algorithm (GA).

Configuring servers with eco-mode effectively enables the capability to switch between the different C-states and P-states. The technique of switching between these states, both C-states and P-states, is developed to conserve energy on a server. When configuring all servers in a cluster with eco-mode, potentially, a significant reduction in energy consumption is obtained. This depends on the technologies that switch between the C-states and P-states.

Energy-efficient dynamic frequency and voltage scaling (DFVS) (by switching between P-states) can alter the power dissipation curve of a server into an exponentially shaped curve, whereas advanced technologies for C-states result in the power dissipation of a server being lower when the server is (nearly) idle. DFVS thus creates a larger overall difference compared to the power dissipation curve of a server in high-performance mode, which is the reason why the potential energy savings are larger for advanced technologies for the P-states compared to advanced technologies of the C-states.

However, the P-states are assumed to have a larger impact on the performance than the C-states on a cluster level. This research has demonstrated that this is not true for an energy-efficient power dissipation curve due to the P-states. To a further extent, well-developed P-states could improve the performance of a cluster. Moreover, the C-states neither have a positive nor a negative impact on the performance of a cluster when enabled.

Note that this research has not considered the delays caused by these states. Especially the P-states are prone to cause delays since it lowers the operational frequency of the servers. The lowering of this frequency will cause instructions to be executed more slowly.

Furthermore, the choice for a VM placement policy can be made independent of the decision on whether to configure servers with eco-mode or not. The choice for VM placement policy neither affects the energy savings due to eco-mode nor the performance degradation/improvements due to eco-mode.

Besides, within the scope of this research, the BFD algorithm seems to perform best overall. This algorithm does not result in the lowest energy consumption but is relatively close to the results of the other algorithms and scores best on the performance metric used in this research.

Lastly, the computational resource dimensions of the VMs that need to be hosted on the server cluster

do not affect the impact made by eco-mode regarding the potential energy savings. The performance degradation/improvement due to eco-mode is affected by the computational resource requests of the VMs. For a larger request, the performance improves (more).
Even though configuring servers with eco-mode has a more positive influence for VMs that have a large(r) request in its computational resource, a server cluster would meet its performance requirements more easily when the workload consists of many small VM instances.
All in all, the potential of configuring servers with eco-mode on all servers in a cluster shows promising potential energy savings and potentially could improve performance. However, this is based on simulations and not yet actual measurements, so more research is recommended.

## 6.2. Recommendations for further research

This section addresses some opportunities and aspects that have not been touched upon in this research which could be explored in further research.
Firstly, creating a simulation environment that is closer to a real-life scenario. This consists of multiple aspects, that have not been considered within the scope of this research. One of the aspects that should be considered is the memory resource. This is a resource that plays a substantial role in many VM placement policies. The memory capacity of a server could be fully utilised whilst the computational resource is not nearly fully utilised. Especially in enterprise applications of data centers, this is the case. However, as shown by Rteil et al. [16], this might cause an even bigger potential for configuring servers with eco-mode.
Furthermore, some data centers such as those for telecom operators have a more defined use case. The applications that run on the servers within these data centers are often network bandwidth-bound. It would prove valuable to model an environment that covers these dynamics more accurately.
Moreover, creating heterogeneity in the simulation environment would give more useful insights. This consists of creating a simulation environment that consists of hosts that have a variety in their resource capacities and a variety in their power dissipation curve. An environment that resembles certain real-life use cases should be carefully chosen, such as the aforementioned environment for telecom operators. Besides, the workload that needs to be executed on the VMs could be chosen more carefully to resemble real-life scenarios. The reasoning for choosing a specific workload depends on the use case that is researched.
Furthermore, different performance metrics can be analysed that consider the delays caused by configuring servers with eco-mode. The performance metric used in this research lacks this information. The CloudSim tooling has a variety of performance metrics which are all related to a misallocation of VMs. However, performance metrics such as actual delays, which can be caused by DFVS for example, are not considered sufficiently in this research. New metrics should be designed, within the perimeters of CloudSim, to cover other performance metrics such as delays.
Lastly, research into the use case of containers, or a hybrid version of containers and virtual machines should be considered. Containers are used more and more and will increase the dynamicity of cloud computing.

## 6.3. Recommendations for common practices

This section provides some recommendations on how to implement these findings as a data center operator. Before configuring all servers within a data center with eco-mode, an analysis of the specifics of the data center should be made.
Firstly, one should research the power dissipation curves of the servers that are placed within the data center or a cluster. The analysis of these power dissipation curves should be analysed with a typical set of instructions (thus a typical application) running on these servers and the load should vary from very low to very high.
Once these power dissipation curves are known, one could estimate whether it is beneficial to configure a server with eco-mode. When the power dissipation curve shows an exponential relationship when compared to the load on that server, there is little to no risk for configuring that server with eco-mode.
When the power curve shows a more linear or root-like relationship, the data center operator should consider whether a small performance degradation would be acceptable. Performance degradation is mostly experienced in the form of queries taking longer to process, thus having a slightly slower application.

So far the enabling of eco-mode, a data center operator could also consider having a joint analysis with the provider of the hypervisor about the VM placement policy. This research has not proposed a novel VM placement policy, but in general, these policies could be aligned with the objectives of a data center operator. A data center operator can have multiple objectives such as minimising the energy consumption of a data center whilst meeting certain performance requirements such as a maximum delay.
Lastly, designing the workload in many small VM instances would have better results regarding both energy consumption and performance. This research has not provided a framework for containerised applications, but due to the assumptions made in this research, the results of a containerised workload will probably be comparable to having many small VM instances.

# References


[1] Vida Rozite, Emi Bertoli, and Brendan Reidenbach. *Data centres & networks*. en. URL: https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks.

[2] Mar. 2021. URL: https://www.dutchdatacenters.nl/en/factsheet/.

[3] Strategic Media Asia Limited, Mar. 2020. URL: https://green-data.blogspot.com/2020/03/electrical-distribution-system-data-center.html.

[4] Fawaz AL-Hazemi et al. "Dynamic allocation of power delivery paths in consolidated data centers based on adaptive UPS switching". In: *Computer Networks* 144 (2018), pp. 254–270. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.08.004. URL: https://www.sciencedirect.com/science/article/pii/S1389128618307138.

[5] Rădulescu Constanța Zoie, Rădulescu Delia Mihaela, and Sipică Alexandru. "An analysis of the power usage effectiveness metric in data centers". In: *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*. 2017, pp. 1–6. DOI: 10.1109/ISEEE.2017.8170650.

[6] *Data center average annual power usage effectiveness (PUE) worldwide 2007-2022*. en. https://www.statista.com/statistics/1229367/data-center-average-annual-pue-worldwide/. Accessed: 2023-9-28.

[7] Silvio Marcacci. *How Much Energy Do Data Centers Really Use?* Energy Innovation: Policy and Technology. Mar. 17, 2020. URL: https://energyinnovation.org/2020/03/17/how-much-energy-do-data-centers-really-use/ (visited on 09/28/2023).

[8] UEFI. *Advanced configuration and power interface specification*. URL: https://uefi.org/sites/default/files/resources/ACPI_5_Errata_A.pdf.

[9] Dirk Harryvan, Marco Verzijl, and Max Amzarakov. *Analysis LEAP Track 1 "Powermanagement"*. Tech. rep. Accessed: 17-2-2023. Certios and WCooliT.

[10] W. Lloyd Bircher and Lizy K. John. "Analysis of dynamic power management on multi-core processors". In: *Proceedings of the 22nd annual international conference on Supercomputing* (2008). DOI: 10.1145/1375527.1375575.

[11] Jan M. Rabaey and Massoud Pedram. *Low power design methodologies*. Vol. 336. The Springer International Series in Engineering and Computer Science. Springer-Verlag.

[12] Intel Corporation. *Intel Idle Driver for Linux*. Accessed: 2023-12-2023. URL: https://github.com/torvalds/linux/blob/master/drivers/idle/intel_idle.c.

[13] Xin Zhan et al. "CARB: A C-State Power Management Arbiter for Latency-Critical Workloads". In: *IEEE Computer Architecture Letters* 16.1 (2017), pp. 6–9. DOI: 10.1109/LCA.2016.2537802.

[14] Jawad Haj Yahya et al. "Agilewatts: An energy-efficient CPU core idle-state architecture for latency-sensitive server applications". In: *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* (Aug. 2022). DOI: 10.1109/micro56248.2022.00063.

[15] John Beckett. *BIOS Performance and Power Tuning Guidelines for Dell PowerEdge 12th Generation Servers*.

[16] Nour Rteil et al. "Balancing Power and Performance: A Multi-Generational Analysis of Enterprise Server BIOS Profiles". In: *2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*. 2022, pp. 81–85. DOI: 10.1109/GECOST55694.2022.10010599.

[17] Konstantin Moiseev, Avinoam Kolodny, and Shmuel Wimer. "Timing-Aware Power-Optimal Ordering of Signals". In: *ACM Trans. Des. Autom. Electron. Syst.* 13.4 (Oct. 2008). ISSN: 1084-4309. DOI: 10.1145/1391962.1391973. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/1391962.1391973.

[18] Paul Horowitz and Winfield Hill. *The Art of Electronics*. 3rd. USA: Cambridge University Press, 2015. ISBN: 0521809266.

[19] Mark Weiser et al. “Scheduling for Reduced CPU Energy”. In: *First Symposium on Operating Systems Design and Implementation (OSDI 94)*. Monterey, CA: USENIX Association, Nov. 1994. URL: https://www.usenix.org/conference/osdi-94/scheduling-reduced-cpu-energy.

[20] Akihiko Miyoshi et al. “Critical power slope: understanding the runtime effects of frequency scaling.” In: Jan. 2002, pp. 35–44. DOI: 10.1145/514191.514200.

[21] Chia-Ming Wu, Ruay-Shiung Chang, and Hsin-Yu Chan. “A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters”. In: *Future Generation Computer Systems* 37 (2014), pp. 141–147.

[22] Lizhe Wang et al. “Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS”. In: *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE. 2010, pp. 368–377.

[23] Pengcheng Huang et al. “Energy efficient dvfs scheduling for mixed-criticality systems”. In: *Proceedings of the 14th International Conference on Embedded Software*. 2014, pp. 1–10.

[24] Nikzad Babaii Rizvandi et al. “Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms”. In: *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE. 2010, pp. 388–397.

[25] David Maynard Gerrard, James Edward Mooney, and Dave Thompson. “Digital preservation at Big Data scales: proposing a step-change in preservation system architectures”. In: *Library Hi Tech* 36.3 (Jan. 2018), pp. 524–538. DOI: 10.1108/lht-06-2017-0122. URL: https://doi.org/10.1108%2Flht-06-2017-0122.

[26] Michael Sindelar, Ramesh K. Sitaraman, and Prashant Shenoy. “Sharing-Aware Algorithms for Virtual Machine Colocation”. In: *Proceedings of the Twenty-Third Annual ACM Symposium on Parallelism in Algorithms and Architectures*. SPAA ’11. San Jose, California, USA: Association for Computing Machinery, 2011, pp. 367–378. ISBN: 9781450307437. DOI: 10.1145/1989493.1989554. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/1989493.1989554.

[27] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990, pp. 291–325. ISBN: 0716710455.

[28] Fabio Lopez-Pires and Benjamin Baran. “Virtual machine placement literature review”. In: *arXiv preprint arXiv:1506.01509* (2015).

[29] Haiyan Zhuang and Babak Esmaeilpour Ghouchani. “Virtual machine placement mechanisms in the cloud environments: a systematic review”. In: *Kybernetes* 50.2 (Jan. 2021). Publisher: Emerald Publishing Limited, pp. 333–368. ISSN: 0368-492X. DOI: 10.1108/K-09-2019-0635. URL: https://doi.org/10.1108/K-09-2019-0635 (visited on 12/21/2023).

[30] Alexandre H. T. Dias, Luiz. H. A. Correia, and Neumar Malheiros. “A Systematic Literature Review on Virtual Machine Consolidation”. In: 54.8 (Oct. 2021). ISSN: 0360-0300. DOI: 10.1145/3470972. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/3470972.

[31] György Dósa. “The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is FFD(I) ≤ 11/9OPT(I) + 6/9”. In: *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Ed. by Bo Chen, Mike Paterson, and Guochuan Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–11. ISBN: 978-3-540-74450-4.

[32] György Dósa and Jiří Sgall. “Optimal Analysis of Best Fit Bin Packing”. In: *Automata, Languages, and Programming*. Ed. by Javier Esparza et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 429–441. ISBN: 978-3-662-43948-7.

[33] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. “Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing”. In: *Future Generation Computer Systems* 28.5 (2012). Special Section: Energy efficiency in large-scale distributed systems, pp. 755–768. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2011.04.017. URL: https://www.sciencedirect.com/science/article/pii/S0167739X11000689.

[34] Rohit Gupta et al. “A Two Stage Heuristic Algorithm for Solving the Server Consolidation Problem with Item-Item and Bin-Item Incompatibility Constraints”. In: *2008 IEEE International Conference on Services Computing*. Vol. 2. 2008, pp. 39–46. DOI: 10.1109/SCC.2008.39.

[35] H W Lenstra Jr. “Integer programming with a fixed number of variables”. en. In: *Math. Oper. Res.* 8.4 (Nov. 1983), pp. 538–548.

[36] Daniel Nicolas Dadush. “Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation”. AAI3531709. PhD thesis. USA, 2012. ISBN: 9781267739605.

[37] Victor Reis and Thomas Rothvoss. *The Subspace Flatness Conjecture and Faster Integer Programming*. 2023. arXiv: 2303.14605 [math.OC].

[38] Rym Regaieg et al. “Multi-Objective Mixed Integer Linear Programming Model for VM Placement to Minimize Resource Wastage in a Heterogeneous Cloud Provider Data Center”. In: *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*. 2018, pp. 401–406. DOI: 10.1109/ICUFN.2018.8437036.

[39] Rym Regaieg et al. “Multi-objective optimization for VM placement in homogeneous and heterogeneous cloud service provider data centers”. In: *Computing* 103.6 (June 2021), pp. 1255–1279. ISSN: 1436-5057. DOI: 10.1007/s00607-021-00915-z. URL: https://doi.org/10.1007/s00607-021-00915-z.

[40] Laurent Perron and Vincent Furnon. *OR-Tools*. Version v9.8. Google, Nov. 15, 2023. URL: https://developers.google.com/optimization/.

[41] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[42] V. Dinesh Reddy, G. R. Gangadharan, and G. Subrahmanya V. R. K. Rao. “Energy-aware virtual machine allocation and selection in cloud data centers”. In: *Soft Computing* 23.6 (Mar. 2019), pp. 1917–1932. ISSN: 1433-7479. DOI: 10.1007/s00500-017-2905-z. URL: https://doi.org/10.1007/s00500-017-2905-z.

[43] J. Kennedy. “The particle swarm: social adaptation of knowledge”. In: *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*. 1997, pp. 303–308. DOI: 10.1109/ICEC.1997.592326.

[44] Riccardo Poli. “Analysis of the Publications on the Applications of Particle Swarm Optimisation”. In: *Journal of Artificial Evolution and Applications* 2008 (Feb. 2008). Ed. by Leonardo Vanneschi. Publisher: Hindawi Publishing Corporation, p. 685175. ISSN: 1687-6229. DOI: 10.1155/2008/685175. URL: https://doi.org/10.1155/2008/685175.

[45] Yudong Zhang, Shuihua Wang, and Genlin Ji. “A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications”. In: *Mathematical Problems in Engineering* 2015 (Oct. 2015). Ed. by Shuming Wang. Publisher: Hindawi Publishing Corporation, p. 931256. ISSN: 1024-123X. DOI: 10.1155/2015/931256. URL: https://doi.org/10.1155/2015/931256.

[46] Daniel Bratton and James Kennedy. “Defining a Standard for Particle Swarm Optimization”. In: *2007 IEEE Swarm Intelligence Symposium*. 2007, pp. 120–127. DOI: 10.1109/SIS.2007.368035.

[47] M. Clerc and J. Kennedy. “The particle swarm - explosion, stability, and convergence in a multidimensional complex space”. In: *IEEE Transactions on Evolutionary Computation* 6.1 (2002), pp. 58–73. DOI: 10.1109/4235.985692.

[48] Abdelhameed Ibrahim et al. “PAPSO: A Power-Aware VM Placement Technique Based on Particle Swarm Optimization”. In: *IEEE Access* 8 (2020), pp. 81747–81764. DOI: 10.1109/ACCESS.2020.2990828.

[49] Seyed Ebrahim Dashti and Amir Masoud Rahmani. “Dynamic VMs placement for energy efficiency by PSO in cloud computing”. In: *Journal of Experimental & Theoretical Artificial Intelligence* 28.1-2 (2016), pp. 97–112. DOI: 10.1080/0952813X.2015.1020519. eprint: https://doi.org/10.1080/0952813X.2015.1020519. URL: https://doi.org/10.1080/0952813X.2015.1020519.

[50] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Mar. 1998. ISBN: 978-0-262-28001-3. DOI: 10.7551/mitpress/3927.001.0001. URL: https://doi.org/10.7551/mitpress/3927.001.0001.

[51] Firas Gerges, Germain Zouein, and Danielle Azar. "Genetic Algorithms with Local Optima Handling to Solve Sudoku Puzzles". In: ICCAI '18. Chengdu, China: Association for Computing Machinery, 2018, pp. 19–22. ISBN: 9781450364195. DOI: 10.1145/3194452.3194463. URL: https://doi.org/10.1145/3194452.3194463.

[52] Michael C. Burkhart and Gabriel Ruiz. "Neuroevolutionary representations for learning heterogeneous treatment effects". In: *Journal of Computational Science* 71 (2023), p. 102054. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2023.102054. URL: https://www.sciencedirect.com/science/article/pii/S187775032300114X.

[53] T. Back. "Selective pressure in evolutionary algorithms: a characterization of selection mechanisms". In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. 1994, 57–62 vol.1. DOI: 10.1109/ICEC.1994.350042.

[54] Adam Lipowski and Dorota Lipowska. "Roulette-wheel selection via stochastic acceptance". In: *Physica A: Statistical Mechanics and its Applications* 391.6 (Mar. 2012), pp. 2193–2196. ISSN: 0378-4371. DOI: 10.1016/j.physa.2011.12.004. URL: http://dx.doi.org/10.1016/j.physa.2011.12.004.

[55] David B. Fogel, Thomas Bäck, and Zbigniew Michalewicz. *Evolutionary computation. Vol. 1, Basic algorithms and operators*. eng. Bristol: Institute of Physics Pub. Bristol, 2000. URL: https://search.ebscohost.com/login.aspx?direct=true%5C&scope=site%5C&db=nlebk%5C&db=nlabk%5C&AN=32720.

[56] Zbigniew Michalewicz. "Evolution Strategies and Other Methods". In: *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 127–138. ISBN: 978-3-662-02830-8. DOI: 10.1007/978-3-662-02830-8_9. URL: https://doi.org/10.1007/978-3-662-02830-8_9.

[57] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future". In: *Multimedia Tools and Applications* 80.5 (Feb. 2021), pp. 8091–8126. ISSN: 1573-7721. DOI: 10.1007/s11042-020-10139-6. URL: https://doi.org/10.1007/s11042-020-10139-6.

[58] D.G. Mayer et al. "Survival of the fittest—genetic algorithms versus evolution strategies in the optimization of systems models". In: *Agricultural Systems* 60.2 (1999), pp. 113–122. ISSN: 0308-521X. DOI: https://doi.org/10.1016/S0308-521X(99)00022-0. URL: https://www.sciencedirect.com/science/article/pii/S0308521X99000220.

[59] Mohamed Amine Kaaouache and Sadok Bouamama. "Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud". In: *Procedia Computer Science* 60 (2015). Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings, pp. 1061–1069. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.08.151. URL: https://www.sciencedirect.com/science/article/pii/S1877050915022784.

[60] Mustafa Can Çavdar, Ibrahim Korpeoglu, and Özgür Ulusoy. "A Utilization Based Genetic Algorithm for virtual machine placement in cloud systems". In: *Computer Communications* 214 (2024), pp. 136–148. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2023.11.028. URL: https://www.sciencedirect.com/science/article/pii/S0140366423004267.

[61] Binbin Zhang, Xiao Wang, and Hao Wang. "Virtual machine placement strategy using cluster-based genetic algorithm". In: *Neurocomputing* 428 (2021), pp. 310–316. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.06.120. URL: https://www.sciencedirect.com/science/article/pii/S0925231220312005.

[62] Shahram Jamali and Sepideh Malektaji. "Improving grouping genetic algorithm for virtual machine placement in cloud data centers". In: *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*. 2014, pp. 328–333. DOI: 10.1109/ICCKE.2014.6993461.

[63] Zhe Ding, Yu-Chu Tian, and Maolin Tang. “Efficient Fitness Function Computation of Genetic Algorithm in Virtual Machine Placement for Greener Data Centers”. In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. 2018, pp. 181–186. DOI: 10.1109/INDIN.2018.8472063.

[64] Yefu Wang and Xiaorui Wang. “Performance-controlled server consolidation for virtualized data centers with multi-tier applications”. In: *Sustainable Computing: Informatics and Systems* 4.1 (2014), pp. 52–65.

[65] Vinicius Petrucci, Orlando Loques, and Daniel Mossé. “A dynamic optimization model for power and performance management of virtualized clusters”. In: *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. 2010, pp. 225–233.

[66] Patricia Arroba et al. “Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy-efficient cloud data centers”. In: *Concurrency and Computation: Practice and Experience* 29.10 (2017). e4067 cpe.4067, e4067. DOI: https://doi.org/10.1002/cpe.4067. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4067. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4067.

[67] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. “Energy and Power Efficiency”. In: *The Datacenter as a Computer: Designing Warehouse-Scale Machines*. Cham: Springer International Publishing, 2019, pp. 99–127. ISBN: 978-3-031-01761-2. DOI: 10.1007/978-3-031-01761-2_5. URL: https://doi.org/10.1007/978-3-031-01761-2_5.

[68] Rodrigo N. Calheiros et al. “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”. In: *Software: Practice and Experience* 41.1 (2011), pp. 23–50. DOI: https://doi.org/10.1002/spe.995. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.995. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.995.

[69] Fairouz Fakhfakh, Hatem Hadj Kacem, and Ahmed Hadj Kacem. “Simulation tools for cloud computing: A survey and comparative study”. In: *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*. 2017, pp. 221–226. DOI: 10.1109/ICIS.2017.7959997.

[70] Robert Schöne, Daniel Molka, and Michael Werner. “Wake-up latencies for processor idle states on current x86 processors”. In: *Computer Science - Research and Development* 30 (May 2014). DOI: 10.1007/s00450-014-0270-z.

[71] Jason Mars, Lingjia Tang, and Robert Hundt. “Heterogeneity in “Homogeneous” Warehouse-Scale Computers: A Performance Opportunity”. In: *IEEE Computer Architecture Letters* 10.2 (2011), pp. 29–32. DOI: 10.1109/L-CA.2011.14.

[72] URL: https://www.intel.com/content/www/us/en/products/sku/91317/intel-xeon-processor-e52699-v4-55m-cache-2-20-ghz/specifications.html.

[73] URL: https://aws.amazon.com/ec2/instance-types/.

[74] URL: https://www.milieucentraal.nl/energie-besparen/inzicht-in-je-energierekening/gemiddeld-energieverbruik/.

Cover Image: Tree. *3D illustration of supercomputer server racks in a data center background, computer server, server, database server background image and wallpaper for free download*. URL: https://pngtree.com/freebackground/3d-illustration-of-supercomputer-server-racks-in-a-data-center_13295989.html?sol=downref&id=bef

# A

# Pseudocode

**Algorithm 2** Particle Swarm Optimization algorithm

**Input**: $\mathcal{V}_{migrate}$, $\mathcal{H}_{available}$
**Output**: **Map**(host,vm)

Initialize parameters: $N_t, w^{max}, w^{min}, c_1, c_2$
$X^{max} = |\mathcal{H}_{available}|, X^{min} = 0, S^{max} = |\mathcal{H}_{available}|, S^{min} = -|\mathcal{H}_{available}|$
$gBestFitness =$ ***MAX VALUE***
**for** $p \in \mathcal{P}$ **do**
    $pBestFitness =$ ***MAX VALUE***
    $vm = 0$
    **while** $vm < |\mathcal{V}_{migrate}|$ **do**
        $X^p_{0,vm} =$ randomInt$(X^{min}, X^{max})$
        $S^p_{0,vm} =$ randomInt$(S^{min}, S^{max})$
        $vm = vm + 1$
$t = 0$
**while** $t < N_t$ **do**
    **for** $p \in \mathcal{P}$ **do**
        **while** $vm < |\mathcal{V}_{migrate}|$ **do**
            **if** $X^p_{t,vm} > X^{max}$ **then**
                $X^p_{t,vm} = X^{max}$
            **else if** $X^p_{t,vm} < X^{min}$ **then**
                $X^p_{t,vm} = X^{min}$
            $vm = vm + 1$
        Calculate $f$ ▷ based on Equation 2.8
    **for** $p \in \mathcal{P}$ **do**
        **if** $f < pBestFitness$ **then**
            $pBestFitness = f$
            $vm = 0$
            **while** $vm < |\mathcal{V}_{migrate}|$ **do**
                $pBestX^p_{vm} = X^p_{t,vm}$
                $vm = vm + 1$
        **if** $f < gBestFitness$ **then**
            $gBestFitness = f$
            $vm = 0$
            **while** $vm < |\mathcal{V}_{migrate}|$ **do**
                $gBestX_{vm} = X^p_{t,vm}$
                $vm = vm + 1$
    **for** $p \in \mathcal{P}$ **do**
        determine $w_t$ ▷ based on Equation 2.10
        $vm = 0$
        **while** $vm < |\mathcal{V}_{migrate}|$ **do**
            **determine** $S^p_{t+1,vm}$ ▷ based on Equation 2.9
            $X^p_{t+1,vm} = X^p_{t,vm} + S^p_{t+1,vm}$
    $t = t + 1$
**return Map**$(\mathcal{V}_{migrate}, gBestX)$

**Algorithm 3** Genetic Algorithm

**Input**: $\mathcal{V}_{migrate}$, $\mathcal{H}_{available}$
**Output**: **Map**(host,vm)

Initialize parameters: $N_t, w^{max}, w^{min}$
**for** $p \in \mathcal{P}$ **do**
  $vm = 0$
  **while** $vm < |\mathcal{V}_{migrate}|$ **do**
    $p_{vm} =$ random$(0, |\mathcal{H}_{available}|)$
$t = 0$
**while** $t < N_t$ **do**
  determine $w_t$ ▷ based on Equation 2.13
  $numSurvival =$ round$(w_t|\mathcal{P}|)$
  $survivalParents \leftarrow$ select $numSurvival$ fittest parents
  $c = 0$
  **while** $c < |\mathcal{P}| - numSurvival$ **do**
    $parent1 \leftarrow$ **selectParent** ▷ based on Equation 2.12
    $parent2 \leftarrow$ **selectParent** ▷ based on Equation 2.12
    $child \leftarrow$ **crossover**$(parent1, parent2)$
    $child \leftarrow$ **mutate**$(child)$
    $children.add(child)$
  $population.clear()$
  $population.add(survivalParents)$
  $population.add(children)$
$bestFitness =$ ***MAX VALUE***
**for** $p \in \mathcal{P}$ **do**
  Calculate $f$ ▷ based on equation 2.8
  **if** $f < bestFitness$ **then**
    $bestFitness = f$
    $bestParent = p$
**return Map**$(\mathcal{V}_{migrate}, bestParent)$

**procedure** selectParent($\mathcal{P}$)
  $fitnessSum = 0$
  **for** $p \in \mathcal{P}$ **do**
    Calculate $f$ ▷ based on equation 2.8
    $fitnessSum = fitnessSum + \frac{1}{f}$
  $r =$ random$(0, 1) \cdot fitnessSum$
  $tempFitness = 0$
  **for** $p \in \mathcal{P}$ **do**
    Calculate $f$ ▷ based on equation 2.8
    $tempFitness = tempFitness + \frac{1}{f}$
    **if** $tempFitness > r$ **then return** $p$

**procedure** crossover(parent1,parent2)
  $crossoverPoint =$ randomInt$(0, |\mathcal{V}_{migrate}|)$
  $vm = 0$
  **while** $vm < crossoverPoint$ **do**
    $c_{vm} = parent1_{vm}$
    $vm = vm + 1$
  **while** $vm < |\mathcal{V}_{migrate}|$ **do**
    $c_{vm} = parent2_{vm}$
    $vm = vm + 1$
**return** $c$

**procedure** mutate(child)
  $vm = 0$
  **while** $vm < |\mathcal{V}_{migrate}|$ **do**
    $r =$ random$(0, 1)$
    **if** $r < 0.01$ **then**
      $child_{vm} =$ randomInt$(0, |\mathcal{H}_{available}|$
**return** $child$

# B

# Confidence intervals

## B.1. Spread of results

This image shows the 95% spread (2 standard deviations on both sides) of all simulation results when simulating 100 times with a randomly generated workload for each simulation. Due to time complexity, it is not calculated for the ILP algorithm.

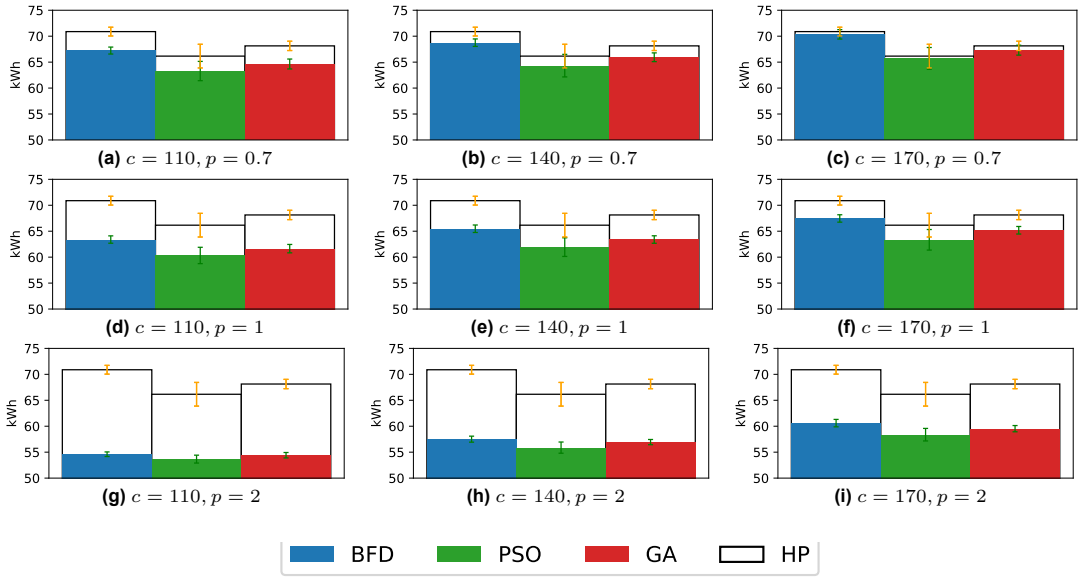


**Figure B.1:** Spread of energy consumption results experiment 1 with randomly generated workloads for 100 simulations

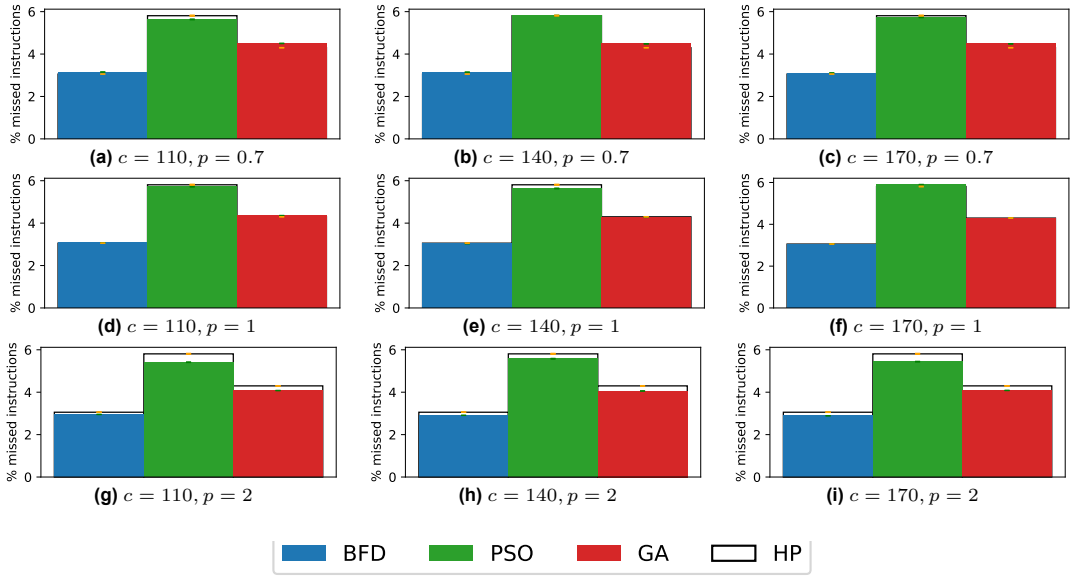


**Figure B.2:** Spread of performance results experiment 1 with randomly generated workloads for 100 simulations

## B.2. Confidence interval on average

The figures below show the 95% confidence interval based on the margin of error. The calculations for this are shown after the figures.

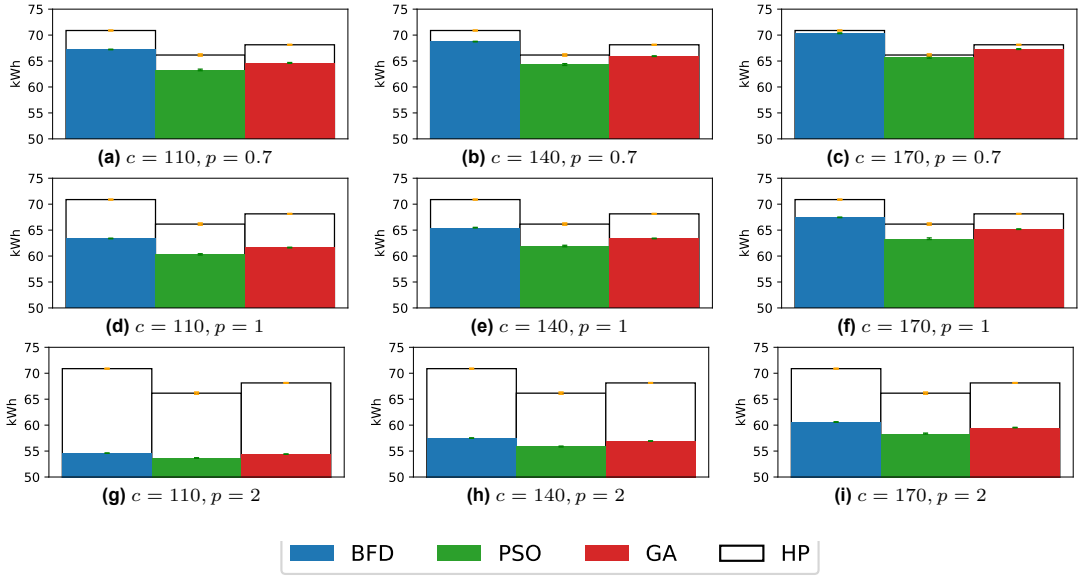


**Figure B.3:** Confidence on average of energy consumption results experiment 1 with randomly generated workloads for 100 simulations

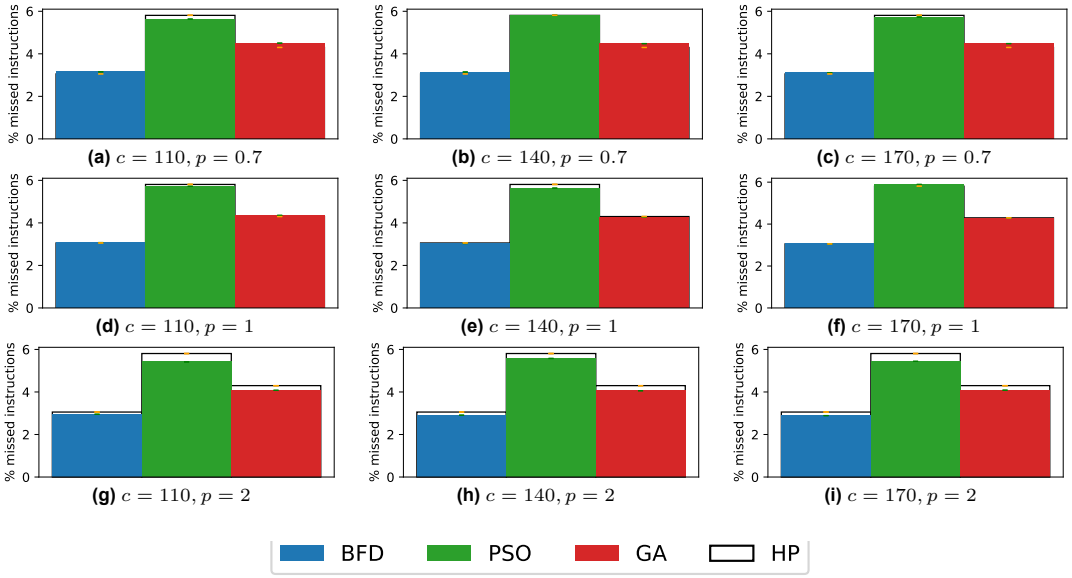


**Figure B.4:** Confidence on average of results on the performance metric of experiment 1 with randomly generated workloads for 100 simulations

The margin of error is based on the following equation:

$$MoE = z(0.95) \cdot \sqrt{\frac{\sigma^2}{n}} =\approx 1.64 \cdot \sqrt{\frac{\sigma^2}{n}} \tag{B.1}$$

Here $MoE$ is the margin of error, $\sigma^2$ is the variance of the results and $n$ is the number of simulations, which is 100 in this case.

# C

# Trace of Active Hosts

Note that for the particle swarm optimisation (PSO) algorithm and the genetic algorithm (GA) the average of all 10 simulations is taken.

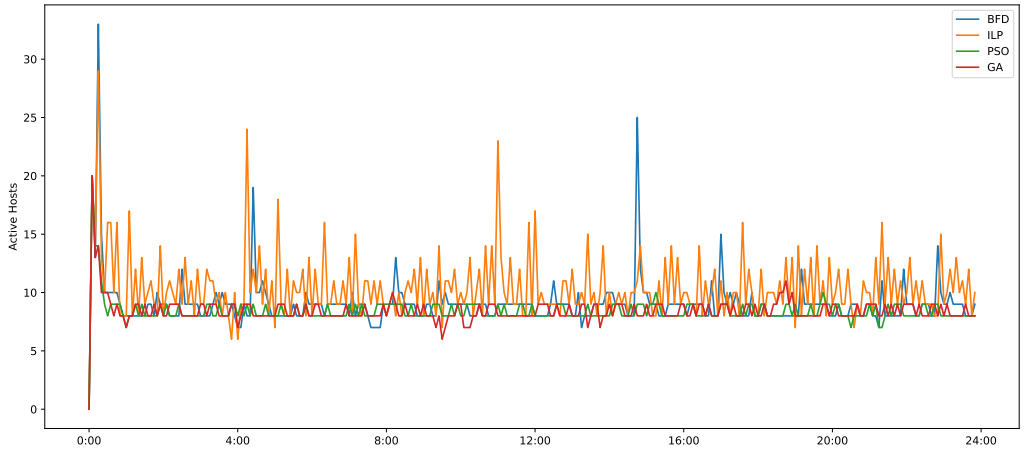


**Figure C.1:** Number of active hosts during simulation ($c = 110, p = 0.7$)

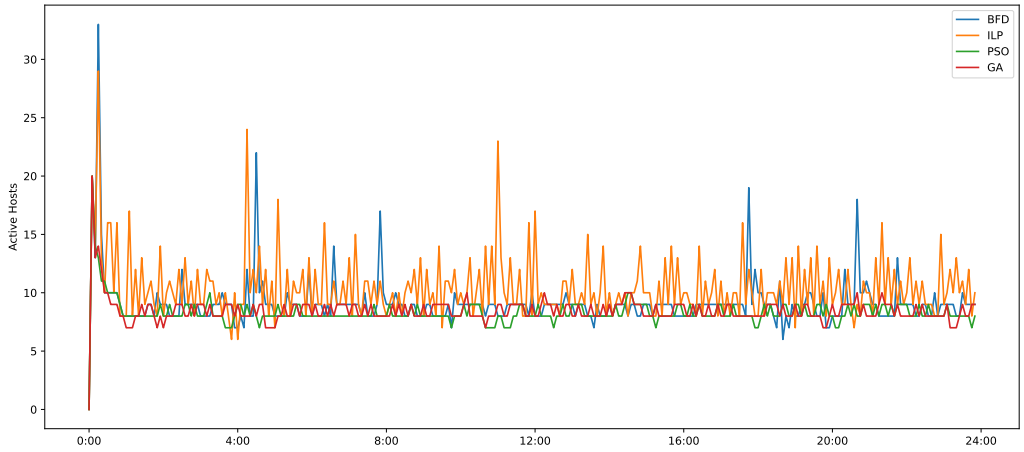


**Figure C.2:** Number of active hosts during simulation ($c = 140, p = 0.7$)

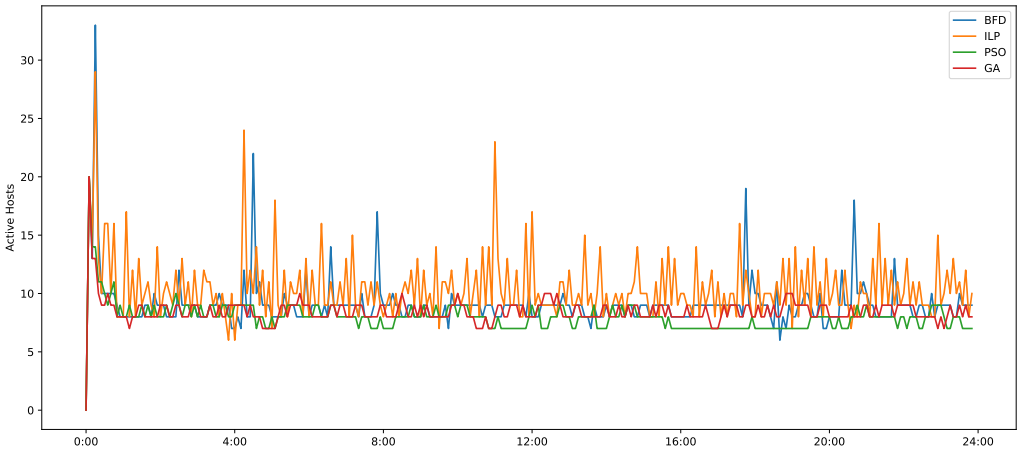


**Figure C.3:** Number of active hosts during simulation ($c = 170, p = 0.7$)


**Figure C.4:** Number of active hosts during simulation ($c = 110, p = 1$)

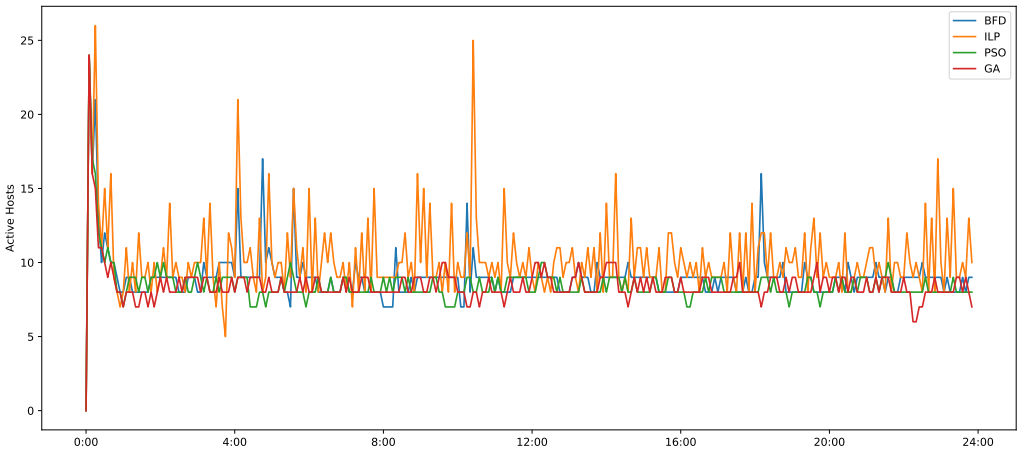


**Figure C.5:** Number of active hosts during simulation ($c = 140, p = 1$)

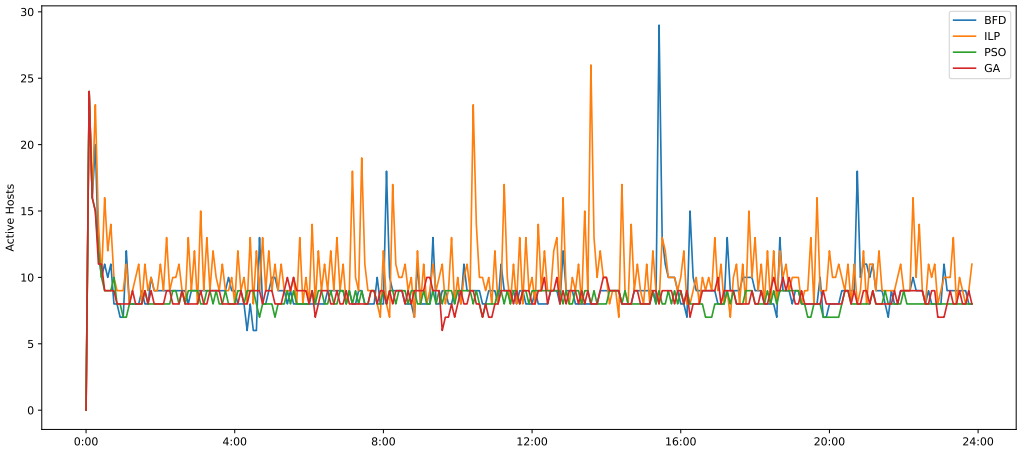


**Figure C.6:** Number of active hosts during simulation ($c = 170, p = 1$)

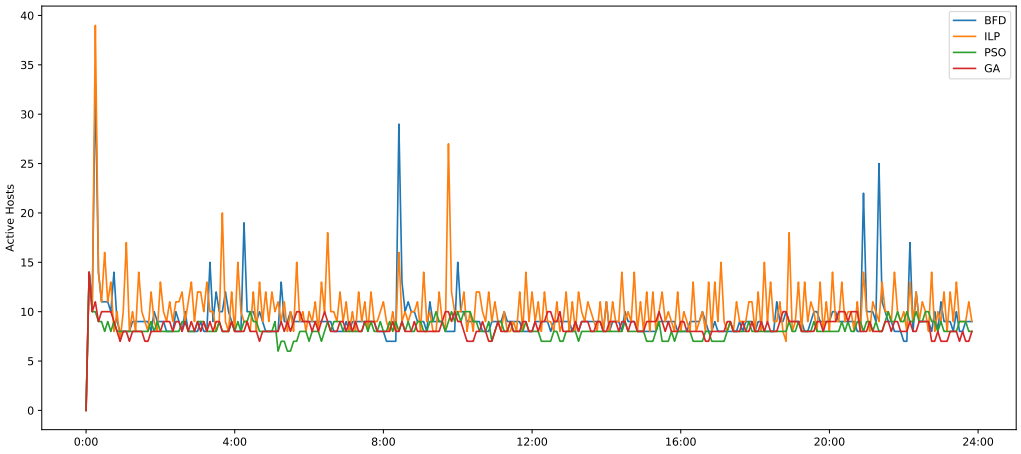


**Figure C.7:** Number of active hosts during simulation ($c = 110, p = 2$)

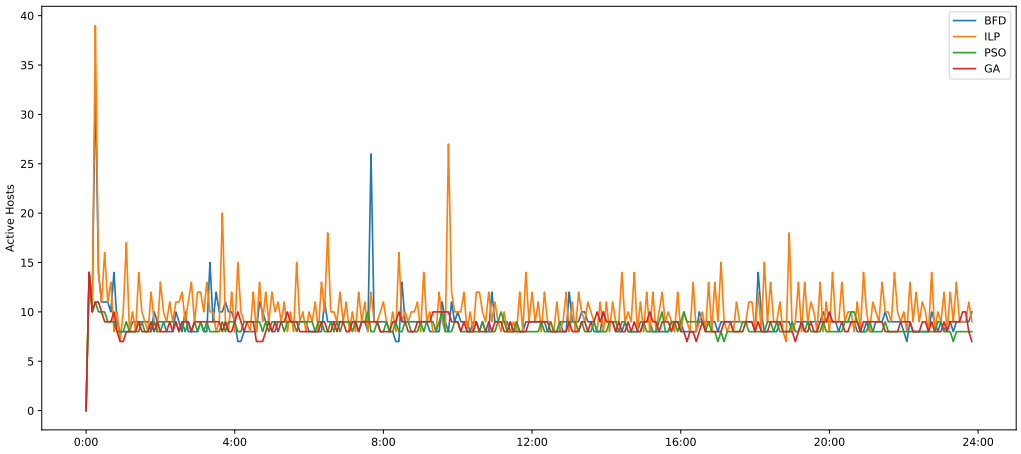


**Figure C.8:** Number of active hosts during simulation ($c = 140, p = 2$)

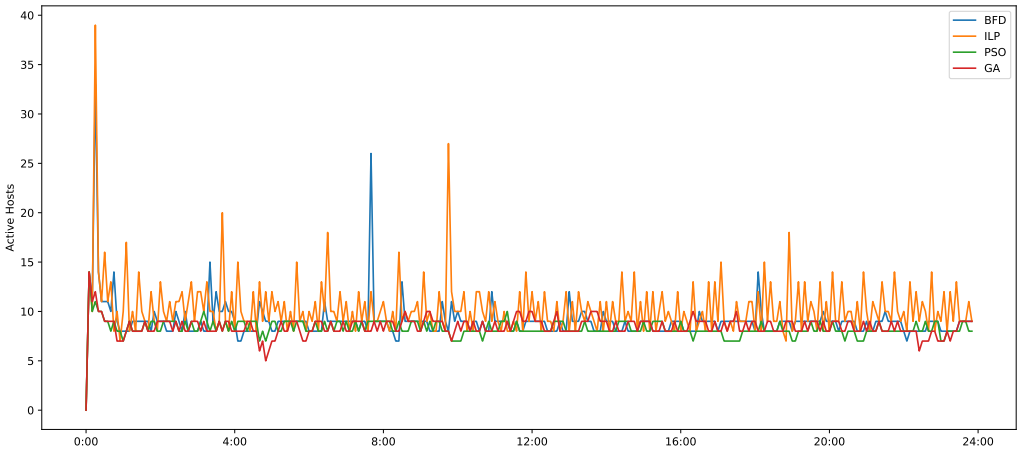


**Figure C.9:** Number of active hosts during simulation ($c = 170, p = 2$)

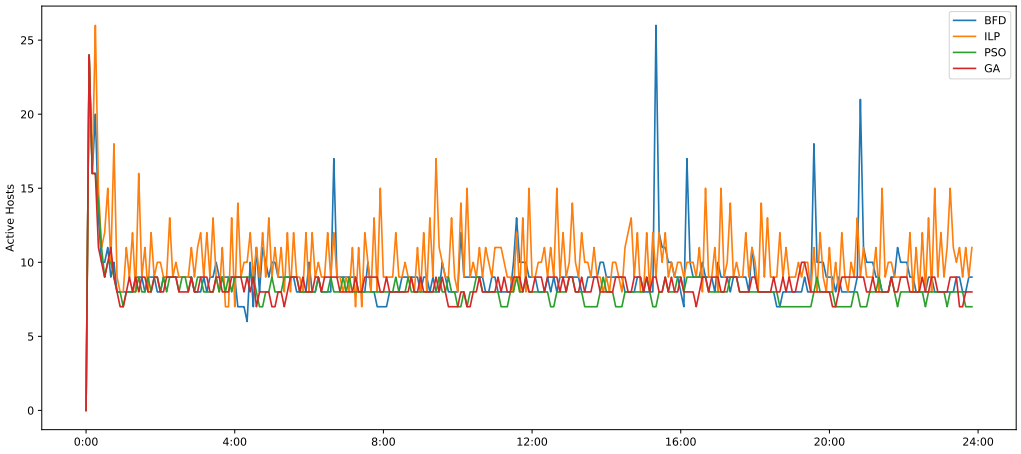


**Figure C.10:** Number of active hosts during simulation ($c = 220, p = 1$ ,high-performance)