UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AN INTELLIGENT, DATA-CENTRIC SCHEME FOR ALLOCATING QUERIES TO QUERY PROCESSORS

A dissertation submitted to the University of Thessaly in accordance with the requirements of the DIPLOMA degree

Authors: Sula Madalena, Karanika Anna

Supervisor: Stamoulis Georgios, Professor

2nd Committee Member: Evmorfopoulos Nestor, Assistant Professor

3rd Committee Member: Bargiotas Dimitrios, Associate Professor

Scientific Consultant: Kolomvatsos Konstantinos, MSc Instructor

June 2019

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AN INTELLIGENT, DATA-CENTRIC SCHEME FOR ALLOCATING QUERIES TO QUERY PROCESSORS

A dissertation submitted to the University of Thessaly in accordance with the requirements of the DIPLOMA degree

Authors: Sula Madalena, Karanika Anna

Supervisor: Stamoulis Georgios, Professor

2nd Committee Member: Evmorfopoulos Nestor, Assistant Professor

3rd Committee Member: Bargiotas Dimitrios, Associate Professor

Scientific Consultant: Kolomvatsos Konstantinos, MSc Instructor

June 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΥΦΥΕΣ ΣΥΣΤΗΜΑ ΑΝΑΘΕΣΗΣ ΕΡΩΤΗΜΑΤΩΝ ΣΕ ΕΠΕΞΕΡΓΑΣΤΕΣ ΕΡΩΤΗΣΕΩΝ, ΠΡΟΣΑΝΑΤΟΛΙΣΜΕΝΟ ΣΤΑ ΔΕΔΟΜΕΝΑ

Διπλωματική Εργασία

Συγγραφείς: Σούλα Μανταλένα, Καρανίκα Άννα

Επιβλέπων Καθηγητής: Σταμούλης Γεώργιος, Καθηγητής

2ο Μέλος Επιτροπής: Ευμορφόπουλος Νέστωρ, Επίκουρος Καθηγητής

3ο Μέλος Επιτροπής: Μπαργιώτας Δημήτριος, Αναπληρωτής Καθηγητής

Επιστημονικός Σύμβουλος: Κολομβάτσος Κωνσταντίνος, Διδάσκων Μεταπτυχιακού

Ιούνιος 2019

ACKNOWLEDGMENTS

Firstly, I would like to thank my Professor George Stamoulis who was always available when I needed his opinion on my studies. No matter how hard this journey was, he supported my plans and helped me out whenever I was in need. Secondly, I would like to appreciate Dr. Kostas Kolomvatsos who supervised us throughout this research. His support and his technical advices were valuable in order to accomplish this thesis.

Furthermore, I would like to express my appreciation to all my friends and especially to my best friends, Panagiotis and Giannis, who have always been there for me. I will keep our past experiences in my heart and I am looking forward for new ones.

To my parents, thank you for showing that continuous support in every possible way. I would like to thank my mother for being that role model I look up to and for making me a fighter in life. To my siblings, Vasiliki and Michael, who keep encouraging me no matter if I succeed or fail. Also, I would like to thank Giorgos for his endless support and love.

Lastly, Anna, I appreciate that you were my partner in this. It is hard to find someone with the same values and aspirations not only in the working environment but also in life. I am grateful our ways came across and we figured out that we can be such good friends and an amazing team.

Sula Madalena

First and foremost, I would like to express my profound appreciation to Dr. Kostas Kolomvatsos for his constant support and guidance throughout the research and writing of this thesis. Furthermore, I would like to thank for this accomplishment Professor Georgios Stamoulis who supervised us during the past months and has always helped us in any way he can.

Moreover, I would like to express my heartfelt gratitude to my family who have always been by my side, supporting my decisions. To my closest friends, thank you for being there whenever I have needed you; these pasts years have been the greatest and I will remember them with fondness. Last but not least, I want to thank my partner for his never-ending support and love.

Finally, Madalena, without you this thesis would not have been possible. Our friendship and shared tenacity helped us make it through the worst of it and I will always be grateful to you for that. Thank you for everything.

Karanika Anna

“The **IMPOSSIBLE** is what people think you should accomplish.

What's **POSSIBLE** is what you dream of doing yourself.”

ABSTRACT

Internet of Things applications play a significant role in daily life by providing simple services such as predicting the weather or, even, offering elegant Smart City functionalities. The huge amounts of data produced by these applications is the reason why Big Data Analytics has been a popular topic in the scientific forefront in recent years. Researchers have been trying to overcome the challenge of large-scale data management in order to achieve real-time responsiveness in systems. This can be accomplished by the queries' distribution to several nodes, aiming for the parallelization of query processing. In this thesis, we propose an intelligent mechanism, adopted by the Query Controller module, which is responsible for the orchestration of the queries' allocation to the appropriate nodes, taking into consideration the queries' and nodes' characteristics. Our scheme is built on top of each of three allocation algorithms: the Hungarian Method, the Clustering Task Allocation algorithm and the Simplified Swarm Optimization for Task Allocation algorithm. Based on our experimental results, we evaluate and compare the algorithms' performance, and propose the best algorithm to be used in a number of different cases.

ΠΕΡΙΛΗΨΗ

Οι εφαρμογές του Διαδικτύου των Πραγμάτων (Internet of Things) διαδραματίζουν σημαντικό ρόλο στην καθημερινή ζωή παρέχοντας απλές υπηρεσίες, όπως η πρόβλεψη των καιρικών συνθηκών ή ακόμα και η προσφορά κομψών λειτουργιών της Smart City. Τα τεράστια ποσά των δεδομένων που παράγονται από αυτές τις εφαρμογές είναι ο λόγος για τον οποίο το Big Data Analytics είναι ένα δημοφιλές θέμα στο επιστημονικό προσκήνιο τα τελευταία χρόνια. Οι ερευνητές προσπαθούν να ξεπεράσουν την πρόκληση της διαχείρισης δεδομένων μεγάλης κλίμακας προκειμένου να επιτευχθεί η απόκριση σε πραγματικό χρόνο από τα συστήματα. Αυτό μπορεί να πραγματοποιηθεί με την κατανομή των ερωτημάτων σε διάφορους κόμβους, με στόχο την παραλληλοποίηση της επεξεργασίας των ερωτημάτων. Σε αυτή τη διπλωματική εργασία, προτείνουμε έναν ευφυή μηχανισμό, ο οποίος υιοθετείται από την οντότητα Query Controller, η οποία είναι υπεύθυνη για την ενορχήστρωση της κατανομής των ερωτημάτων στους κατάλληλους κόμβους, λαμβάνοντας υπόψη τα χαρακτηριστικά των ερωτημάτων και των κόμβων. Το σχήμα μας είναι χτισμένο πάνω από κάθε έναν από τους τρεις αλγορίθμους κατανομής: την Ουγγρική Μέθοδο, τον αλγόριθμο Clustering Task Allocation και τον αλγόριθμο Simplified Swarm Optimization for Task Allocation. Με βάση τα πειραματικά μας αποτελέσματα, αξιολογούμε και συγκρίνουμε την απόδοση των αλγορίθμων και προτείνουμε τον καλύτερο αλγόριθμο που θα χρησιμοποιηθεί σε διαφορετικές περιπτώσεις.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF TABLES

1 INTRODUCTION

Big Data Analytics has been a topic of interest for a significant number of researchers during the last few years. The automated data-centric decision-making is the long-term objective which has brought about this shift. Handling large-scale data efficiently constitutes the main challenge to defeat in order to build intelligent applications in the future. At present, devices and applications generate enormous quantities of data whose efficient processing is required to serve high quality applications.

The Internet of Things (IoT) concept constantly influences daily life since it has gained recognition with the breakthrough of advanced technology. Nowadays, its architecture lies in most of the applications as it enables the connection and interaction between different devices and applications via their integration. IoT technologies have developed a path so as to go beyond the smartphones, wearables etc. and lead to a fully automated world by evolving into connecting everyday objects and embedding intelligence into the environment.

The rapid development of IoT technologies transformed the computing model into a similar process flow to that of the traditional commodities delivery such as water and electricity, since users access information services without regard to where these services are hosted. The infrastructure of Cloud Computing enables businesses and users to access applications and information from anywhere in the world, as cloud consists of data centers where data is transferred, processed and maintained.

The proliferation of IoT and Cloud applications and the huge amounts of data generated were the reasons that boosted the need of instant data processing. This led to the emergence of Edge computing systems which oppose the high processing power and latency that Cloud offers, since the computations are performed locally, at the edge of the network. Edge Computing, in contrast to Cloud Computing, is considered a revolution in immediate processing since data is being produced at an increasing rate and it is not efficient to convey them to the Cloud for processing purposes and then communicate the results back to the user.

The data generated by the IoT infrastructure —sensors and edge nodes— are passed on to the cloud in order to be analyzed. After their collection and through the process of organization into various databases and their tables, as well as cleaning from

duplicate entries, missing values, etc. the data are ready to be examined by various algorithms. This management can lead to the extraction of new knowledge and fresh insights on several cases and issues of daily life, working environments, healthcare, personal, business and community security, even marketing and social media. Such information can be highly valuable to all, including individuals, companies and the state. Analytics' uses can range from applications that predict the weather to elegant Smart Cities that work seamlessly by making life easier for the citizens.

Smart Cities can be very convenient for their citizens, offering applications that manage real-time situations. Smart Public Services is a useful application that can make administrative paperwork, red tape and long waiting queues go extinct. Moreover, security can be reinforced by carefully implemented security programs. Additionally, traffic control, automation and efficiency can be achieved with the assistance of analytics. Likewise, education and healthcare can benefit greatly from the sharing of information, such as historical archives, patient records etc. From all the above, it has become clear that the data can be very big in volume and, thus, the requests for access to them almost countless.

Large-scale data can be managed either with batch-oriented or stream (online)-oriented processing, according to whether incoming queries are processed in large pieces or not. Smart Cities are in need of a processing scheme that can serve the incoming queries directly and qualitatively, which can be accomplished with stream processing. In contrast, batch processing methods do not conform to the requirements of a Smart City application but can be ideal in terms of handling virtually static data. The huge amounts of generated data can hardly be processed in time despite the use of real-time processing, bringing on a system overload. Thus, the requests should be distributed to a number of servers instead of ending up on a single one. This means that the available data should also be portioned and duplicated into smaller sections —partitions— hosted in different nodes. Partitioning is a method adopted to increase the processing speed through query distribution. Hence, it rises the challenge of allocating each query to the node which will serve it in the fastest way while also offering the relevant information.

In this research, we propose an intelligent mechanism that handles the process of query-node allocation. We envision an entity responsible to manage the incoming queries, i.e., the Query Controller (QC). Queries are defined by end users or applications to the QC;

thus, the QC should efficiently respond in the minimum possible time. We have to notice that multiple QCs can be present to the Cloud having direct interaction with nodes where data are stored. In front of each node, we envision another entity, i.e., the Query Processor (QP) that is responsible to receive and execute every query reported by a QC. The QC is the component that is responsible for the orchestration of the incoming queries' allocation to the appropriate nodes/QPs. As an allocation, we define the optimal selection of a node for a distinct query determined by the query's characteristics and the nodes' current performance. Each node/QP can access its corresponding data partition while executing its allocated queries. Such a decision-making process is dynamically influenced by the continuously updated data and nodes' performance. Due to the fact that we are trying to allocate queries, i.e., tasks to the appropriate nodes/QPs, i.e., workers, the problem could be treated as (i) an Assignment Problem [1] where we have to allocate a number of tasks (i.e., queries) to a number of workers (i.e., nodes/QPs), (ii) as a problem where we have to rely on machine learning principles and learn the best possible allocation or (iii) an optimization problem where we have to find the optimal solution under a number of constraints. Actually, in this thesis, we solve the problem proposing specific algorithms for both cases. We propose a cost function for delivering the cost of an allocation decision for pairs of queries – nodes/QPs and implement the solution of our problem adopting: (i) the Hungarian Method (HM); (ii) a machine learning model, i.e., clustering (i.e., the Clustering Task Allocation algorithm (CTA); and, (iii) the Simplified Swarm Optimization (SSO) Task Allocation algorithm (SSO-TA). Our models are realized in a distributed computing scenario where every QC is responsible to apply the proposed schemes and decide the final allocation.

The rest of this thesis is organized as follows. Chapter 2 provides the description of the Internet of Things (IoT), Edge and Cloud Computing technologies. Chapter 3 explores the concepts of Analytics, Query Management and Query Allocation. Chapter 4 presents our approaches for allocating queries to query processors, handling these approaches as solutions to the Assignment Problem. Chapter 5 demonstrates the evaluation of our approaches' performance according to a variety of experiments. Finally, Chapter 6 concludes this thesis by presenting our primary findings and discussing our ambitions for future work.

2 INTERNET OF THINGS, EDGE COMPUTING AND CLOUD

2.1 Internet of Things

The Internet of Things (IoT) refers to a system of Internet-connected computing platforms and personal devices embedded with electronics and other forms of hardware such as sensors [2]. The hardware is able to transmit and receive data as well as to be controlled remotely [3]. IoT applications have revolutionized humankind's daily life, bringing on productivity advancement, economic profits and quality upgrade to work activities [4], [5], [6].

In 1982, a modified Coke vending machine at Carnegie Mellon University constitutes the first IoT appliance; it provided information regarding its inventory and whether newly loaded drinks were cold or not [7]. Mark Weiser's 1991 paper on ubiquitous computing, "The Computer of the 21st Century", as well as Kary Främling and his team's work at Helsinki University of Technology published in 2002 shaped today's vision of the IoT [8]. Cisco Systems has estimated that the IoT conception was "born" between 2008 and 2009, with the things/people ratio growing from 0.08 in 2003 to 1.84 in 2010, which denotes the exponential growth of IoT devices [9].

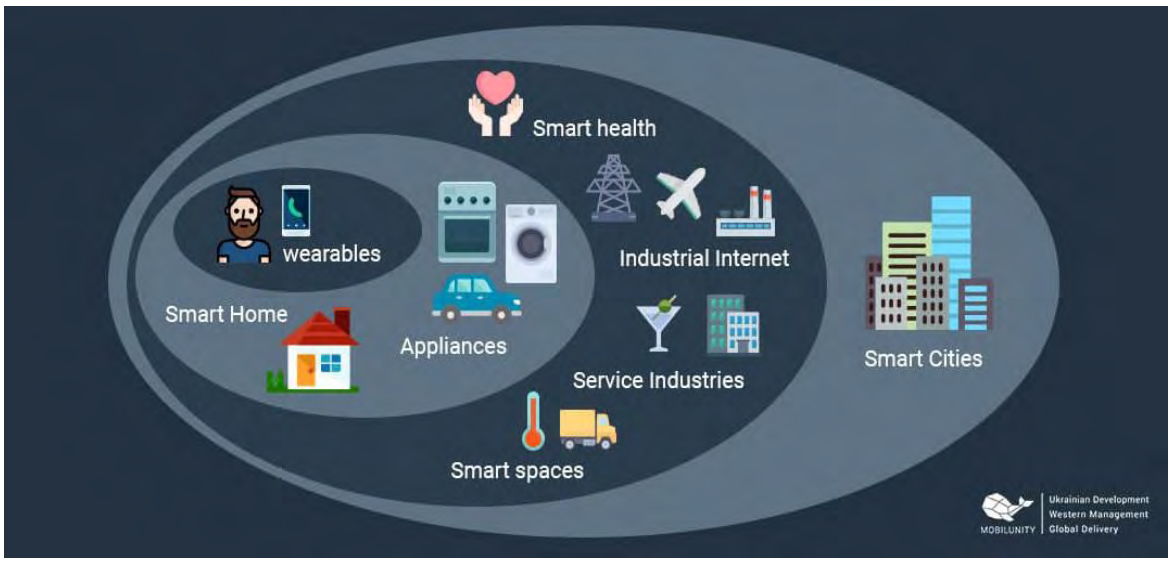


Figure 2.1 Internet of Things Applications [10]

2.1.1 The Architecture of the Internet of Things

The architecture of the IoT is the aggregation of various elements such as sensors, protocols, actuators, cloud services, and layers. The layers of IoT are (i) the client side (IoT

Device Layer), (ii) the operators on the server side (IoT Gateway Layer) and (iii) a pathway for connecting clients and operators (IoT Platform Layer).

The IoT Architecture Diagram[1] consists of the following:

1. **Sensors and actuators**

   The most important feature of a sensor is its ability to transform the gathered information into the appropriate format for data analysis. Similarly, the actuators are able to obtain further information for analysis purposes. These devices can intervene in the physical world by making intelligent decisions such as adjusting the temperature in the room.

2. **Gateways and Data Acquisition Systems**

   The information received from sensors and actuators is converted from analog to digital and then aggregated. Gateways and data acquisition systems (DAS) are located close to the source of data in order to minimize the transmission latency.

3. **Edge Computing**

   Edge nodes are placed in close proximity to sensors and actuators. Their systems are responsible for pre-processing activities and preliminary analytics.

4. **Data center and cloud**

   Data center/Cloud is where the key procedures take place. The resulting information from the thorough and comprehensive analysis is delivered back to the physical world sometimes expecting feedback from the users. It should be noted that the returned data must meet certain quality standards and requirements.

5. **User's control**

   Optionally, only a partial control over the IoT system can be provided to the user. The users are able to monitor and manage any situation they have the rights to do. More specifically, users can perform tasks such as visualization and management.

[1]https://www.coursera.org/lecture/iot-wireless-cloud-computing/2-1-iot-architecture-part-1-K6pdR

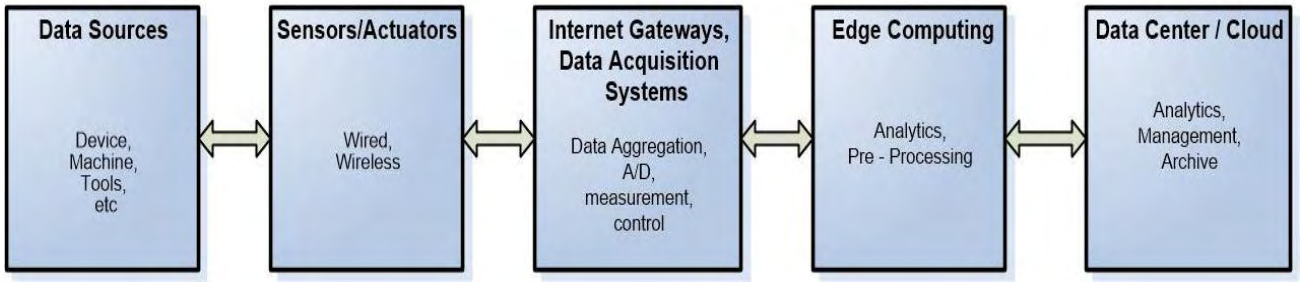


Figure 2.2 Internet of Things Architecture

2.1.2 Internet of Things Applications

The plethora of applications developed for IoT devices can be categorized into consumer, commercial, industrial and infrastructure fields.

Consumer IoT Applications include automated vehicles, smart home devices, wearables [11] and remotely accessible monitoring appliances [12]. Smart home devices provide home automation such as lighting, heating, security systems etc. which can be fully commanded by users. Such devices include Amazon Echo and Google Home [13], [14]. Wearables consist of sensors and several other tools which enable recognition and recording of patterns as long as there is physical contact between the wearable and user [15].

Commercial applications entail IoT for medical and health objectives, research purposes and monitoring activities as well as (autonomous) transportation systems. Remote health monitoring and emergency notification systems such as blood pressure monitors or even devices capable of monitoring implants are based on IoT technology [16]. Moreover, large amounts of patient data emphasized the need of IoT applications which collect and combine data to better diagnose patient health status [17]. Vehicles, infrastructure, drivers and every component of transportation systems can interact through an IoT application. In this way accidents are diminishing and smart traffic control as well as safety throughout the city are ensured [18], [19].

Industrial applications involve a network of sensors, instruments, and other devices connected to computers' applications used for production and energy efficiency management allowing for data gathering and analysis. It is claimed that Industrial applications boost productivity, service delivery and efficiency via the technology revolution of the IoT technologies they use. The Industrial Internet of Things vision of the world is one where smart connected assets (the things) operate as part of a larger system

or system of systems that make up the smart manufacturing enterprise. The “things” possess varying levels of intelligent functionality, ranging from simple sensing and actuating, to control, optimization and full autonomous operation [20]. An intelligent IoT system which can be embedded in band saw machines developed to monitor and notify regarding the degradation rate of the band saw belt and need for replacement [21].

Infrastructure applications include monitoring and controlling processes of road, railway, bridge and tunnel networks as well as power generation and consumption. Songdo, South Korea, is being built to be the first fully functioning Smart City. By June 2018, 70 percent of its business district was designed and completed so that it will operate mostly in an automated way[2]. Furthermore, energy usage is controlled by the smart grid —an electrical power grid which utilizes automated control, high-power converters, modern communications infrastructure, sensing and metering technologies— which enables systems making intelligent decisions about energy generation and distribution [22].

2.2 Cloud Computing

Cloud computing can be described as a service of rendering computer system resources, in particular data storage and computing power, without the need for the user’s frequent involvement. This service is available to many users and is realized by using data centers. At present, large clouds are very popular and can be accessed by users worldwide [23]. In the event that a user and the data center are in a close distance, the term edge node may be applied. The available clouds are either private to one organization, in which case they are called enterprise clouds, or used by many organizations and called public clouds. There is a third category, that combines the aforementioned; those clouds are called hybrid clouds.

As early as 1993, the term “cloud” could be used to refer to distributed computing frameworks. In addition, the first time that the term “cloud computing” was used can be traced back to 1996, when Compaq referenced it in an internal document, but it became popular only when Amazon launched its Elastic Compute Cloud product in 2006 [24], [25], [26].

[2] https://www.citylab.com/life/2018/06/sleepy-in-songdo-koreas-smartest-city/561374/

The use of cloud computing can speed up the setting up of applications, while also decreasing the need for continual maintenance or managing sessions and simplifying the process of changing the available resources in case additional or less of them are needed after the initial settings. That, along with the offer of non-expensive computers and networks of great bandwidth has skyrocketed the demand for cloud computing.

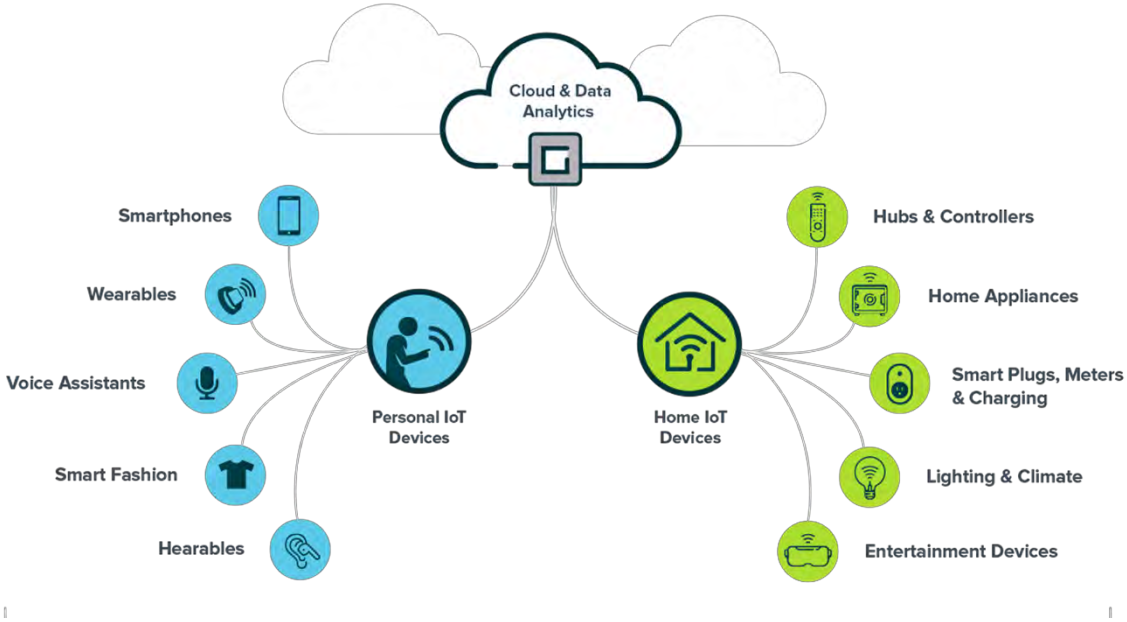


Figure 2.3 The usage of cloud computing [27]

2.2.1 Cloud Service Models

Cloud service providers equip the modern world with their services based on the three standard models per the National Institute of Standards and Technology (NIST) [28] as demonstrated below:

1. **Infrastructure as a Service (IaaS)**

   NIST defines IaaS as the service where the user is capable of deploying and running software which may consist of operating systems and applications running on top of a virtual machine[3]. The consumer totally controls the operating systems, storage and deployed applications and partially networking components (e.g. host firewalls). More specifically, these kinds of services provide high-level APIS[4] so that

[3] In computing, a virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination [134].

[4] In computer programming, an application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer

low-level network information such as physical computing resources and security details are dereferenced[5]. IaaS provides the opportunity of using cloud orchestration technology so that virtual machines are created and distributed to hypervisors which in turn take over running them amongst each other [29].

2. **Platform as a Service (PaaS)**

   NIST's definition of PaaS is the service that provides the cloud infrastructure on top of which custom applications are deployed using libraries and services offered by the cloud service provider. The user fully controls and manages the deployed applications but not the underlying infrastructure which consists of networks and the operating system. Hence, the technical project team is able to focus on the development, execution and management of applications without having to dedicate time in building and maintaining the infrastructure [30].

3. **Software as a Service (SaaS)**

   NIST defines SaaS as the service that offers a specific software which is accessible through a program interface or web browser. The consumer can only modify configuration settings whereas the infrastructure and applications are immutable. Other names for SaaS applications are Web-based software, on-demand software and hosted software. SaaS applications are used for business technologies such as customer relationship management and sales management [31].

2.2.2 Cloud Applications

Cloud computing applications are numerous, vary around us and have become of mainstream usage in daily routine for both business and personal life. Cloud computing is used from personal emails to software development, testing and deployment [32].

Cloud computing provides the opportunity to store information while the user interacts with applications, which run on several computer devices and mobile phones. In

program by providing all the building blocks, which are then put together by the programmer. An API may be for a web-based system, operating system, database system, computer hardware, or software library [135].

[5] In computing, the term "dereferencing" refers to manipulating and probably demonstrating a low-level information in a more understandable way (e.g. programming languages provide the opportunity to access a variable using a pointer) [136].

this way, user preferences are gathered and used in order to provide customized messages or customer support and recommend products to users. The so called chatbots or bots constitute cloud-based applications able to process text, audio and visual data. Siri and Alexa are two chatbots used at a great extent in everyday life and they seem to be very useful.

Currently, huge amounts of data are generated in a daily basis from household devices and mobile phones to industrial machines. There would have been impossible to store these data if it weren't for the cloud applications. Messaging and calling apps such as Viber and Skype are developed in a way that the information generated is stored in a cloud so that it can be accessible via internet [32]. Communication is vital for both users and enterprise environments, and people can easily leverage cloud computing in order to facilitate it and make it possible from every corner of the world.

Cloud services extend to enterprise solutions such as customer relationship management (CRM) and enterprise resource planning applications so that businesses are capable of designing and maintaining their processes in order to access their applications simply by using a web browser. Furthermore, cloud computing enables enterprise environments to back up critical data at a time that hacking events are numerous. Dropbox and Google Drive constitute common and useful "clouds" between users and the business world [33]. Moreover, cloud computing facilitates application development and testing since developers and engineers save their time by using platforms equipped with lots of pre-coded tools and libraries. Project time is also saved for facing critical issues as long as several tools simplify, based on the cloud service provider, the testing in order to launch the developed software.

2.2.3 Cloud Computing Characteristics

Cloud computing constitutes the model on which frameworks, platforms and several applications are built providing access via internet to huge pools of computing resources, as well as repositories of enormous amounts of data. The model of cloud computing exhibits five key characteristics according to the National Institute of Standards and Technology as demonstrated below [28]:

1. **On-demand self-service**

   Computing capabilities and resources (server time, network storage etc.) can be accessed and managed without additional interaction between the consumer and the service provider.

2. **Broad network access**

   Interconnected devices such as laptops and mobile phones can reach information, data and computing resources since cloud-based software enables such functionalities.

3. **Resource pooling**

   Cloud service providers provision their services in such way that multiple consumers are able to be provided, giving a sense of independence to the users since the latter have no knowledge regarding the exact location of resources. Consumers can only get into high-level information such as the country and state of the provided resources.

4. **Rapid elasticity**

   Cloud computing gives the sense of unlimited access to unlimited data and resources since cloud capabilities can scale outward and inward based on customer requirements in any case and time.

5. **Measured service**

   Resource usage is controlled (within transparency for both providers and consumers by cloud-based systems) in order for the usage to be both improved and reported in the proper way.

2.2.4 Cloud Architecture

The cloud architecture refers to the way the software components of the cloud computer services are integrated to produce the resulting service and is typically implemented connecting the different parts using mechanisms like message queues. Furthermore, cloud engineering is the systematic approach to affairs of designing, developing, running and maintaining cloud computing systems. It demands the knowledge of areas like systems, software, web, performance, information technology engineering, security, platform, risk, and quality engineering.

The desire for reducing the events of discontinuation of the computational processes, as well as for improving the data center network design drives the continuous cloud technology and development research.

2.3 Edge Computing

The term "Edge Computing" refers to any computer system/program which provides minimum delay rates to requests [34]. In 2014, as well as in 2015, Karim Arabi defined "Edge Computing" as the computing executed in applications where real-time data are generated and processed [35], [36]. In contrast with Cloud Computing —which refers to "Big Data" processing— Edge Computing handles real-time data produced by sensors and applications' users.

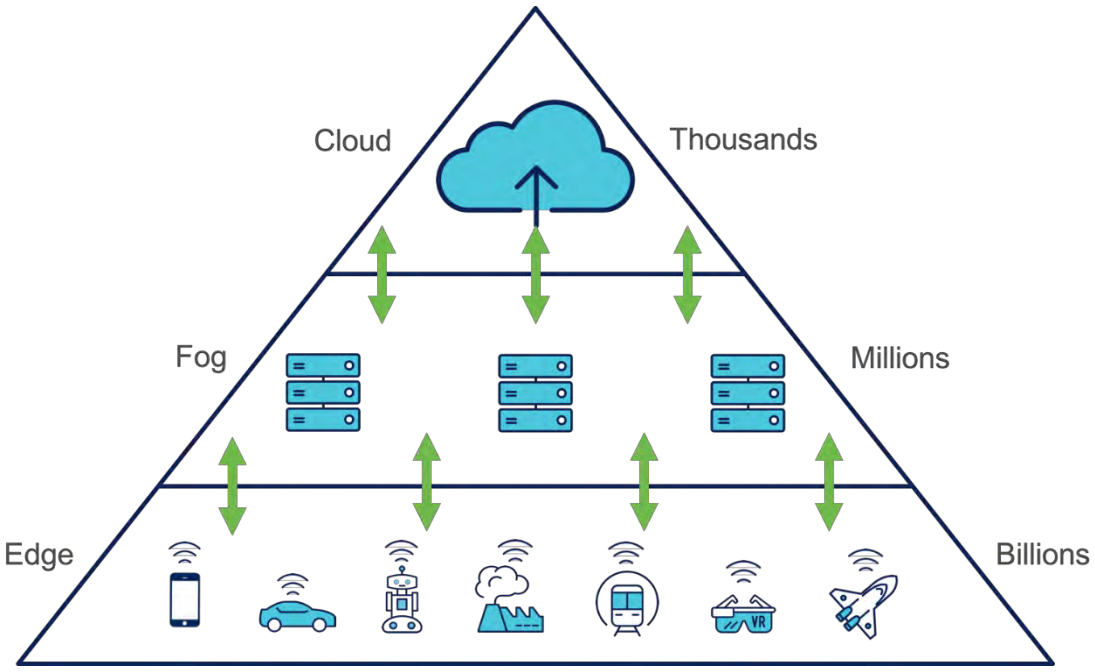


Figure 2.4 Data flow across the Internet of Things architecture [37]

Edge Computing constitutes a computing system which allows the execution of operations nearby the source of data generation instead of delivering the information to data centers or clouds for processing purposes. Hence, applications, data and computational power serve the users in the most rapid and efficient way [38]. Edge Computing systems perform a virtual or physical data separation process bringing off the minimum backhaul traffic, in which several parts of information are gathered to be processed locally and the rest of them are delivered to the central repository.

There are several reasons Edge Computing is considered to be the most suitable computing system such as cases when there is poor network connectivity to IoT devices. Additionally, Edge Computing is preferred in services where increased delays are undesirable since data are processed locally without transmitting to data centers or clouds [39], [40]. While Edge Computing development suggests a great deal of advantages, there are disagreements regarding the security levels it provides. The technology's proponents consider that it is secure due to the fact that data is saved and processed locally while the other side opposes this trader raising the issue of Edge devices' vulnerability compared to the Cloud.

2.3.1 Edge Computing Applications

Edge Computing is widely applied on video streams [41], [42], [43]. In particular, it assumes the role of a swift mediator between the source and the recipients. For instance, during a sports event, a concert or, in general, an event that takes place and can be streamed live, the volume of the produced data is immediately sent to the edge. The edge consists of a number of nodes which are located close to the end-users. Consequently, there is little to no cost or latency in the transmission of the —possibly real-time— video, while also accomplishing to avoid quality problems that could be caused by bottleneck situations due to the huge data traffic on the network if a single node was used.

Another application of Edge Computing concerns traffic management [44]. Due to the massive amount of computations that are involved with traffic management, the need of data conciseness arises. By doing this, the data is reduced without losing important information. Edge computing contributes to that end by analyzing, filtering and compressing data streams before they reach the cloud data centers. Network expenses and, storage and functioning costs are diminished owing to the preprocessing of data.

Autonomous vehicles are also greatly enhanced by Edge Computing [45], [46], [47], [48]. In this case, the necessity of advanced computational power makes the embedded computers obsolete. Safety, spatial consciousness and interoperability among autonomous Artificial Intelligence (AI) systems on vehicles call for exceptionally large bandwidth and real-time parallel computing capacity, which are based on a multitude of composite sensory technologies. This is where edge and distributed computing techniques

take over decreasing the work that would otherwise have to be done onboard the autonomous vehicle, allowing for data corroboration and decision improvement.

Edge Computing assists infrastructure and utilities remote monitoring [49], [50] in that it can be carried out without delay. With the purpose of constantly keeping machinery and systems properly secured, sensors are used to collect data with respect to temperature, humidity, moisture, pressure, radiation, image and sound. All these attributes are relayed from the sensors to the edge via huge data streams, where data analysis and combination takes place in order to extract information regarding the status and functionality of the system.

3 ANALYTICS, DATA MANAGEMENT AND QUERY ALLOCATION

The large volumes of data generated by IoT applications and devices such as mobile and financial services, and humanity sciences are at the heart of the current research. The generated data vary in type, being anything from environmental data to online posts such as videos, photos and text, offering insights which are critical for success in the age of social media [51]. With the emerging technologies and all associated devices, it is predicted that massive amounts of data will be created in the next few years, a trend that will continue for the foreseeable future. It is worth noting that as much as 90% of the current data were created in the last couple of years [52].

3.1 Analytics

Analytics can be used to extract knowledge from data. “Analytics is the process of developing actionable insights through problem definition and the application of statistical models and analysis against existing and/or simulated future data”, according to [53]. This means that if one wants to decide on a matter, the information derived by the analysis on relevant data can be either taken into consideration or ignored, based on metrics such as the confidence level or the statistical significance of the results. Another, simple, definition for analytics can be found in [54]: “an overarching concept that is defined as data-driven decision making”. Unlike the previous definition [53], this approach is solely focused on the purpose of taking action based on information generated by data processing. Over the years, world renowned companies have boosted their revenues and reputations by using analytics. Organizations such as Amazon, Harrah’s, Capital One and Boston Red Sox have dominated their fields by deploying industrial - strength analytics across a wide variety of activities [55].

3.1.1 The Stages of Analytics

The Big Data can be analyzed, thus, result in information which can lead to improvements in daily life by bringing on health advancements, technology progress etc. Research focuses on suggesting new models for the better analysis of both structured[6] and

[6] Structured data is comprised of data in tables that can be easily integrated into a database and, from there, fed into analytics software or other applications.

unstructured[7] data [56]. The process of analytics consists of data collection, organization, cleaning and analysis in order to derive beneficial information.

3.1.1.1 Data collection

Data collection constitutes the procedure of gathering data and extracting information. It is a fact that computers cannot directly handle real-world data since they can only manage digital forms of them. Data acquisition systems (DAS or DAQ) are responsible of converting analog signals into an understandable language for computers [57]. A data acquisition system consists of sensors, signal condition circuitry and analog-to-digital converters. The task of converting real-world behaviors into an electrical signal is performed by sensors. The signal conditioning circuitry takes over the conversion of sensor signals into a form that can be afterwards transformed into digital form by the analog-to-digital converters. After the digitization of the signals, data can be organized and processed by computer systems.

3.1.1.2 Data organization

Data organization refers to data classification and orchestration so that researchers are enabled to conduct a rough breakdown of the study elements that emerged during data collection [58]. Organizing data in exploitable forms lead to better data management since the time and the resources needed to consume in order to analyze them decrease. Data organization can rely on chronological order of information, order of importance, as well as to any information which befalls the research.

3.1.1.3 Data cleaning

Data collections which are stored in files and databases often include inaccurate, incorrect, irrelevant or even missing records in data sets [59]. Data cleaning —also data cleansing or data scrubbing— is the process executed in order to improve data quality by scanning the record sets, correcting or removing the inconsistencies and inaccuracies from data. Several methods have been developed in the field of research for the purpose of data clearance [60]. The most popular data cleaning techniques are demonstrated below:

[7] Unstructured data is data that is raw and unformatted, the kind of data that you find in a simple text document, where names, dates and other pieces of information are scattered throughout random paragraphs [137].

1. **Parsing**

   Parsers enabled to perform this process, detect syntax errors which constitute exceptionable within the data structure permitted. The grammar rules followed are specified by pattern learning techniques.

2. **Data transformation**

   This procedure is prerequisite with the intension to map data between formats since applications are designed and developed so that to process specific data structures and formatting. Normalization and standardization constitute possible transformation techniques so that to reduce or remove inaccuracies within data cleansing.

3. **Duplicate Elimination**

   Duplicate detection methods perform sorting algorithms on data and then get into comparison processes in order to determine the duplicate elements to be removed from the data sets. The verified duplicate entries are derived from rules which fall under specific research areas.

4. **Statistical Methods**

   These techniques are used for both data auditing and fixing incorrections. Metrics such as mean, mode, standard deviation, etc. assist in determining complex unexpected errors and setting anti-values in their positions to a statistical value.

3.1.1.4 Data processing

The stage of data processing requires clean data to enter the system and to be translated into an understandable language for the software. Data processing is performed through algorithms which may vary depending on the source of data being processed and the purpose to be used. Some of the types of data processing used in research areas are demonstrated below:

1. **Batch Processing**

   This type is widely used and, also known as sequential, tacked/queued of offline processing. It enables the processing of different jobs completed by different users in a massive manner, which leads to the reduction of the total processing time and cost.

2. **Real-time processing**

Real-time processing is preferred when there is the need of immediately getting results, displaying them in the lowest time possible. This type of process requires internet connection as data is processed once it enters the system and results are provided almost simultaneously.

3. **Multiprocessing**

Multiprocessing constitutes the most widely used processing type since it divides the data or tasks between the available CPUs so as be processed in parallel and, thus, to increase the system's level of efficiency and throughput. Moreover, in case of a CPU failure, data processing continues as CPUs do not rely to one another.

Subsequently, after the data processing stage, the output of data processing is obtained in forms such as tables, images, charts, vectors, text etc. and interpreted within the investigation taking place.

3.2 Analytics Use Cases

3.2.1 Smart Cities

Big Data and analytics' contribution to the progress of Smart Cities is evident in many publications concerning this matter. Specifically, analytics can be used to ameliorate waste management, traffic management, environmental care, security and planning, among others. In [61], various definitions for the terms "Smart City" and "Big Data" are presented, while Big Data applications' possible benefits, drawbacks and difficulties in implementation are described and commented on. A Smart City's true objectives are the improvement of the quality of life for humankind, the conservation, revivification and achievability of a society [62]. The authors of [61] demonstrate how analytics can improve the management of a Smart City by designing efficient resource utilization, facilitating daily life and smoothly enforcing accountability within a community. These features can make the management of Smart Cities close to effortless, since virtually everything can be automated and followed exactly the way that it must be. However, Smart City can only function properly if the analytics' resulting information is suitably stored and swiftly available [63]. Two models that have been proposed for working on extensive data are the following. In [64], a framework is described that makes the composition of real-time data processing pipelines easier; the platform is based on the block-based programming

paradigm, mirroring that of the Scratch programming language which was designed for elementary school students. The second model can be separated into three parts; one that generates and collects diverse data related to Smart City operations, one which processes the aforementioned data by filtering, analyzing and then storing the results in order for it to be able to later make decisions and order events without human intervention, and another that actually makes the decisions and executes the planned events [65].

A lot of research has been carried out regarding insights coming from real-time data analysis. However, the vast quantity of data and the hardware systems impose restrictions when trying to retrieve a real-time response. On the other hand, users seem to be satisfied with "close-enough" answers provided that they come at a fast pace. Querying data samples and not whole datasets has been a technique which overcomes the aforementioned limitations and satisfies users' expectation [66]. The domain, the underlying infrastructure and several other parameters can affect data sampling. Researchers and academics have already suggested a lot of sampling techniques [67], [68], [69], [70]. Progressive analytics is another solution which overcomes the limitations of Big Data and hardware performance [71]. Specifically, approximate query processing systems handle the degree of inaccuracy in results since the process of data – sampling stops when results determine acceptable levels of accuracy. These techniques opposed to the traditional manual data-sampling do not involve user participation [72]. Moreover, AQP techniques provide confidence intervals on top of which an intelligent mechanism for early results handling could be incorporated by users [73], [74], [75].

A Smart City's resources such as sensors and personal devices generate large amounts of streaming data. The numerous nodes to which the data end up, can be located in various parts throughout the city so as to obtain a profound "picture" of all geographical areas. The reason why data are gathered is to be incorporated into intelligent applications which aim to improve citizens' way of life by making cities serve the community in a knowledgeable manner. As soon as data are collected, they can be processed on-line or put aside for management in the near future.

3.2.2 Marketing

Consumer data are incessantly collected by big firms which offer products or services. They know virtually everything about their customer's preferences by recording

any and every move they may make while on the company's site or posts they upload to social media [76] and make new decisions based almost solely on their demands, seeking new ways to please them even more than before. Customer analytics are used to model customer behavior which is primarily based on market segmentation[8] and predictive analytics [77].

There are difficulties in this endeavor since the volume, velocity and variety [76] of data can be daunting to overcome. Having said that, handled efficiently, data can lift companies to never expected success and profits. More specifically, the customers of a company can be practically innumerable, thus, rendering the data they produce Big Data and more than enough to lead to sound conclusions about the company's next steps. Moreover, considering the rate at which the data is generated, there is no need for any delay in receiving new information. Changes can be made as soon as customers develop new preferences. Lastly, the diversity of the data that is gathered can lead to the categorization of customers into groups. This intermediary step allows for more effective targeting of the populations and more beneficial marketing-strategy building. Accordingly, Big Data Analytics are essential for the desirable success of a company to be within the bounds of possibility.

3.2.3 Security

Analytics are also crucial for fraud detection, cyber security, criminal security and so on and so forth. Data for security analytics are collected from a variety of sources such as IP addresses, emails, log files, information extracted from attack investigations of any kind and are used to identify anomalies and threats, as well as verify alerts and security events to neutralize cyber-attacks or attacks in the physical world [78]. Data used for analysis are not just current; historical data are also brought into play since they can specify a pattern to look for concerning attacks and anomalies. However, not all existing data should be accessible for collection due to the privacy people and enterprises are entitled to.

Fraud detection was one of the earliest systems to be developed by credit card and phone companies. Analytics give it a great boost seeing as they improve previously

[8] Market segmentation refers to the division of existing or potential customers into groups (segments) based on one or more characteristics [133].

unstructured, noisy and incomplete data, additionally correlating, consolidating, and contextualizing them into even more diverse data sources [79]. Furthermore, some common types of cyber-attacks are spamming [80], search poisoning [81], Denial of Service (DoS) [82], phishing [83], [84], malware and website threats [85]. Analytics can aid network administrators especially in the monitoring and surveillance of real-time network streams and real-time detection of both malicious and suspicious (outlying) patterns [86].

3.2.4 Social Media

Social media analytics "is concerned with developing and evaluating informatics tools and frameworks to collect, monitor, analyze, summarize, and visualize social media data … to facilitate conversations and interactions between online communities and extract useful patterns and intelligence…" [87]. According to [88], The Social Media Analytics Process consists of 3 stages. The first is called the capture stage, in which social media data relevant to a topic are gathered from various social media sources. In the second stage —the understand stage— the most relevant data are kept while noisy and incomplete data are ignored; then, "various advanced data analytic methods are employed to analyze the data retained and gain insights from it". The present stage involves showing the results of the previous stage in a telling manner.

3.2.5 Healthcare

Medical data has always been registered despite the fact that, up until recently, it was in hard copy form. Nowadays, technology has taken over and new records are created electronically while some old ones have been digitized over the years. Now, analytics is the easiest way to go through the gathered data which can easily be called Big Data and can in no realistic way be effectively explored otherwise. Big Data in healthcare includes "clinical data from CPOE and clinical decision support systems (physician's written notes and prescriptions, medical imaging, laboratory, pharmacy, insurance, and other administrative data); patient data in electronic patient records (EPRs); machine generated/sensor data, such as from monitoring vital signs; social media posts, including Twitter feeds (so-called tweets), blogs, status updates on Facebook and other platforms, and web pages; and less patient-specific information, including emergency care data, news feeds, and articles in medical journals" [89]. Analytics in healthcare mostly focus on trying to assist doctors in diagnosing a patient, combining symptoms and comparing them with potential causes in

a more effective and fastidious way than a human has the ability to. Of course, the doctor may merely consult with the analytics results and then form their own opinion.

3.3 Data Management

3.3.1 Databases

Databases are collections of related and organized data which serve as information storage and retrieval systems. The database management system (DBMS) is the software which enables the interaction between users and databases [90]. As far as data are concerned, they do not always have the same organizational structure which leads to the use of different database models. The way data are stored, related and manipulated is defined by database design and structure which in turn are determined by a database model that constitutes a type of data model[9]. The most widely used database model is the Relational Database but professionals have developed a great number of additional database models with the intention to meet all research needs. Some characteristics of the most popular database models are demonstrated below [91]:

1. **Relational Model**

   In the relational model, data belongs to a part of multitude that is divided into tuples[10]. The values of each tuple are correlated with each other. Thus, relations are formed within the database.

[9]A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to properties of the real world entities. For instance, a data model may specify that the data element representing a car be composed of several other elements which, in turn, represent the color and size of the car and define its owner [138].

[10]In mathematics, a tuple is a finite ordered list (sequence) of elements. An n-tuple is a sequence (or ordered list) of n elements, where n is a non-negative integer. There is only one 0-tuple, an empty sequence, or empty tuple, as it is referred to. An n-tuple is defined inductively using the construction of an ordered pair [139].

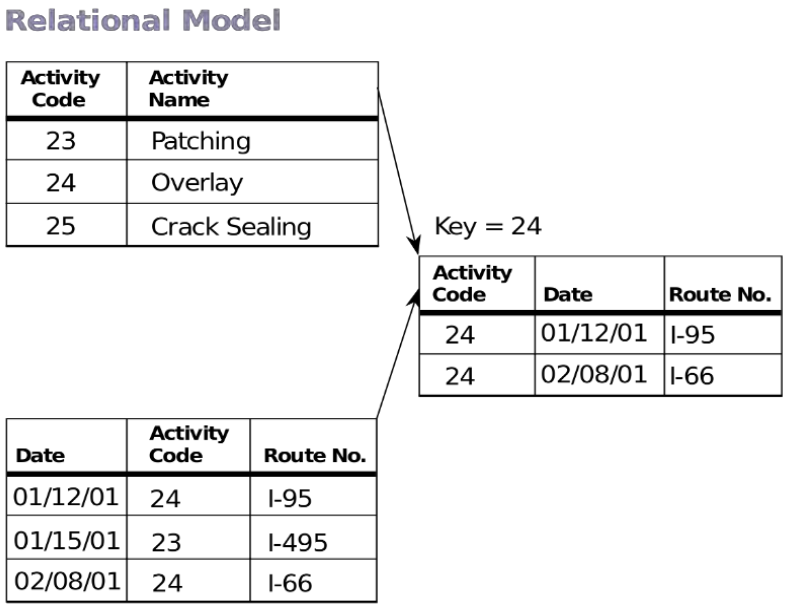


Figure 3.1 Relational Database Model [92]

2. Hierarchical Model

This model determines the organization of data into a tree structure with a single root to which all information is linked. Moreover, a single entry may have relationships to many other elements (one to many) of different data type.

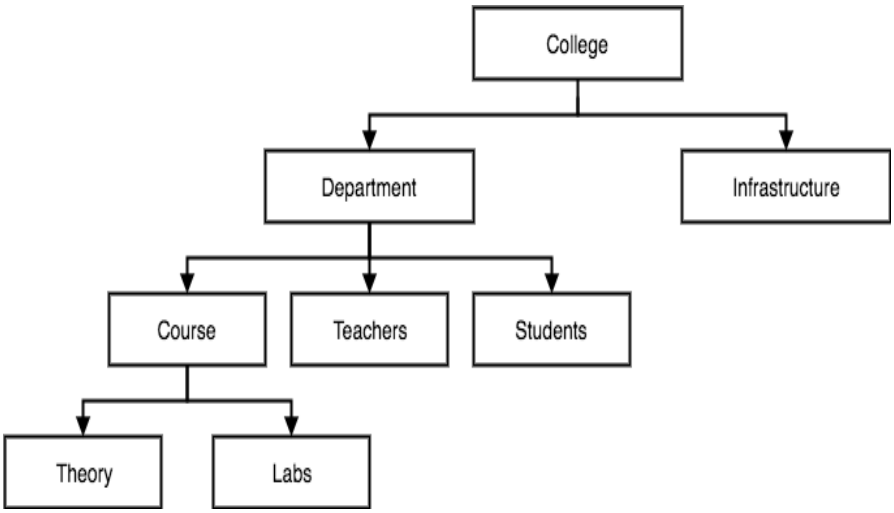


Figure 3.2 Hierarchical Database Model [93]

3. **Network Model**

The Hierarchical model extends to the Network model and determines the organization of data into a graph structure. In the Network model, elements have relationships to a lot of elements (many to many).

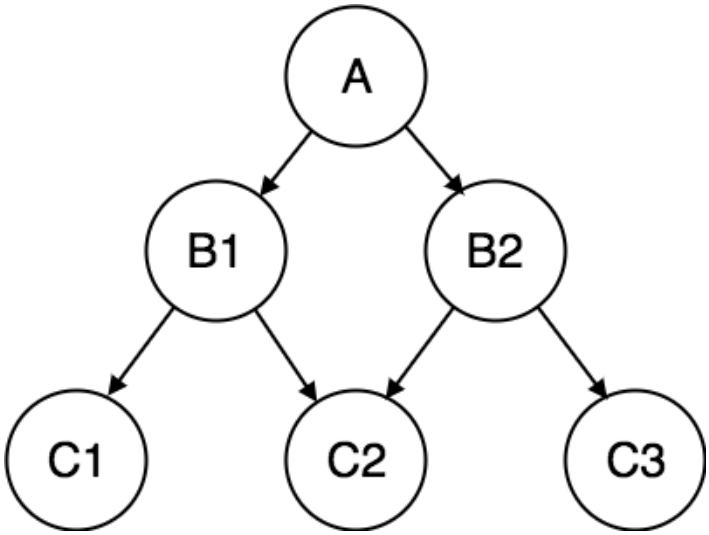


Figure 3.3 Network Database Model [91]

3.3.2 Query Management

As far as the gathered data are concerned, they are stored in databases and can be retrieved by executing the appropriate queries. Practically, queries are requests for information usually issued by human users. Such requests include catalog lookups and simple or complex transactions which either read or change database content [94]. Query languages, like SQL, or natural language questions may form the queries. Query clients are programs responsible for submitting a query whereas query processors constitute the programs which deal with query processing. QPs put together query results by combining the requested information coming from a set of different databases. Results are communicated back to the specific query client which demanded them. The aforementioned programs can either be executed on the same machine or on a distributed computer system [95].

3.4 Data Partitioning and Query Allocation

3.4.1 Data Partitioning

Data Partitioning is the separation of data into distinct and independent partitions in order to facilitate manageability and control. The elements which are actually divided are the databases and they can be either built into small pieces or separated into pieces after their creation, in a later phase. Hence, tables, indexes and index-organized tables are divided into small parts leading to performance increasement since engineers can work just with specific and relevant data [96]. Moreover, data availability is improved since every partition constitutes an independent object. It is worth noting that, partitioning decreases costs as data is stored in a no time consuming and a minor processing power manner.

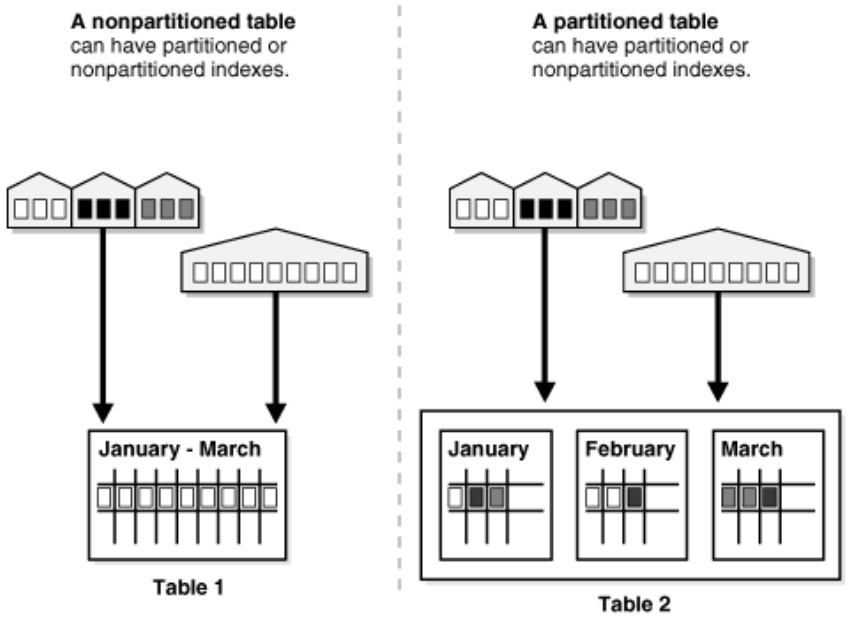


Figure 3.4 Database Partitioning [97]

The major types of separation are **vertical** and **horizontal** partitioning. The former concerns the partitioning using a rule for the table's rows and the latter concerns the partitioning based on a formula for the table's columns. Oracle and Microsoft database technologies offer several partitioning techniques such as range partitioning which enables the data division based on value ranges, list partitioning through which a list of values defines the data distribution, etc. The portioning methods vary so as to serve business and research needs.

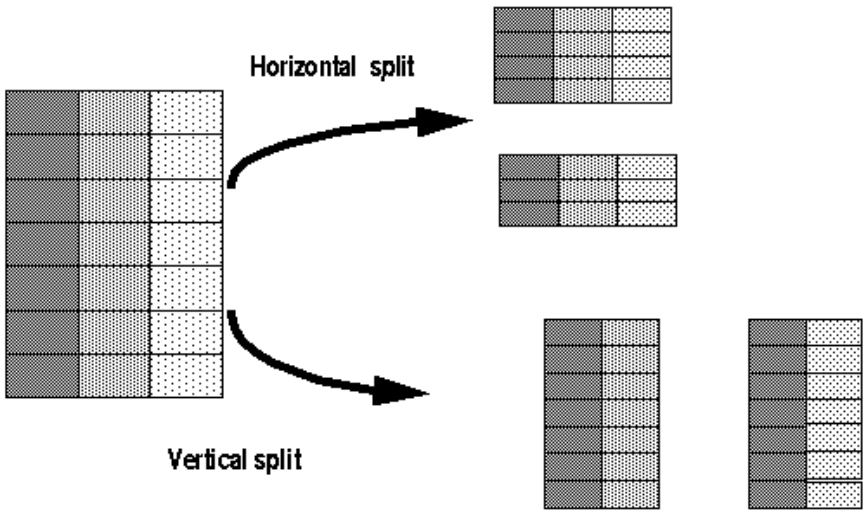


Figure 3.5 Horizontal and Vertical Partitioning [98]

3.4.2 Query Allocation

A Database Management System (DBMS) server can run as a distributed system, consisting of a network of a master node and a set of lower-layer nodes, i.e. QPs. In this case, a master process is executed on the master node which coordinates and assigns tasks to the processes on the slave nodes through an interconnected network. More specifically, as soon as a query is received by the master process a decision must be made about the slave node(s) which will finally execute it. Afterwards, the partial results are aggregated by the master node and the final information is sent back to the user application. As queries arrive at a high rate, each slave node is responsible to handle streams of queries. Researchers have devoted significant effort to develop several frameworks which handle streams query processing efficiently [99], [100], [101], [102], [103], [104]. Data can be split up or duplicated into multiple nodes in order to minimize the response time and memory space needed for computations [105]. Furthermore, this separation/duplication facilitates the concurrent production of information through parallel data processing. This is necessary due to the fact that requests may be too many for a single machine to handle or because of a possible node failure[11]. The query execution on top of multiple data partitions has already been introduced by companies such as Microsoft[12] and Oracle[13].

[11] https://docs.funnelback.com/more/extra/supporting_multiple_query_processors.html
[12] https://docs.funnelback.com/more/extra/supporting_multiple_query_processors.html
[13] https://docs.oracle.com/cd/A58617_01/server.804/a58238/ch1_unde.htm#2820

4 THE PROPOSED MODELS

Our problem can be modeled or categorized as an assignment problem. The assignment problem is the problem of mapping each of n tasks to one of the m available workers in the most efficient manner. The total cost equals the sum of the partial costs of every individual allocation.

Our work focuses on allocation techniques which build on top of queries' characteristics, the state and the features of the system. The objective of such algorithms is the achievement of an optimal query allocation which leads to the minimization of execution and communication costs, load balancing among the nodes and, generally, to the efficient usage of the system resources. In our discussion, communication costs are considered negligible and, thus, are omitted.

Let a set of n queries be $Q = \{q_1, q_2, q_3, \dots q_n\}$ and a set of m nodes/QPs be $P = \{p_1, p_2, p_3, \dots p_m\}$ in a database system. Every incoming query q_i is defined by two characteristics, i.e., its complexity c_i and its deadline d_i $\forall i = 1,2, \dots, n$. The former represents the 'magnitude' of calculations required for the successful completion of the transaction, while the latter stands for the time constraint, set by the user. The values of these characteristics can vary greatly, all the while resembling those of a real-world functioning system. At the same time, for each node/QP p_j, s_j denotes the processing speed and l_j its load $\forall j = 1,2, \dots, m$. Specifically, the speed constitutes a feature set by the rate at which data is processed, whereas the load depicts the amount of queries waiting for execution by the node during a specific time unit. In our implementations, the characteristics' range in the interval of [0, 1].

A function which aims to allocate the set of queries to the appropriate nodes can be formally described as $f: Q \rightarrow P$. The number of possible allocation solutions among queries and nodes is m^n. Consequently, the use of a brute-force algorithm is not an affordable option for any application demanding (near) real-time results. In the following subsections, we present algorithms which provide an optimized and real-time-friendly approach for the assignment problem.

4.1 Equal number of Queries and Query Processors

4.1.1 The Hungarian Method

The Hungarian method constitutes a combinatorial optimization algorithm[14] which was the first to solve the assignment problem with a 'reasonable' complexity. The method is used when the number of queries equals the number of nodes, whereas there are extensions of the algorithm not conforming to this constraint. In 1955, Harold W. Kuhn, influenced by the earlier work of the Hungarian mathematicians Dénes Kőnig and Jenő Egerváry, published a paper presenting this method [106]. Later on, in 1957, James R. Munkres released a new and improved version of the original algorithm, which was referred to as the Kuhn-Munkres algorithm or Munkres assignment algorithm from that point onwards [1]. The time complexity of the original version of the algorithm was $O(n^4)$, whereas the Kuhn-Munkres method yields a complexity of $O(mn^2)$ for orthogonal matrices.

In 1965, Jack R. Edmonds generalized the Hungarian method to solve the assignment problem for both bipartite and non-bipartite matchings [107], also noticing along with Richard M. Karp that a complexity of $O(n^3)$ is achievable for the Hungarian method [108]. Furthermore, Tomizawa [109] as well as E. A. Dinits [110] proved that the algorithm is strongly polynomial. Lester R. Ford Jr and Delbert R. Fulkerson developed an extension of the algorithm for general transportation problems [111]. In 2006, researchers uncovered that Carl Gustav Jacobi had already solved the assignment problem during the 19th century. He provided an equivalent solution to the Hungarian algorithm which was published after his death in 1890 in Latin [112]. It is also worth noting that, in 2016, Chidambaram Annamalai proposed an approach for the solution of a generalized version of the assignment problem concerning hypergraphs [113].

4.1.1.1 The Kuhn-Munkres Algorithm

Let A be a $n \times n$ matrix representing the costs of each of n workers performing any of n tasks. More specifically, every element a_{ij} of this matrix represents the cost for task i

[14] Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects. In many such problems, exhaustive search is not tractable. It operates on the domain of those optimization problems, in which the set of feasible solutions is discrete or can be reduced to discrete, and in which the goal is to find the best solution [140].

to be performed by worker j. The aim of this algorithm is to allocate the tasks to the workers properly, so as to minimize the total cost.

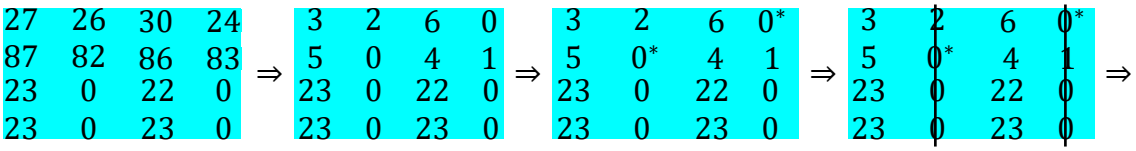
In Algorithm 1, we present the Hungarian method, as modified by J. Munkres.

Algorithm 1 The Kuhn-Munkres Algorithm

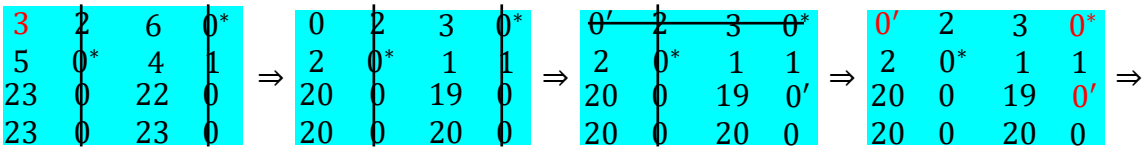
Input: Set of n tasks T, Set of n workers W
Output: Solution Allocation $X(t_i, w_j), \forall i = 1, 2, \ldots$
Begin
$A = create\_cost\_matrix(T, W)$;
for each row i **do**
$min\_val = find\_smallest\_row\_element()$;
for each column j **do**
$a_{ij} -= min\_val$;
end for
end for
for each row i **do**
for each column j **do**
if $(a_{ij} == 0$ && ! $star\_in\_row(i)$ && ! $star\_in\_col(j))$ **then**
$star(a_{ij})$;
end if
end for
end for
while (true) **do**
$covered\_cols = cover\_columns\_with\_starred\_zeros()$;
if $(covered\_cols == n)$ **then**
for each row i **do**
$X_i = find\_starred\_zero\_in\_row(i)$;
end for
return X;
else
while (true) **do**
while (true) **do**
$r, c = find\_uncovered\_zero()$;
if $(c == -1)$ **then**
break;
end if
$prime(a_{rc})$;
$col = find\_starred\_zero\_in\_row(r)$;
if $(col\ ! = -1)$ **then**
$cover\_row(r)$;
$uncover\_col(col)$;
else
$add\_to\_path(r, c)$;
while (true) **do**
$r = find\_starred\_zero\_in\_col(c)$;
if $(r\ ! = -1)$ **then**
break;
end if
$add\_to\_path(r, c)$;
$c = find\_primed\_zero\_in\_row(r)$;
$add\_to\_path(r, c)$;
end while
for each element e in path **do**

```
                        if (starred(e)) then
                            unstar(e);
                        else if (primed(e)) then
                            star(e);
                        end if
                    end for
                    erase_primed();
                    uncover_rows_and_cols();
                end if
            end while
            min_uncovered = find_min_uncovered_val();
            for each row i do
                for each column j do
                    if (row_covered(i)) then
                        a_ij += min_uncovered;
                    end if
                    if (! col_covered(j)) then
                        a_ij −= min_uncovered;
                    end if
                end for
            end for
        end while
    end if
end while
End
```

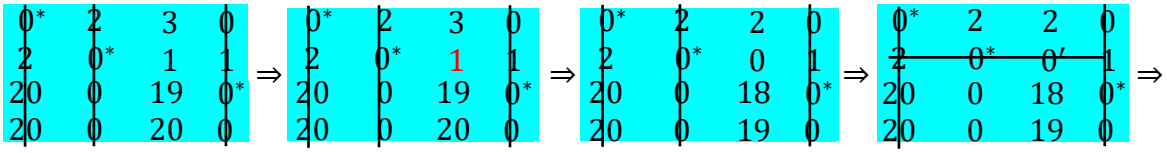
Here is an example of how the algorithm finds the solution, given a cost matrix.



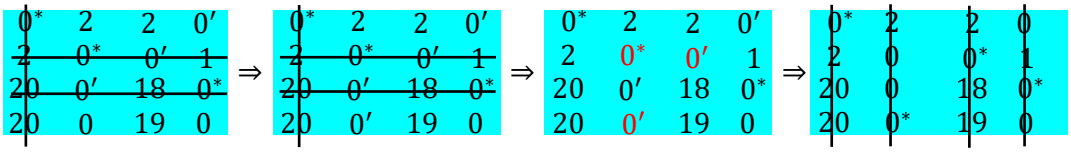
The preliminaries, as Munkres calls the first steps.



Step 1 Step 3 Step 1 Step 2



Step 2 Step 1 Step 3 Step 1



Step 1 Step 1 Step 2 Step 2

The resulting Query-Node allocation pairs are: (1, 1), (2, 3), (3, 4), (4, 2).

4.1.1.2 Adopting the Algorithm in our problem

Each element of the cost matrix A is calculated as demonstrated by Algorithm 2. The first step is to calculate the rounded ratio of the query complexity compared to the query deadline. This ratio represents the speed demanded for the query to be executed and completed in compliance with the deadline. Subsequently, the node's speed is compared to the aforementioned ratio in order to decide whether the node is fast enough to serve the query in the required time interval. Then, the node's load is compared to specific high and low thresholds. For each speed and load interval combination, an appropriate reward or penalty is attributed to the allocation cost. We consider 0.8 as a high threshold and 0.3 as a low threshold for a node's load, whereas the interval between the aforementioned thresholds can be thought of as a medium value. We implemented the rest of the Hungarian algorithm following the steps J. Munkres describes in his paper [1].

Algorithm 2 The Cost Calculation

Input: complexity c_i, deadline d_i, speed s_j, load l_j
Output: cost C_{ij}
Begin
  $cost = round(\frac{c_i}{d_i})$;
  if $(s_j > cost)$ ***then***
    if $(l_j > 0.8)$ ***then***
      $cost += random\_integer(5, 10)$;
    else if $(l_j < 0.3)$ ***then***
      $cost -= 20$;
    else
      $cost += random\_integer(2, 5)$;
    end if
  else
    if $(l_j > 0.8)$ ***then***
      $cost += 20$;
    else if $(l_j < 0.3)$ ***then***
      $cost += random\_integer(2, 5)$;
    else
      $cost += random\_integer(5, 10)$;
    end if
  end if
  return $cost$;
End

4.2 Number of Queries exceeds number of Query Processors

4.2.1 Clustering

Clustering is the process of grouping objects of similar characteristics. The cluster analysis, in all its forms and algorithms, constitutes a common technique for data mining and statistical data analysis. It provides solutions to a variety of Research & Development areas, such as machine learning, pattern recognition, image analysis, computer graphics, information retrieval, data compression and bioinformatics. Its origin lies in H. Driver's and A. Kroeber's research during 1932 in the field of anthropology [114]. Later on, J. Zublin in 1938 [115] and R. Tryon in 1939 [116] applied the idea of clustering to psychology. Moreover, in 1943 Cattell [117] went even further, in the context of his research field of personality psychology, by proposing the trait theory classification.

4.2.1.1 The Clustering Task Allocation Algorithm

The basic idea of cluster analysis is adopted and adjusted by H. Meireles Valadares in order to solve the task assignment problem [118]. The clustering mathematical approaches [119] are not used in this algorithm and, thus, any depiction of the clusters serves just for understanding purposes. A "cluster centroid" depicts any task which is not yet allocated to a worker, whereas a "cluster" is a set of workers that compete for the task represented by the corresponding cluster centroid. In addition, the four following notions are incorporated in the algorithm. The term "free worker" describes a worker that is not executing any task at present, while a "candidate" is a free worker which belongs to a specific cluster. The term "distance" represents the difference between the workers' capacities and the tasks' requirements, and "speed" is the worker's processing rate.

The algorithm creates a cluster centroid for every task and populates its cluster with free workers, rendering them candidates. The selection of the candidates is determined by the heuristic R which is defined, for every worker, by the following equation:

$$R = \frac{distance}{speed} \quad (1)$$

The workers whose R value is the minimum are chosen as candidates to a cluster. The worker that will finally execute the specific task is chosen among its candidates by using the cost function presented in Algorithm 2. It should be noted that the tasks' priorities are

taken into consideration in the cluster population process, as well as in the candidate selection process.

In Algorithm 3, the Clustering Task Allocation algorithm is presented.

Algorithm 3 The Clustering Task Allocation Algorithm

Input: Set of n tasks T, Set of m workers W
Output: Solution Allocation $X(t_i, w_j), \forall i = 1, 2, \dots n$
Begin
$new\_tasks = receive\_new\_tasks()$;
$tasks = count\_all\_tasks()$;
$free = count\_free\_workers()$;
if $(free == 0)$ **then**
$add\_to\_queue(new\_tasks)$;
end if
$prioritize\_all\_tasks()$;
$clusters = min(tasks, free)$;
$define\_new\_tasks\_as\_cluster\_centroids()$;
if $(tasks \geq free)$ **then**
$C = 1$;
$remainder = 0$;
if $(tasks > free)$ **then**
$add\_to\_queue(extra\_tasks)$;
end if
else
$C = free \div tasks$;
$remainder = free \,\%\, tasks$;
end if
for each cluster i in priority **do**
for $c = 1, 2 \dots C$ **do**
for each free worker j **do**
$R_{ij} = \frac{distance_{ij}}{speed_j}$;
end for
$worker = find\_worker\_with\_min\_R\_for\_task(i)$;
$add\_to\_cluster(worker, i)$;
end for
end for
for each cluster i in priority **do**
for each free worker j **do**
$R_{ij} = \frac{distance_{ij}}{speed_j}$;
end for
$worker = find\_worker\_with\_min\_R\_for\_task(i)$;
$add\_to\_cluster(worker, i)$;
end for
for each cluster i **do**
for each worker j in cluster i **do**
$cost_{ij} = cost\_function(i, j)$;
end for
$best = find\_worker\_with\_min\_cost(i)$;
$X_i = best$;
end for
return solution X;
End

An example execution of the above algorithm is presented. In the example, the tasks are considered as locations that need to be visited and the workers as the vehicles that will make the trips. We suppose that there are 8 vehicles and 3 locations. They are graphically shown in Figure 4.1.

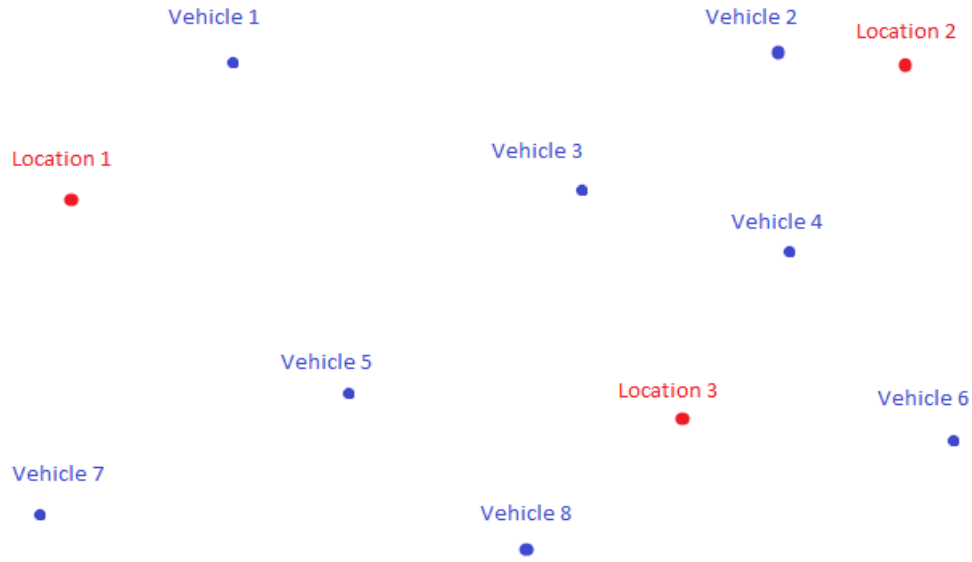


Figure 4.1 Clustering algorithm example execution: Map of 3 Locations and 8 Vehicles

We assume that all vehicles are available for assignment to a location and the priority of the locations is Location 2, Location 1 and Location 3. We define every location as a cluster centroid and calculate the number of vehicles that will be assigned to each cluster. Since the vehicles are more than the locations, there are going to be at least $C = \frac{free}{tasks} = 8 \div 3 = 2$ vehicles in each cluster, with a remainder of 2 more for the highest locations in priority. Now, we are going to find out which vehicles are going to be assigned to each cluster. For that, the speeds of the vehicles, as well as their distances with the locations are needed.

For the most prioritized location, Location 2, we are going to calculate the R values of the algorithm for every vehicle. The 2 vehicles with the smallest R value are Vehicles 2 and 4 and, therefore, are added to Cluster 2. The second most prioritized location is Location 1, for which the two vehicles with the smallest R value from the remaining ones are Vehicles 1 and 5. On to Location 3, the 2 chosen vehicles for its cluster are Vehicles 6 and 8. It is evident that for Location 3 Vehicle 5 is the best choice, however, it is already taken by Location 2, whose priority is greater than the other locations. The 2 remaining

vehicles are going to be assigned to clusters 2 and 1. Cluster 2, due to its greater priority is going to get Vehicle 3, while Cluster 1, gets the remaining Vehicle 7.

Vehicle	Average Fuel Consumption (l/h)	Speed (km/h)	Distance from Locations (km)		
			1	2	3
1	5	40	3	8	7
2	6	70	8	2	5
3	3	45	7	4	3
4	7	65	9	3	3
5	8	90	4	8	4
6	7	75	11	5	3.5
7	4	50	4	12	8
8	6	80	7	7.5	3

Table 1 Clustering algorithm example execution: Vehicle Speeds and Distances from Locations

R values			
Vehicle	Location		
	1	2	3
1	0.075	0.2	0.175
2	0.114	0.029	0.071
3	0.156	0.089	0.067
4	0.138	0.046	0.046
5	0.044	0.089	0.044
6	0.147	0.067	0.047
7	0.08	0.24	0.16
8	0.088	0.094	0.038

Table 2 Clustering algorithm example execution: R values

The last step of the algorithm is for every location to choose one of the vehicles in its cluster to actually execute it. This decision is based on the cost function values that are calculated for each candidate. Let's assume that in this example, the cost function represents the fuel that is needed for the trip and is calculated as $Cost_{ij} = f_j * R_{ij}$, where f is the average fuel consumption per hour and R the values generated just before and we are looking for the smallest cost. Based on the data provided by Table 1, the costs for each

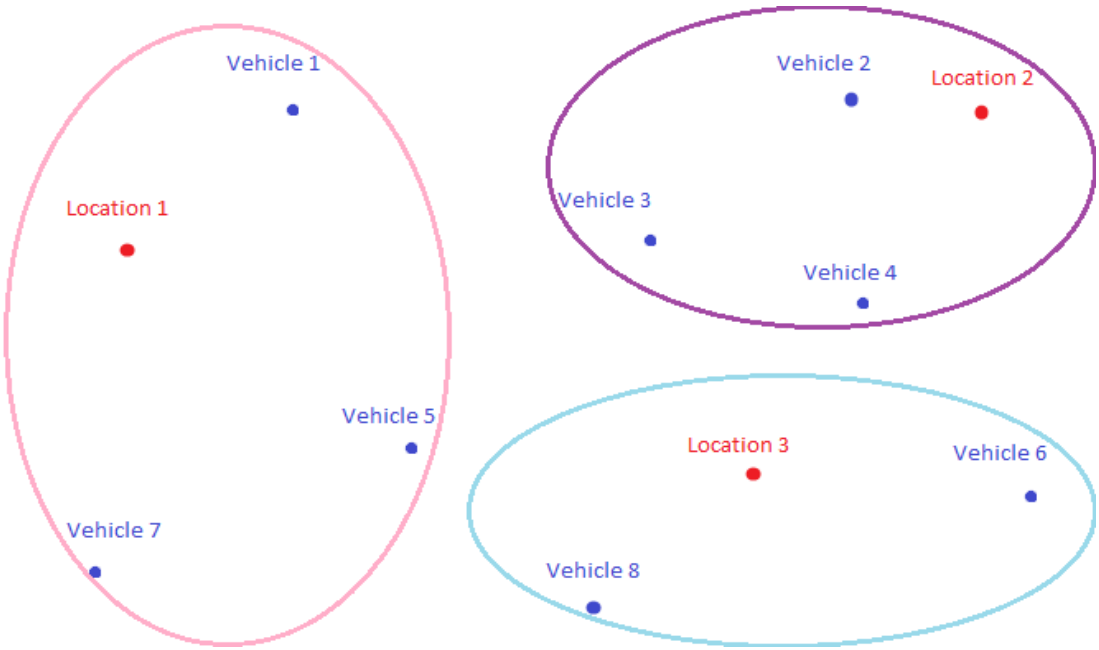


Figure 4.2 Clustering algorithm example execution: Final clusters

cluster are generated and shown on Table 3. Hence, the assignments of the locations to vehicles are: Vehicle 7 to Location 1, Vehicle 2 to Location 2 and Vehicle 8 to Location 3.

Cluster 1		Cluster 2		Cluster 3	
Vehicle	Cost	Vehicle	Cost	Vehicle	Cost
1	0.375	2	0.174	6	0.329
5	0.352	3	0.267	8	0.228
7	0.32	4	0.322		

Table 3 Clustering algorithm example execution: Cost values

4.2.1.2 Adopting the Algorithm in our problem

As the system receives new queries, free nodes are counted and, in case all the available nodes are busy, the queries enter a waiting queue. Otherwise, the queries are prioritized by the use of the $Funcion\ (2)$, where d_i represents the query's deadline and i specifies the query's generation order. Specifically, the queries are prioritized from low to high values.

$$priority_i = d_i + \frac{i}{\max(i)} \quad (2)$$

The next step is to define every new query as a cluster centroid and then calculate and set the number of candidates belonging to each cluster. Choosing which nodes are assigned to a cluster is directed by the heuristic $Functin\ (3)$. The clusters select their candidates based on the priority order of the cluster centroid.

$$\frac{distance_{ij}}{speed_j} = \frac{1 - l_j - c_i}{s_j} \quad (3)$$

Recall that l_j denotes the node's load, c_i represents the query's complexity while s_j stands for the node's speed. The last step of the algorithm involves making the final decision about the node —among the candidates— which will execute the query. This decision is based on the costs calculated as demonstrated in Algorithm 2.

4.3.1 Swarm Optimization

Particle Swarm Optimization (PSO) constitutes a method which involves a population of candidate solutions or, in the words of the algorithm, a swarm of particles. A candidate solution —particle— can be improved in an iterative manner, given a quality threshold. Consisting in the constant search of the best solution, the algorithm uses mathematical functions —taking into consideration the parameters of position and velocity— over the particles in order to cause their movement within the search-space. The movement of each particle is influenced by its own local best position as well as by the best-known position among all particles in the search-space. As particles move around, the discovery of better positions by some particles leads to the movement of the entire swarm towards that area of solutions. The whole process hopefully results to the convergence of the swarm to the best solution, despite there being no guarantee.

The first exposure of the PSO is dated back to 1995, when J. Kennedy and R. Eberhart, in their attempt to simulate swarm behaviors —bird flocks and fish schools—, proposed the paper "Particle Swarm Optimization" [120]. A philosophical aspect of the PSO algorithm and the broader concept of swarm intelligence are described in their book, the "Swarm Intelligence" [121]. Later on, in 1998, R. Eberhart and Y. Shi, modified the original algorithm by introducing a new parameter called inertia weight [122]. Furthermore, R. Poli gathered and analyzed publications within his survey related to PSO [123]. A recent research presented by M. Bonyadi and Z. Michalewicz, includes a concise review related to theoretical as well as experimental PSO projects [124].

In 1997, Kennedy suggested that the PSO algorithm be simplified [125] an idea which has been studied extensively [126], [127], [128], [129] and appeared to improve the optimization performance as well as the tuning and performance consistence across different platforms. The original PSO algorithm requires the initialization of velocities for

the particles, which possibly demands more input, whereas a variant that has no need for velocities is described in [130] by Kennedy and a simpler and usually faster one is proposed in [131].

4.3.1.1 The Simplified Swarm Optimization Algorithm

For the solution of our problem, we have chosen to implement the Simplified Swarm Optimization Algorithm for the Task Allocation Problem, presented by Yeh et al. [132]. In this variant of the PSO algorithm, the concept of stochasticity is introduced and incorporated in the choosing of a particle's next position. In every iteration, each candidate solution's fitness value is calculated according to a fitness function that is based on the execution and communication costs —ec and cc respectively— of the particle and the memory and processing constraints, which are essentially calculating how much of the memory or processing capacities of the nodes are left given the requirements of the tasks allocated to them. A candidate solution's personal best position is updated if the current fitness value is better, and the global best position is also updated to be the best of the local best solutions. The last step for every iteration is to calculate each solution's new position choosing every task's worker randomly among its personal best, the global best, its current position and a random value.

In Algorithm 4, the Simplified Swarm Optimization algorithm is presented.

Algorithm 4 The Simplified Swarm Optimization Algorithm for the Task Allocation Problem in a distributed computing system

Input: Set of n tasks T, Set of m QPs P, number of candidate solutions S

Output: Solution Allocation $X(t_i, w_j), \forall i = 1, 2, \dots n$

Begin

$initialize\_constants()$; // C_g, C_p, C_w, λ

$X = produce\_random\_candidate\_solutions(S)$;

$first = true$;

while $(criterion\_not\_met())$ **do**

for each candidate solution X_k **do**

$$fX_k = \sum_{j=1}^{m}\sum_{i=1}^{n} ec_{ij}(X_{ki} == j ? 1 : 0) + \sum_{i=1}^{n-1}\sum_{w=i+1}^{n} cc_{iw}(1 - \sum_{j=1}^{m}(X_{ki} == j ? 1 : 0)(X_{kw} == j ? 1 : 0));$$

$$FX_k = fX_k + \lambda(\sum_{t=1}^{m}\frac{[\max(0, M_t - \sum_{i=1}^{n} m_i(X_{ki} == t ? 1 : 0))]^2}{\sum_{w=1}^{S}[\max(0, M_t - \sum_{i=1}^{n} m_i(X_{wi} == t ? 1 : 0))]^2} + \sum_{t=1}^{m}\frac{[\max(0, P_t - \sum_{i=1}^{n} p_i(X_{ki} == t ? 1 : 0))]^2}{\sum_{w=1}^{S}[\max(0, P_t - \sum_{i=1}^{n} p_i(X_{wi} == t ? 1 : 0))]^2});$$

$best = 1$;

if $(first \;||\; FX_k > pBestFitness_k)$ **then**

$pBest_k = X_k$;

$pBestFitness_k = FX_k$;

$first = false$;

end if

if $(pBestFitness_k > pBestFitness_{best})$ **then**

$best = k$;

end if

$gBest = pBest_{best}$;

end for

for each candidate solution X_k **do**

for each query X_{ki} **do**

$\rho = random\_double(0,1)$;

if $(\rho < C_g)$ **then**

$X_{ki} = G_i$;

else if $(\rho < C_p)$ **then**

$X_{ki} = P_{ki}$;

else if $(\rho \geq C_w)$ **then**

$X_{ki} = random\_integer(1, m)$;

end if

end for

end for

end while

End

An example execution of the first iteration of the algorithm will be presented as follows. Suppose there are three tasks that don't communicate with each other, two nodes, three candidate solutions and $C_g = 0.5, C_p = 0.85, C_w = 0.95, \lambda = 10^{10}$, the first iteration of the algorithm will play out as follows.

The characteristics of the tasks and the nodes, whose values range in the interval [0,1], are provided in Table 4 and the execution costs for each combination are shown on Table 5. Let's assume that the execution starts with the following three random candidates solutions: $X_1 = [2, 1, 1],\ X_2 = [1, 2, 2],\ X_3 = [2, 1, 1]$.

Task	Memory requirement	Processing requirement	Node	Memory capacity	Processing capacity
1	0.69	0.29	1	0.89	0.54
2	0.23	0.47	2	0.26	0.67
3	0.93	0.41			

Table 4 SSO algorithm example execution: Tasks and Nodes

Task	Node	
	1	2
1	0.54	0.43
2	0.87	0.7
3	0.76	0.61

Table 5 SSO algorithm example execution: Execution Costs

The $pBest$ solutions are identical to the candidate solutions X_1, X_2, X_3. In subsequent iterations, the current fitness of a particle i would be compared to the $pBestFitness_i$ value and the candidate solution version with the best value would be kept as $pBest_i$. The $gBest$ solution is $X_2 = [1, 2, 2]$.

Suppose that for the first candidate solution we have:

$i = 1, \rho = 0.27: X_{11} = G_1 = 1,$

$i = 2, \rho = 0.64: X_{12} = G_2 = 1$ and

$i = 3, \rho = 0.99: X_{13} = G_3 = random\_integer(1,2) = 2,$

hence the updated X_1 will be $X_1 = [1, 1, 2]$. X_2 and X_3 are updated similarly and, while X_1 and X_3 had the same value before, they could be totally different after the change. The $gBest$ solution simulates the most efficient solution for the assignment problem.

4.3.1.2 Adopting the Algorithm in our problem

The cost of a query q_i's execution by a specific node p_j is calculated by $Function\ (4)$, while the communication costs are ignored, as we have already mentioned.

$$ec_{ij} = \frac{c_i}{s_j} \quad (4)$$

On to the constraints, the memory capacity of a node has been replaced by its load l_j and its processing capacity refers to the node's speed characteristic s_j. A query's memory requirement has been replaced by its complexity feature c_i and its processing requirement by the deadline d_i. The process has no stopping criterion except that it is repeated for several iterations.

5 EVALUATION

We present the performance of the proposed allocation algorithms, i.e., the Hungarian Method (HM), the Clustering Task Allocation Algorithm (CTA) and the Simplified Swarm Optimization Task Allocation Algorithm in a Distributed Computing System (SSO-TA). We adopt a set of performance metrics and simulate a query streaming environment by setting the values of the queries and nodes' characteristics. Specifically, a node's load value is drawn from two datasets; the Cooling Load feature of the Energy Efficiency dataset[15] and the Computer Utilization feature of the Optical Interconnection Network dataset[16]. At this point, we highlight that the aforementioned values are used in the experiments after their normalization in the interval $[0,1]$. The remaining features —the node's speed and the query's complexity and deadline— are initialized randomly and uniformly in the same interval.

5.1 Performance Metrics

We define the metric T which represents the time required for concluding an allocation of a query q_i to a query processor p_j. This metric's values may vary depending on the algorithm used and the number of queries or nodes involved in the allocation process. Our main aim is to reduce the average T value for each simulation scenario.

As far as the chosen node's load concerns us, we adopt the metric Λ. This metric depicts the difference of the selected node's load with the lowest load among all nodes. The following equation holds:

$$\Lambda = l_{selected} - l_{lowest} \quad (5)$$

When $\Lambda \to 0$ the selected node's load is low, thus this node is the most appropriate to serve the query since the waiting time for its execution is the least among all nodes. In contrast, when $\Lambda \to 1$ this node is the worst to choose with respect to the waiting time for the query's execution.

Moreover, another metric that we use in our experiments in order to evaluate the selected node's speed is Σ. It is defined as the difference of the highest speed among all nodes with the chosen node's speed. The Σ metric is calculated by the following equation:

[15] https://archive.ics.uci.edu/ml/datasets/Energy+efficiency
[16] https://archive.ics.uci.edu/ml/datasets/Optical+Interconnection+Network+

$$\Sigma = s_{highest} - s_{selected} \quad (6)$$

When $\Sigma \to 0$ the selected node's speed is high, thus this node is the most appropriate to serve the query since the query will be processed and the results will be returned as early as possible. On the other hand, the greater the Σ metric is, the lower the processing speed for the query will be.

The final metric used, Φ, is a linear combination of the Λ and Σ metrics and it is specified as follows:

$$\Phi = \alpha * \Lambda + (1 - \alpha) * \Sigma, \qquad \alpha \in [0,1] \quad (7)$$

where a constitutes a factor, whose value is set depending on the metric —between Λ and Σ— we would like to focus on and give more significance to. More specifically, when $a \to 0$ Σ gains full attention and Λ has no importance in $Equation\ (7)$, as $a \to 1$ Λ is the main parameter which affects $Equation\ (7)$. In case $a = 0.5$, the Φ metric depends equally on Λ and Σ. When $\Phi \to 0$ the best performance for Λ, Σ that our scheme achieves at the same time. It should be noted that the higher the Φ metric is, the lower the overall performance becomes.

5.2 Performance Assessment

5.2.1 The Hungarian Method

In Figure 5.1, we present our results regarding the T metric, which represents the average time required for a query to be allocated to a node. Generally, as the number of queries N increases so does the T metric. We notice that when $N \in \{10, 50, 100\}$ the time values remain almost constant since they are increasing at a small rate. On the contrary, when $N \in [100, 1000]$, T increases drastically and especially when $N \in [500, 1000]$ the highest time costs are observed.

The results of Λ vs N are demonstrated in Figure 5.2. We notice that for small N values we have low Λ outputs until the metric's behavior is stable. For a greater number of queries/nodes the Λ metric remains almost invariant. This happens due to the fact that all nodes are allocated exactly one query to execute, thus the mean Λ value is equal to the mean node load value minus the minimum load value, which is a constant value:

$$E[\Lambda] = E[l - l_{min}] = E[l] - l_{min} \quad (8)$$

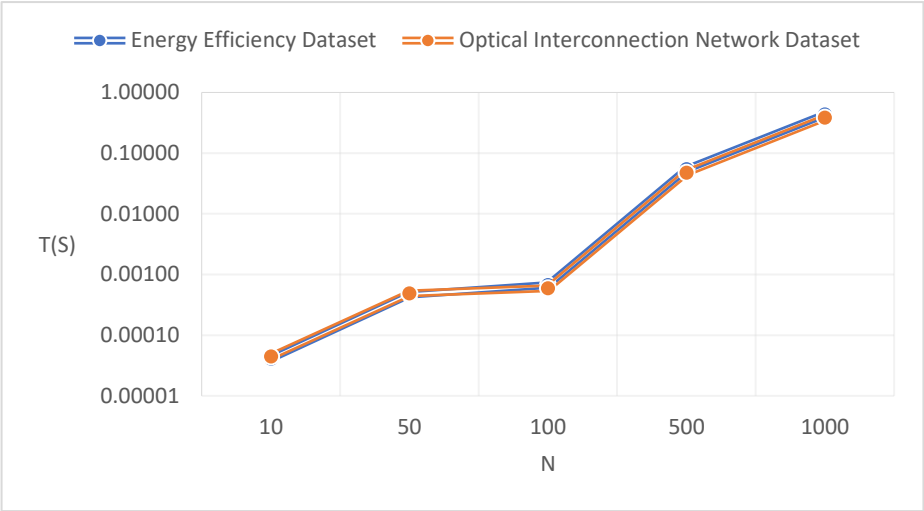


Figure 5.1 Experimental results for the T metric (T vs N)

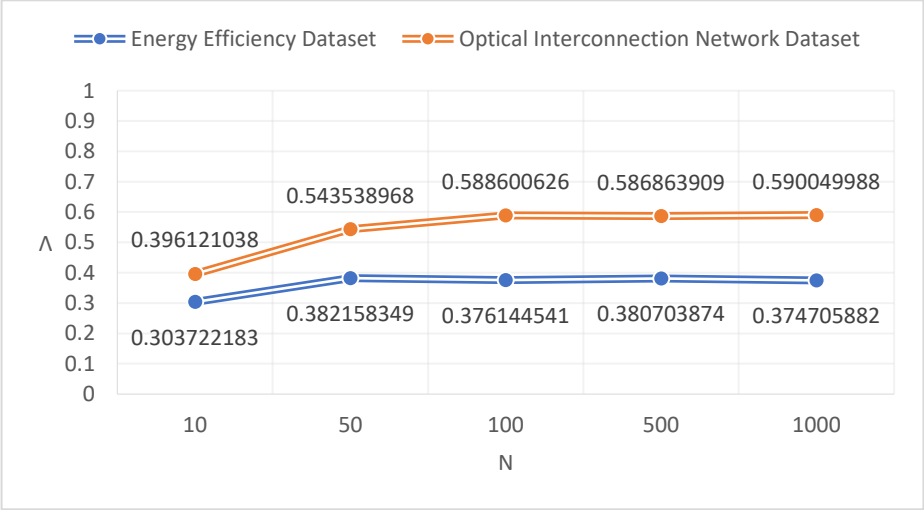


Figure 5.2 Experimental results for the Λ metric (Λ vs N)

In Figure 5.3, the experiment results regarding the Σ metric versus the number of queries/nodes N are shown. The graph depicts an almost straight line for the Σ outputs that represent either one of the load value datasets. The metric's behavior is quite similar to that of the Λ metric, since the algorithm achieves 1-1 query-node allocation.

Figures 5.4 and 5.5 demonstrate the results of our experiments for the Φ metric, each for one of the load datasets. Φ values are defined by Equation (7), hence Λ, Σ, Φ behave alike. Φ's results depend on the parameter α's value, as it defines different weights for Λ and Σ.

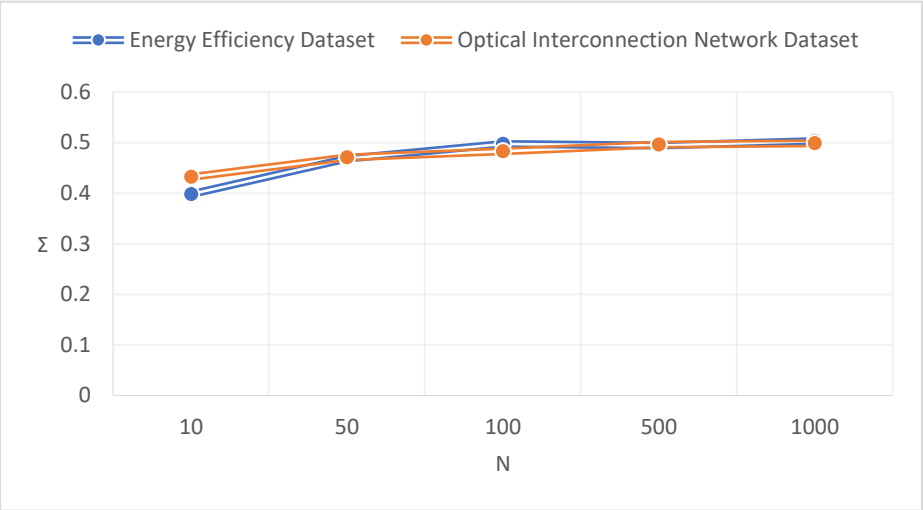


Figure 5.3 Experimental results for the Σ metric (Σ vs N)

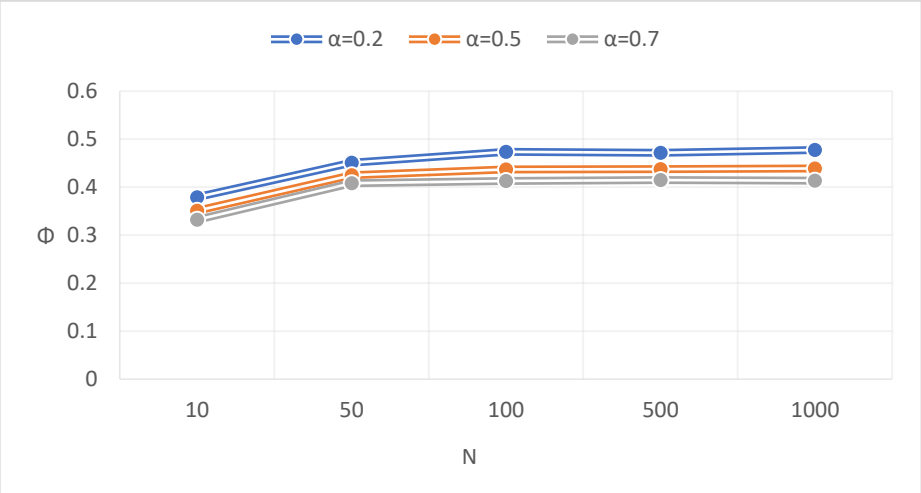


Figure 5.4 Experimental results of the Energy Efficiency Dataset for the Φ metric (Φ vs N)

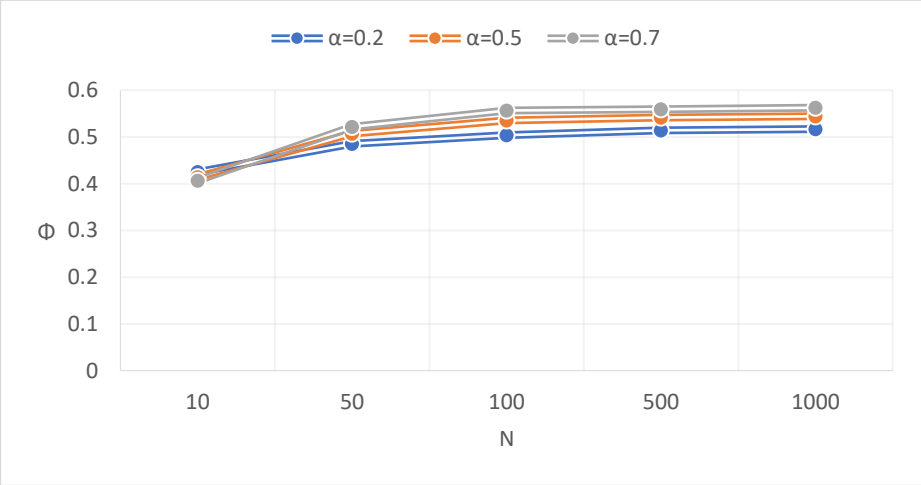


Figure 5.5 Experimental results of the Optical Interconnection Network Dataset for the Φ metric (Φ vs N)

5.2.2 The Clustering Task Allocation Algorithm

In Figure 5.6, we observe the minimum T value for $N = 10$ when $M = 10$. As M increases, the number of nodes allocated to each query's cluster increases as well. As a result, a single node among the candidates must be chosen to execute the query hence the T values get higher. The experimental results in Figure 5.7 show that for $M = 10$ when $N = 10$ the lowest value is attributed to the T metric, similarly to Figure 5.6, while for greater N values T increases. The latter event happens due to the fact that as the number of queries N gets higher, queries are divided into groups that are allocated to nodes, each group in a separate iteration. The more iterations we have, the more time is required for the completion of the allocations. Generally, as Figure 5.8 shows, when $N = M$ the T metric is minimized, when $N < M$ the QC must select the best node, among the cluster's candidates, for every query's execution resulting in higher T values and, lastly, when $N > M$ the number of iterations maximizes the T metric.

In Figure 5.9, we notice that the Λ metric's values tend to have small deviations among them for both load datasets. In Figure 5.10, a 3D graph depicts the Λ values of the Energy Efficiency dataset which range in the interval $[0.29, 0.43]$. It appears that as long as $M > N$, which indicates that the nodes outnumber the queries, every query selects the most appropriate candidate among the nodes which are assigned to its cluster. In contrast, in case the number of queries N is high and the number of nodes M is low, the Λ metric maximizes. Specifically, in each iteration, queries with the highest complexities choose nodes with low loads (for their clusters), rendering those nodes unavailable until the queries' completion. As a result, the remaining free nodes, whose load values are high, are repeatedly chosen by low-complexity queries leading to the maximization of the average Λ metric. In Figure 5.11, a 3D graph shows the Λ values of the Optical Interconnection Network dataset which range in the interval $[0.37, 0.61]$. In cases where N values are small, the Λ results are also low due to the wide variety of nodes for every query to choose from. Furthermore, Λ values are observed to be low when $M = 10$, due to the randomness of the load realizations without giving the full picture on the load of the available nodes especially when their number is low.

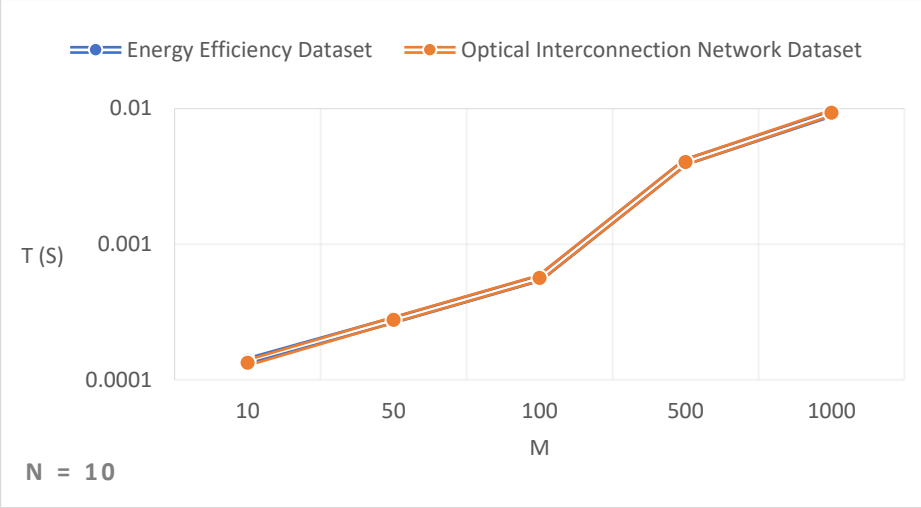


Figure 5.6 Experimental results for the T metric (T vs M) when N = 10

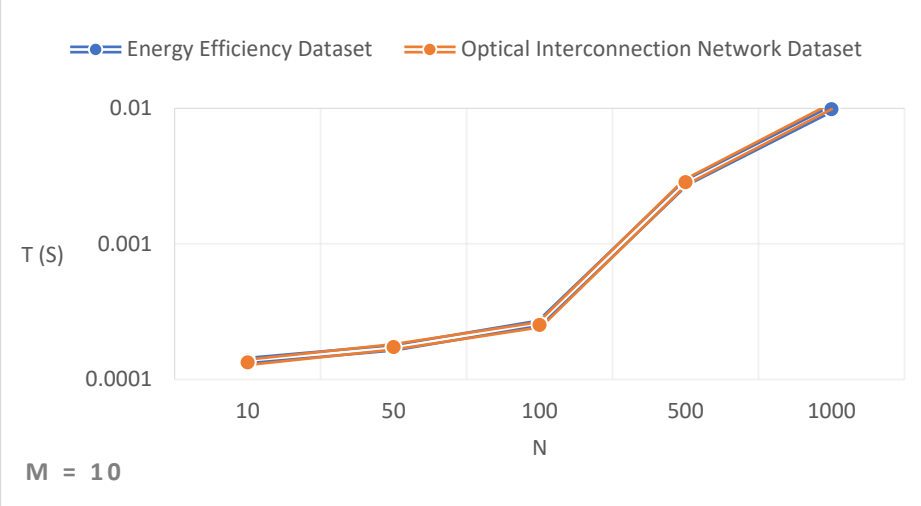


Figure 5.7 Experimental results for the T metric (T vs N) when M = 10

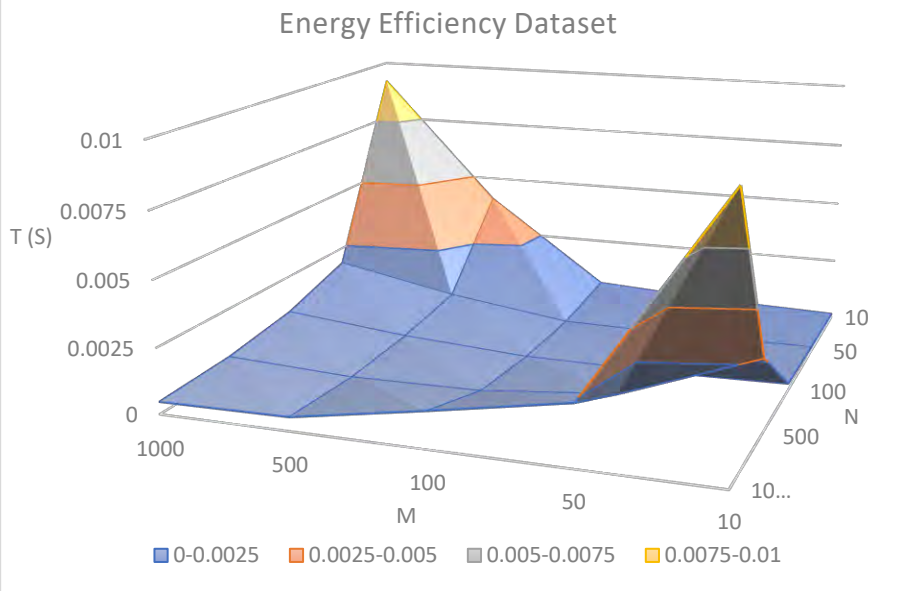


Figure 5.8 A 3D point of view on the T metric regarding the Energy Efficiency Dataset

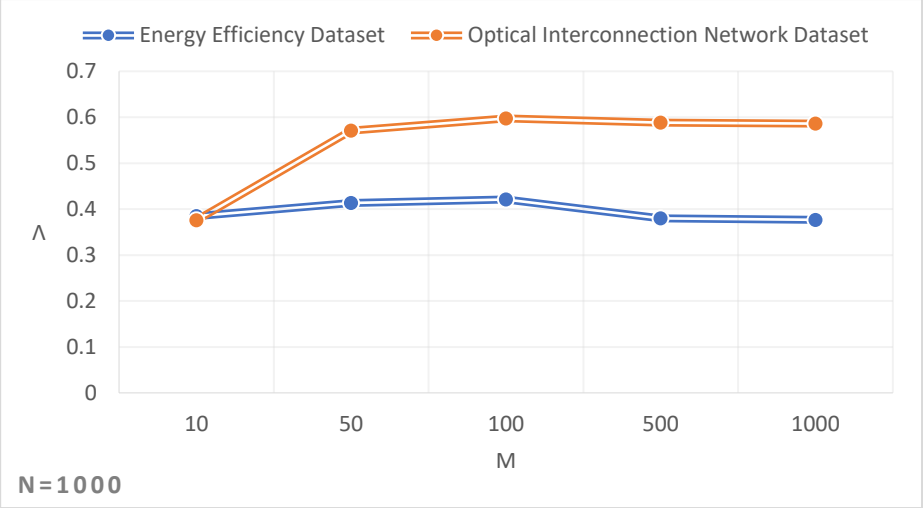


Figure 5.9 Experimental results for the Λ metric (Λ vs M) when N = 1000

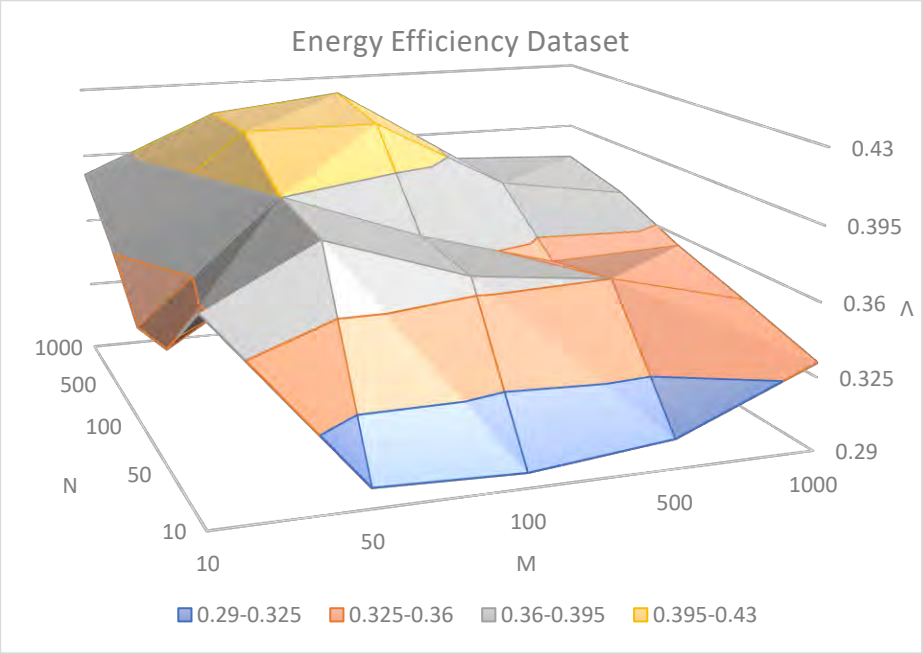


Figure 5.10 A 3D point of view on the Λ metric regarding the Energy Efficiency Dataset

Regarding the Σ metric, in Figure 5.12 we observe that there are no major differences between the two datasets when $N = 1000$. Figures 5.13 and 5.14 show that for both datasets the Σ values are low when the number of nodes is low. This is explained by the fact that, after the first iteration of the allocation process, the fastest nodes are mainly available to execute the next set of queries, leading to the reduction of Σ. In other cases, Σ increases because the selection of a node in a query's cluster depends on the node's load and speed and the query's complexity, as defined in $Function\ (2)$ of Chapter 4, and not solely on the node's speed.

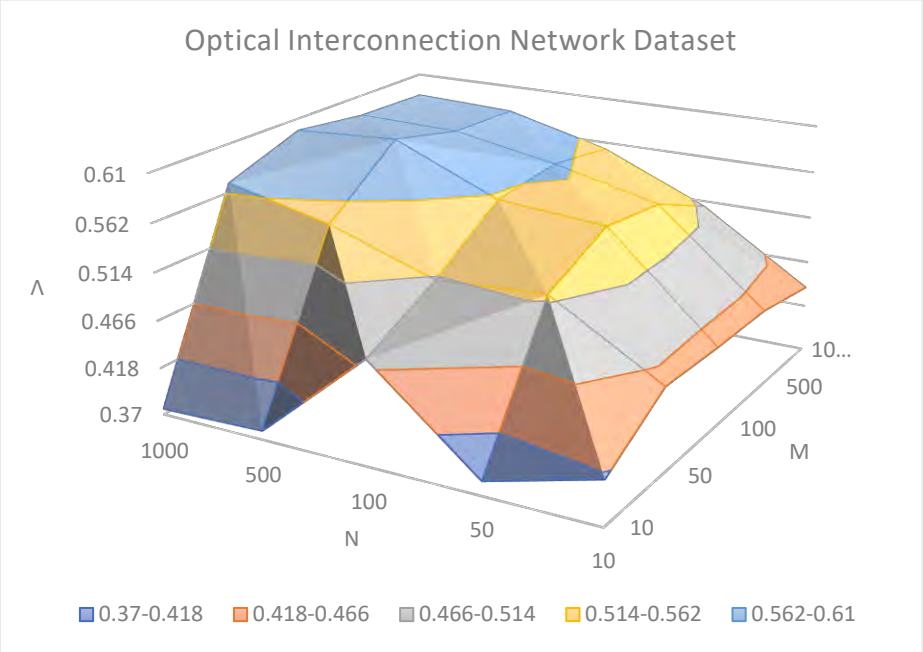


Figure 5.11 A 3D point of view on the Λ metric regarding the Optical Interconnection Network Dataset

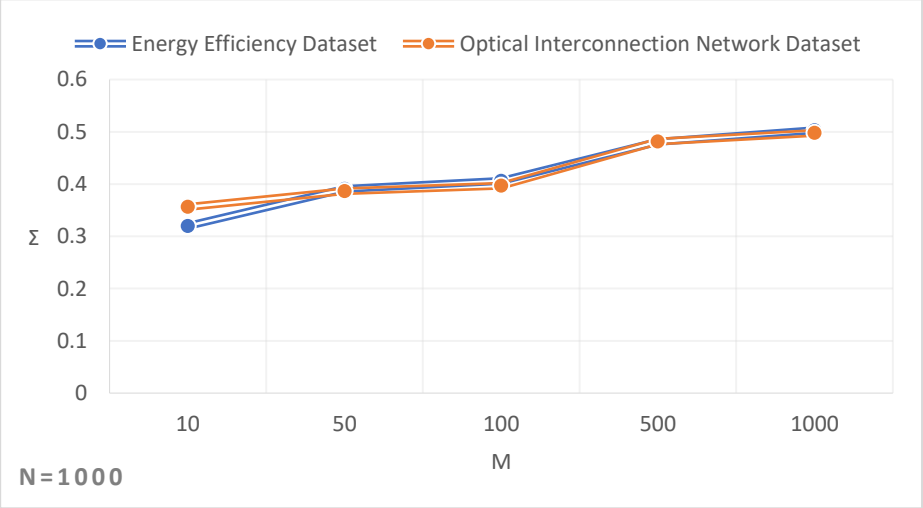


Figure 5.12 Experimental results for the Σ metric (Σ vs M) when N = 1000

Figure 5.15 shows the Φ metric's results for both of the datasets when $N = 10$ and for different α values. The best performance of the Φ metric is achieved when $\alpha = 0.7$, due to Λ's better outputs. Figures 5.16 and 5.17 demonstrate the results of our experiments for the Φ metric, each for one of the load datasets. The Φ values are defined by $Equation\ (7)$, hence Φ's behavior is influenced by Λ and Σ.

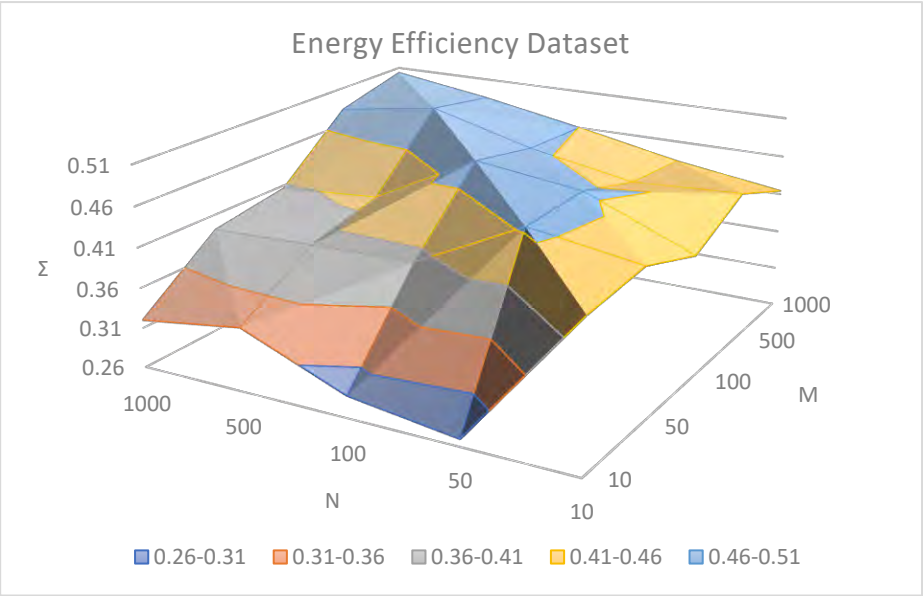


Figure 5.13 A 3D point of view on the Σ metric regarding the Energy Efficiency Dataset

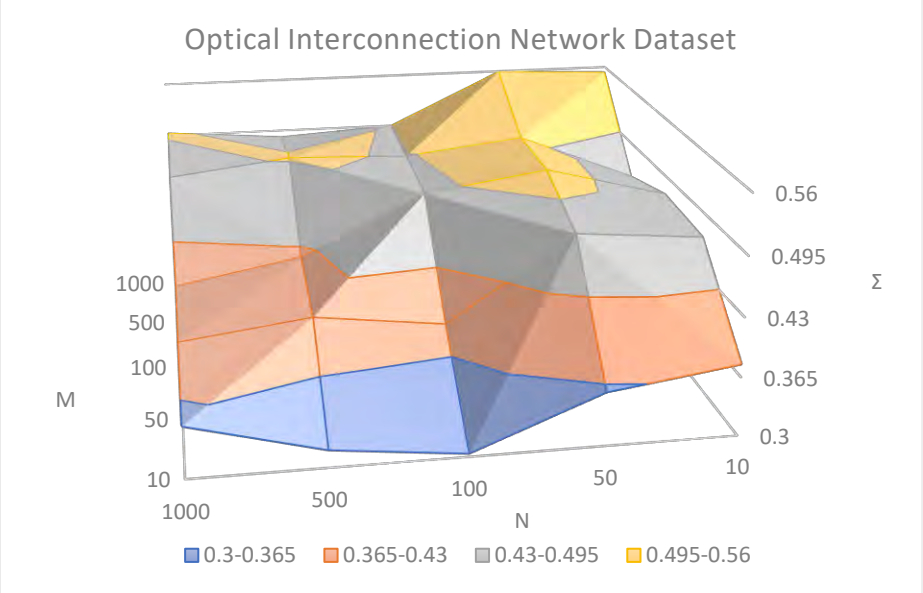


Figure 5.14 A 3D point of view on the Σ metric regarding the Optical Interconnection Network Dataset

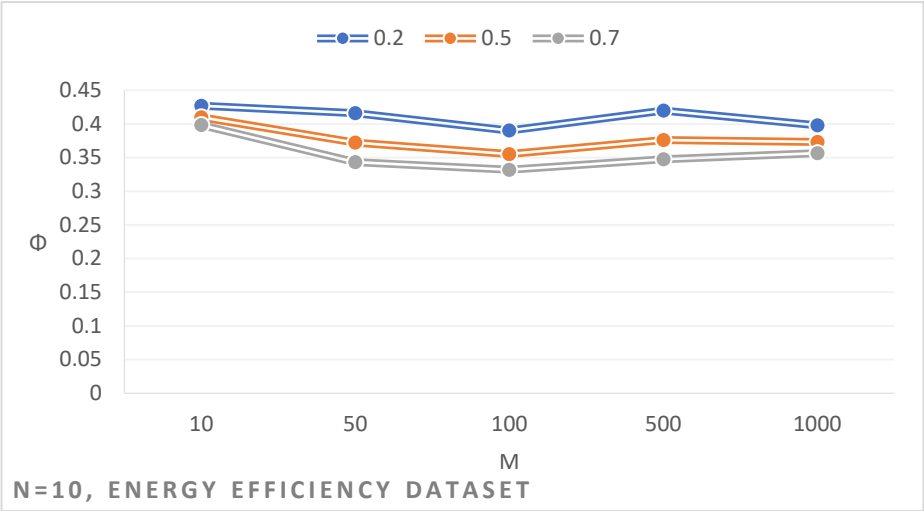


Figure 5.15 Experimental results for the Φ metric (Φ vs M) when N = 10 for the Energy Efficiency Dataset

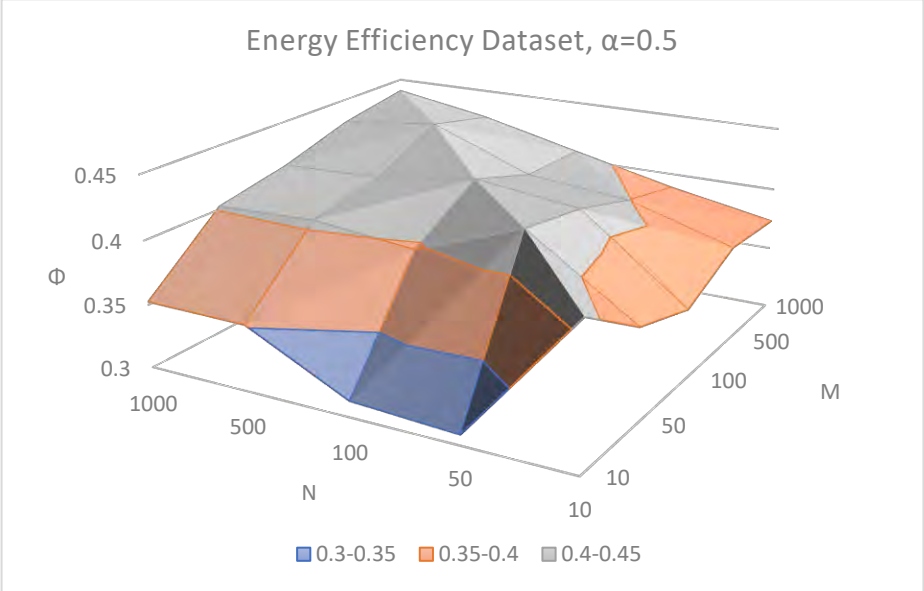


Figure 5.16 A 3D point of view on the Φ metric regarding the Energy Efficiency Dataset for α = 0.5

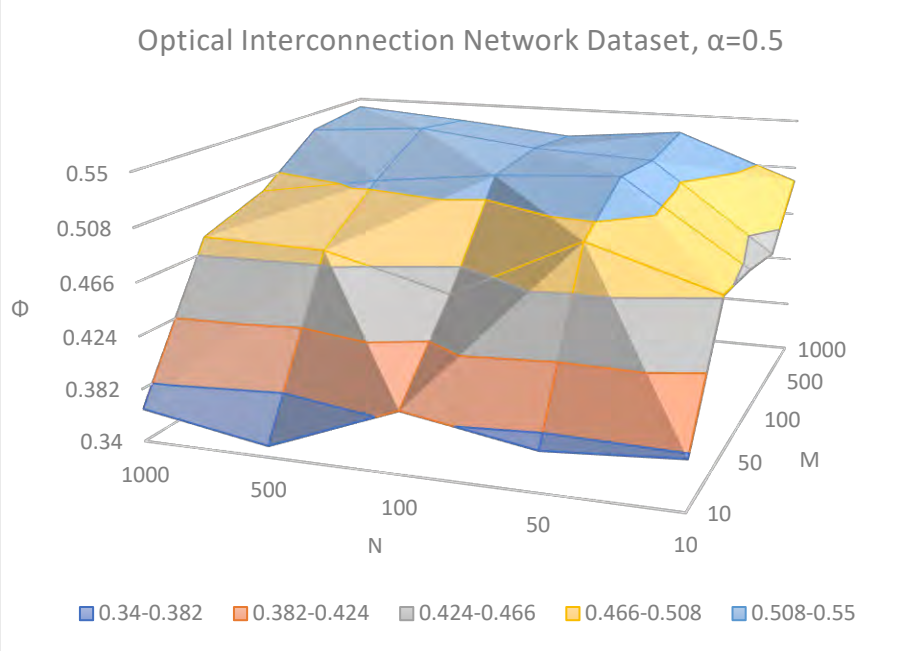


Figure 5.17 A 3D point of view on the Φ metric regarding the Optical Interconnection Network Dataset for α = 0.5

5.2.3 The Simplified Swarm Optimization Algorithm

In Figures 5.18 and 5.19, we observe that when the number of queries N is low and, especially, when the number of nodes is high the T metric is maximized. This is caused by the fact that for every node and every candidate solution a set of constraints have to be calculated according to the SSO-TA Algorithm. For high N values the T metric is minimized, since the total allocation time required is divided by the number N to produce T.

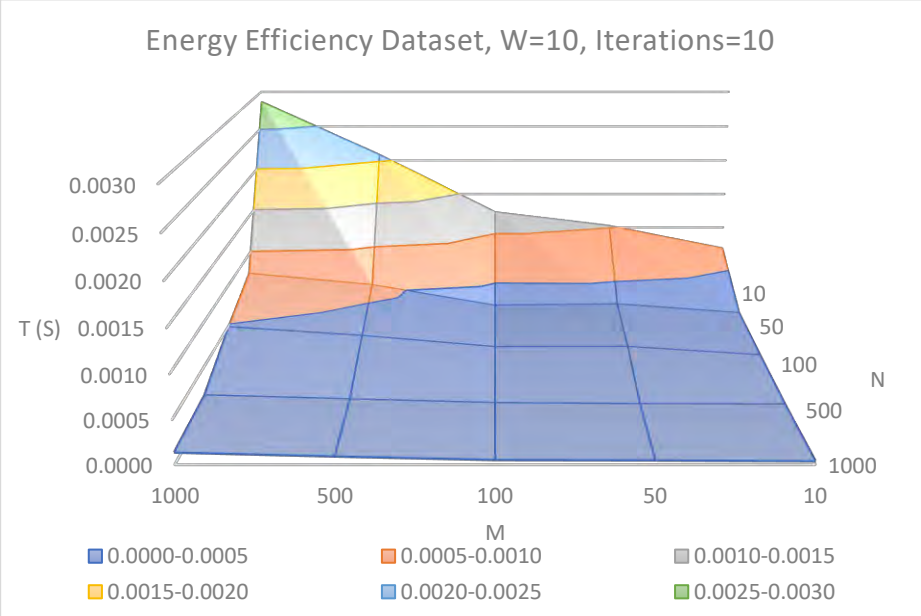


Figure 5.18 A 3D point of view on the T metric (M vs N vs T) regarding the Energy Efficiency Dataset when W = 10 & Iterations = 10

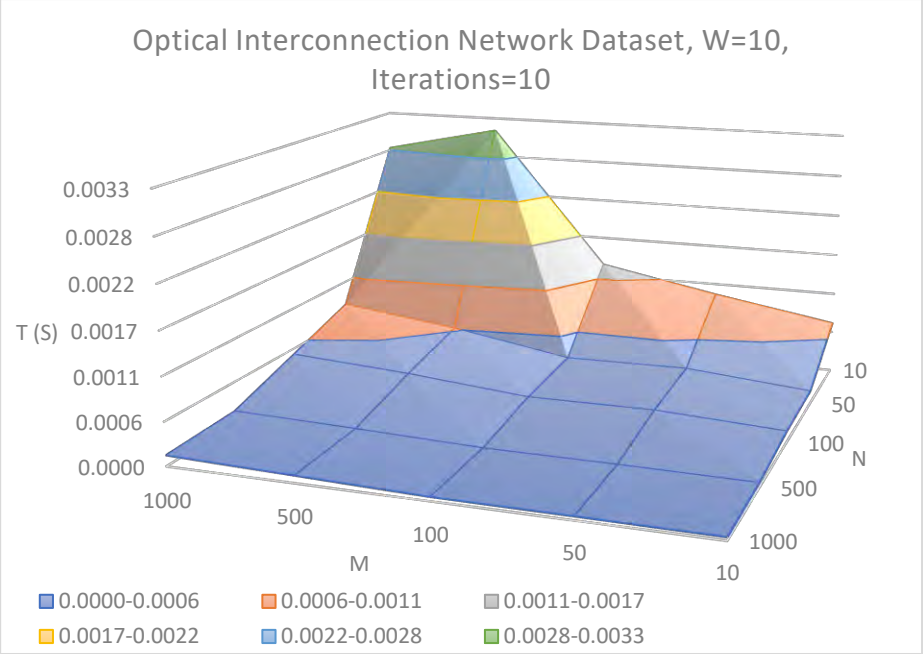


Figure 5.19 A 3D point of view on the T metric (M vs N vs T) regarding the Optical Interconnection Network Dataset when W = 10 & Iterations = 10

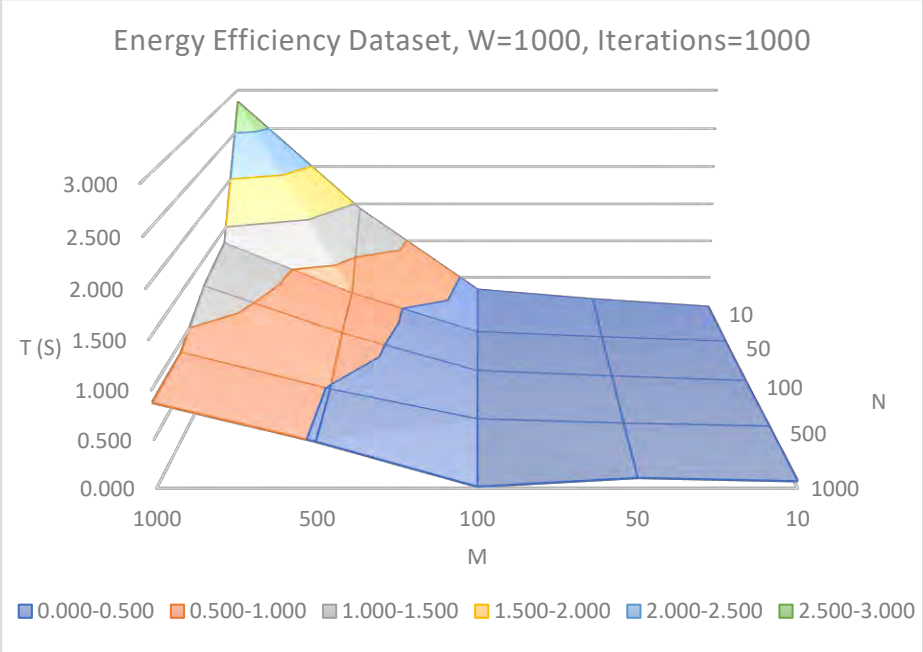


Figure 5.20 A 3D point of view on the T metric (M vs N vs T) regarding the Energy Efficiency Dataset when W = 1000 & Iterations = 1000

In Figure 5.20, we notice that for high M values and, especially, when N is low T is also maximized for the same reasons presented for Figures 5.18 and 5.19. Figures 5.21 and 5.22 show that when the number of queries N and nodes M are invariant, the T metric is defined by a linear function of the number of the candidate solutions W and the number of $Iterations$. The highest T value appears when $W = Iterations = 1000$.

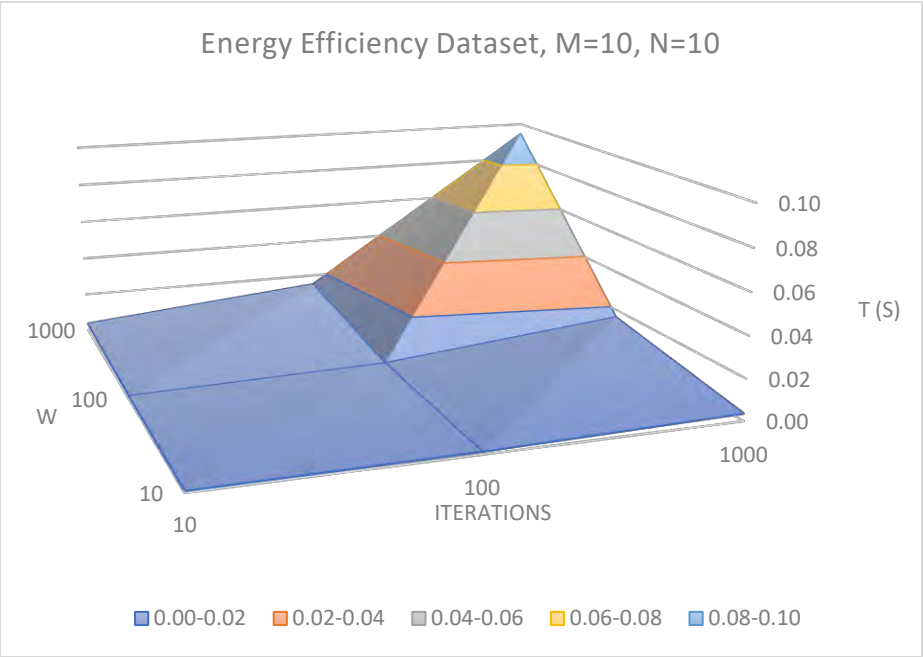


Figure 5.21 A 3D point of view on the T metric (W vs Iterations vs T) regarding the Energy Efficiency Dataset when M = 10 & N= 10

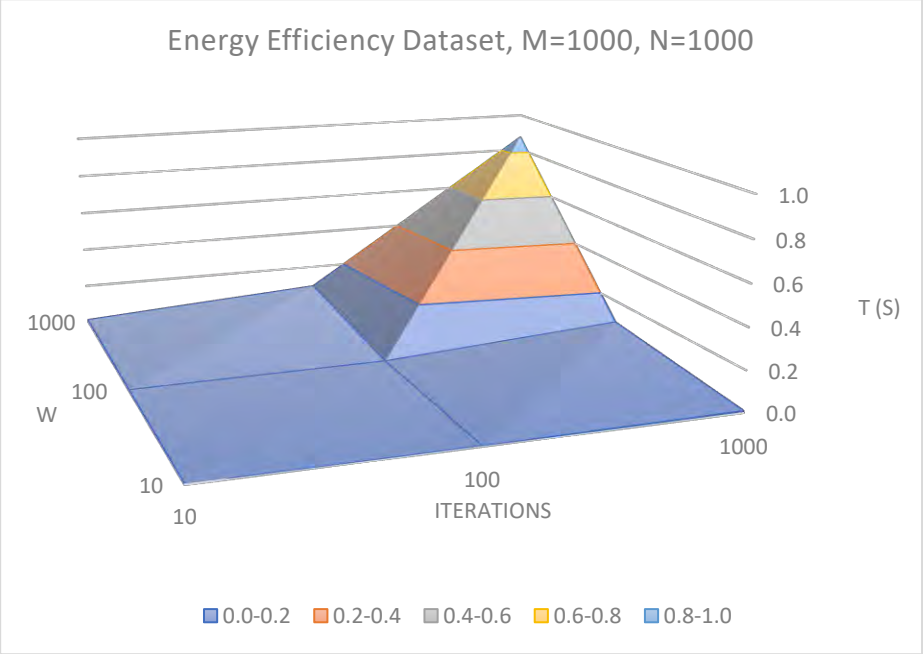


Figure 5.22 A 3D point of view on the T metric (W vs Iterations vs T) regarding the Energy Efficiency Dataset when M = 1000 & N = 1000

In Figures 5.23 through 5.26, we observe that for $W = Iterations = 10$ as well as for $W = Iterations = 1000$ the Λ metric's behavior is as follows. When the number of nodes M is small, on most occasions, it appears to have its lowest values. This is natural, since as M increases the probability of finding the most appropriate node for every query decreases, because the probability distribution is uniform, and the candidate solutions are generated and changed randomly. Furthermore, in Figures 5.27 through 5.30 we notice that as the

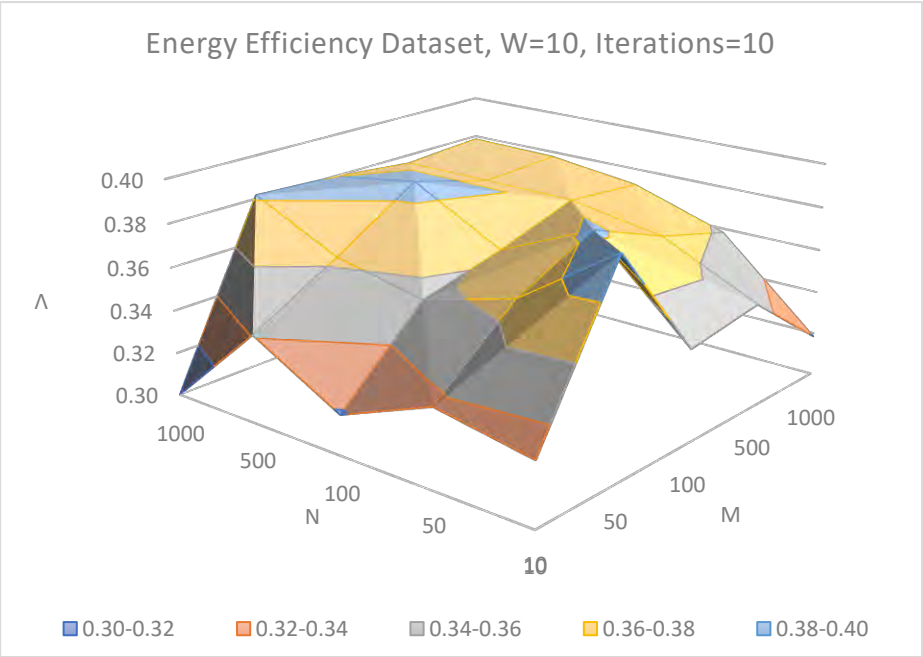


Figure 5.23 A 3D point of view on the Λ metric (N vs M vs Λ) regarding the Energy Efficiency Dataset when W = 10 & Iterations = 10

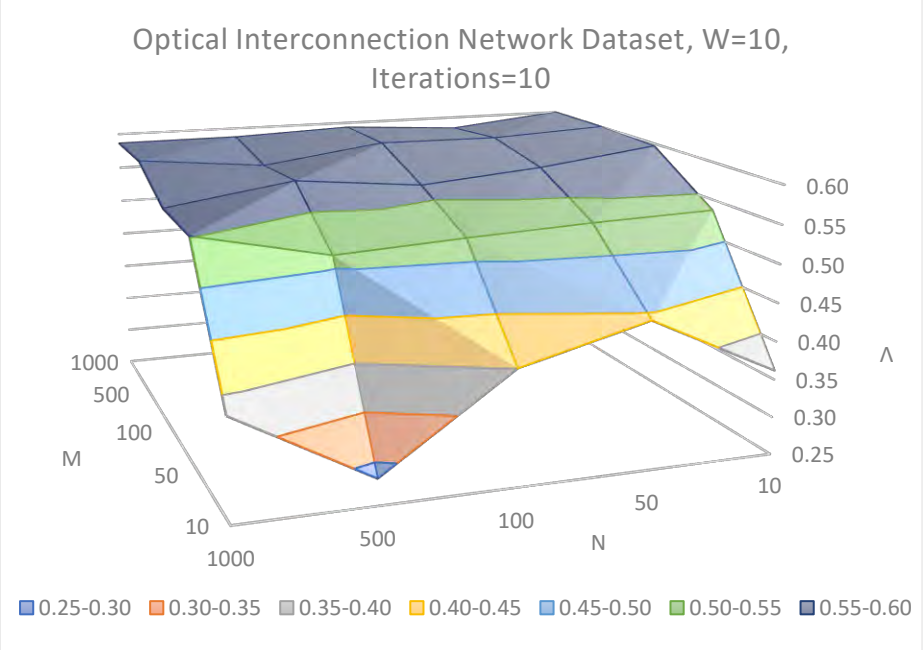


Figure 5.24 A 3D point of view on the Λ metric (N vs M vs Λ) regarding the Optical Interconnection Network Dataset when W = 10 & Iterations = 10

number of solutions W and the number of $Iterations$ increase, the Λ values, for the most part, decrease.

As far as the Σ metric —which depicts the average difference between the selected node's speed and the minimum node speed— is concerned, Figures 5.31 and 5.32 show that when $W = Iterations = 10$ for a small number of nodes M and, especially, for a small number of queries N the Σ values decrease. When M is low, as explained previously

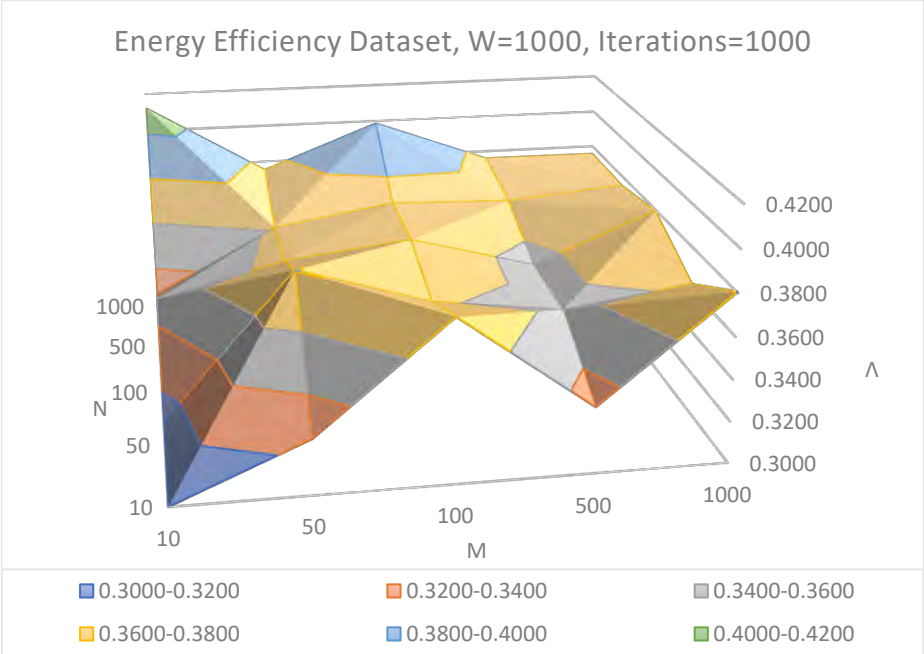


Figure 5.25 A 3D point of view on the Λ metric (N vs M vs Λ) regarding the Energy Efficiency Dataset when W = 1000 & Iterations = 1000

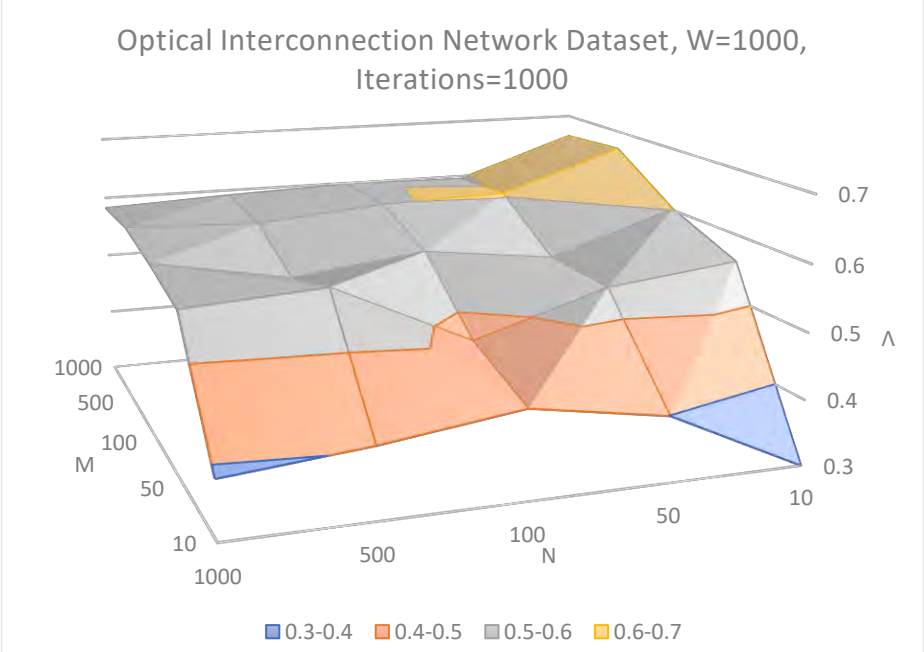


Figure 5.26 A 3D point of view on the Λ metric (N vs M vs Λ) regarding the Optical Interconnection Network Dataset when W = 1000 & Iterations = 1000

for the Λ metric, the probability of generating the appropriate solution increases. On the other hand, when N is low, the queries complexity and deadline requirements don't exceed the nodes' load and speed limits, thus leading to better allocations and, by extension, low Σ values. Figures 5.33 and 5.34 demonstrate Λ's behavior for $W = Iterations = 1000$. As long as M is low and N is high, Λ's values are increased due to the fact that the queries' requirements exceed the nodes' capacities which leads to inaccurate

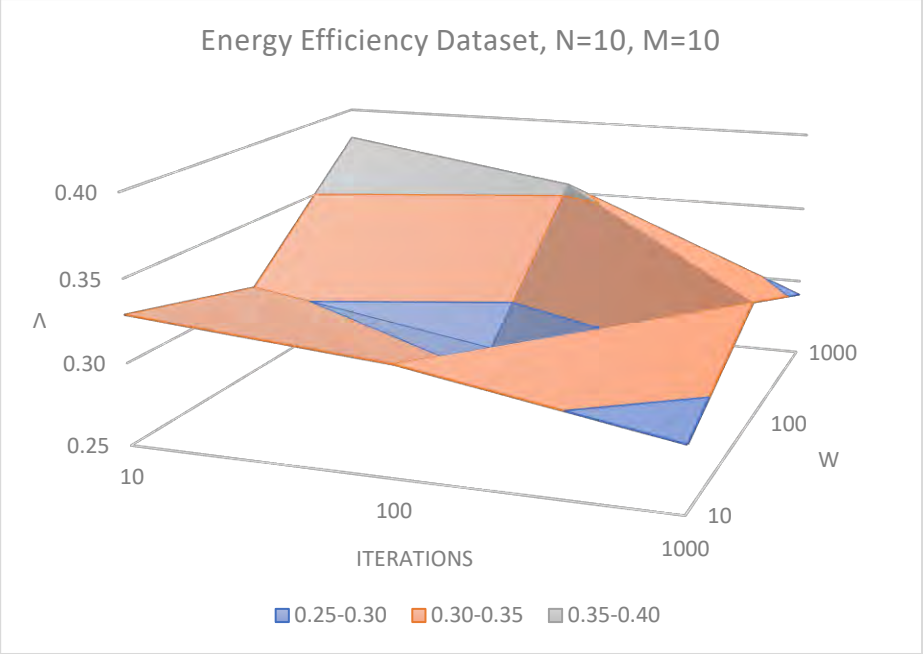


Figure 5.27 A 3D point of view on the Λ metric (W vs Iterations vs Λ) regarding the Energy Efficiency Dataset when N = 10 & M= 10

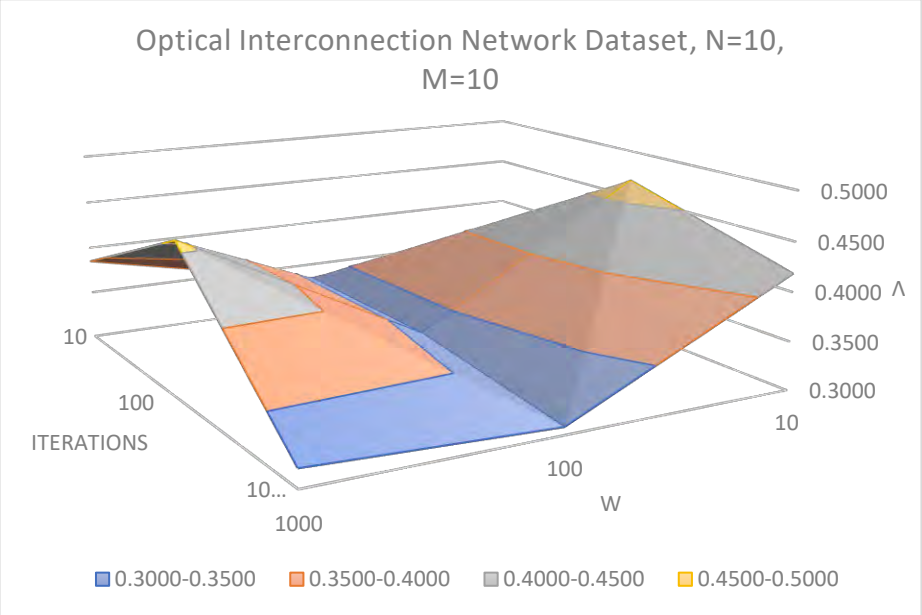


Figure 5.28 A 3D point of view on the Λ metric (W vs Iterations vs Λ) regarding the Optical Interconnection Network Dataset when N = 10 &M= 10

allocations. In Figures 5.35 through 5.38, the experimental results for a specific number of queries and nodes show no consistency among each other, since the SSO-TA Algorithm functions mostly based on randomly generated solutions. Regardless of the number of $Iterations$ and the number of solutions W, there is no guarantee that an appropriate solution will be generated and chosen, a matter which can be researched at length in future work.

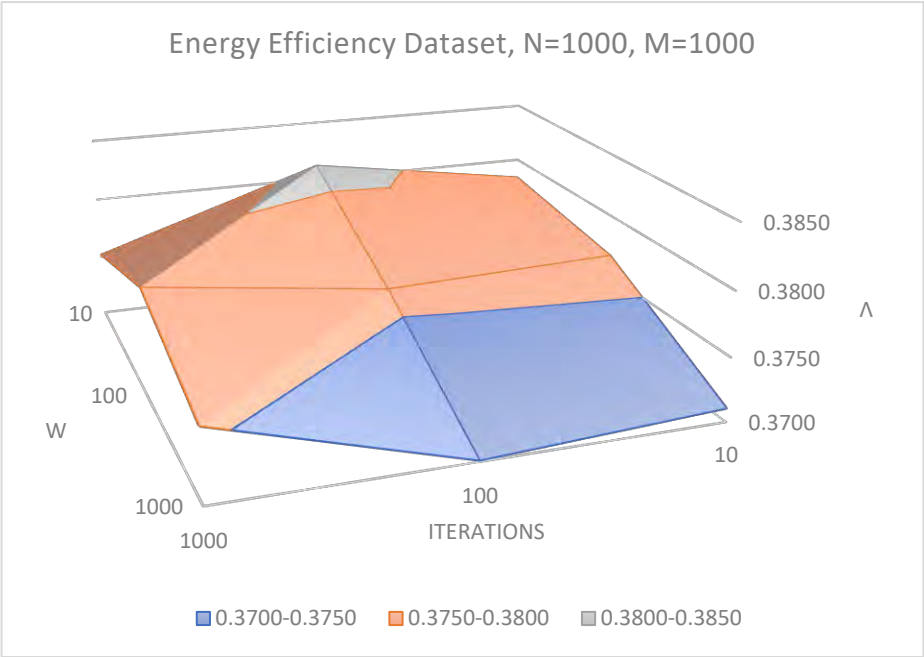


Figure 5.29 A 3D point of view on the Λ metric (W vs Iterations vs Λ) regarding the Energy Efficiency Dataset when N = 1000 & M= 1000

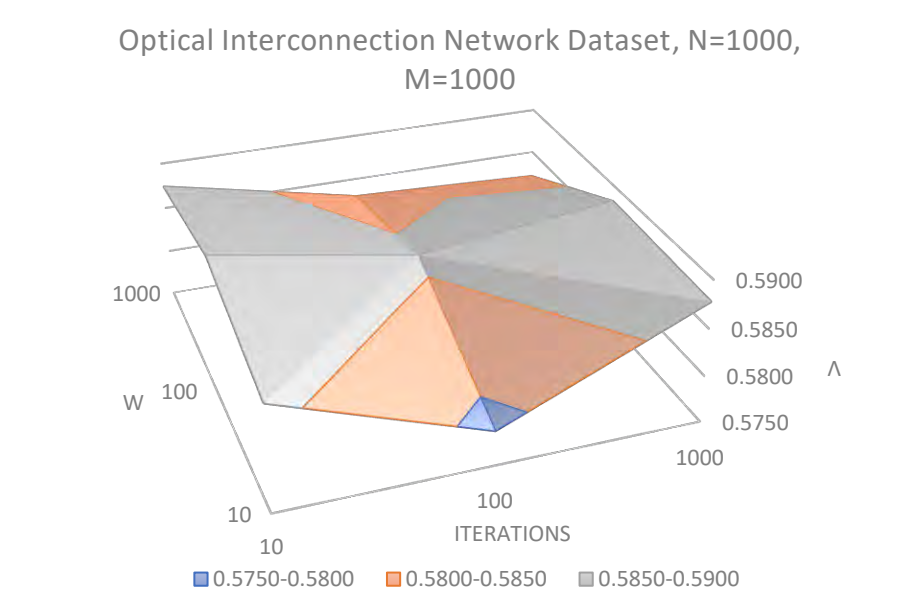


Figure 5.30 A 3D point of view on the Λ metric (W vs Iterations vs Λ) regarding the Optical Interconnection Network Dataset when N = 1000 & M= 1000

Figures 5.39 through 5.46 demonstrate the results of our experiments for the Φ metric, for both of the load datasets. Φ values are defined by $Equation\ (7)$, hence Φ's behavior is similar to that of Λ and Σ's.

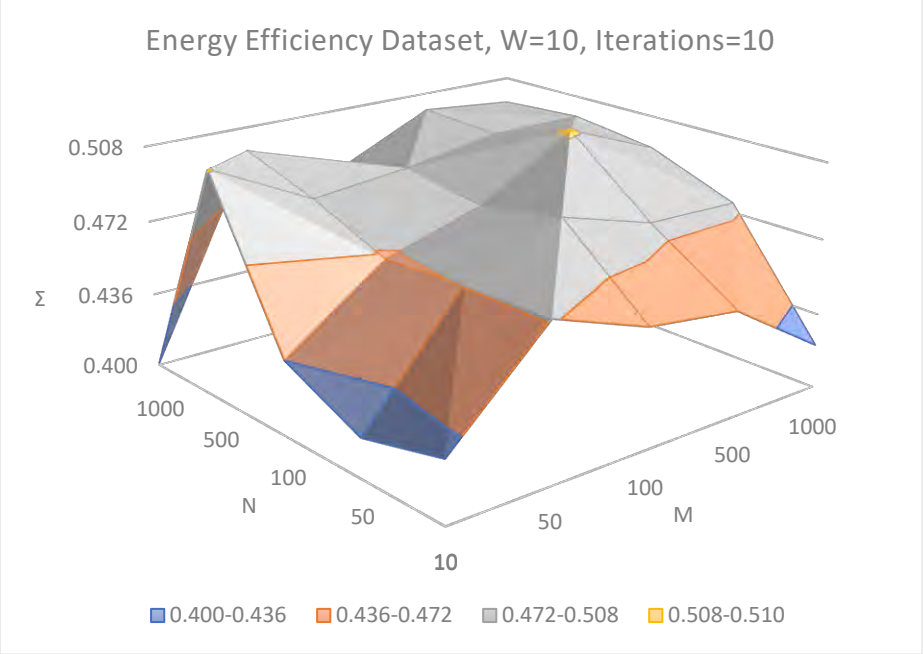


Figure 5.31 A 3D point of view on the Σ metric (N vs M vs Σ) regarding the Energy Efficiency Dataset when W = 10 & Iterations = 10

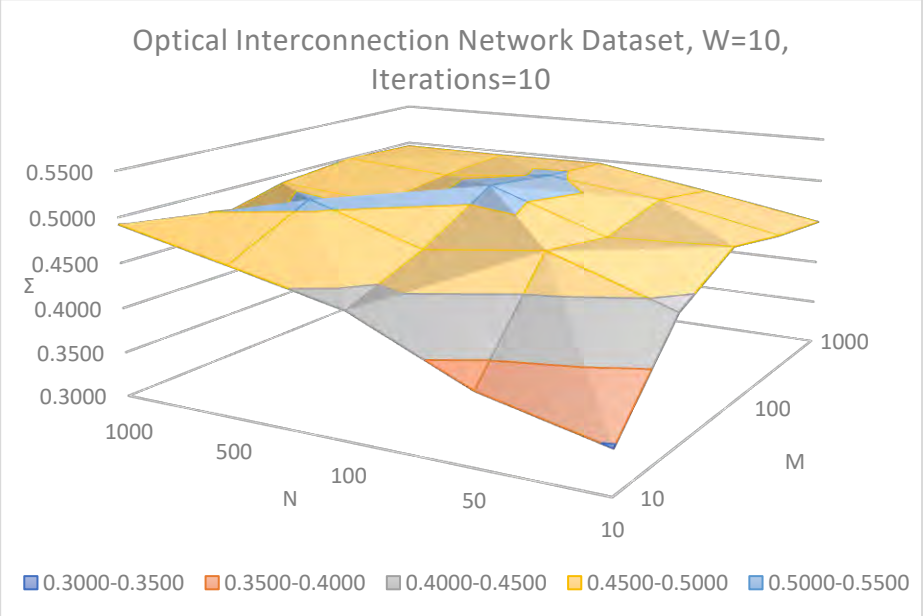


Figure 5.32 A 3D point of view on the Σ metric (N vs M vs Σ) regarding the Optical Interconnection Network Dataset when W = 10 & Iterations = 10

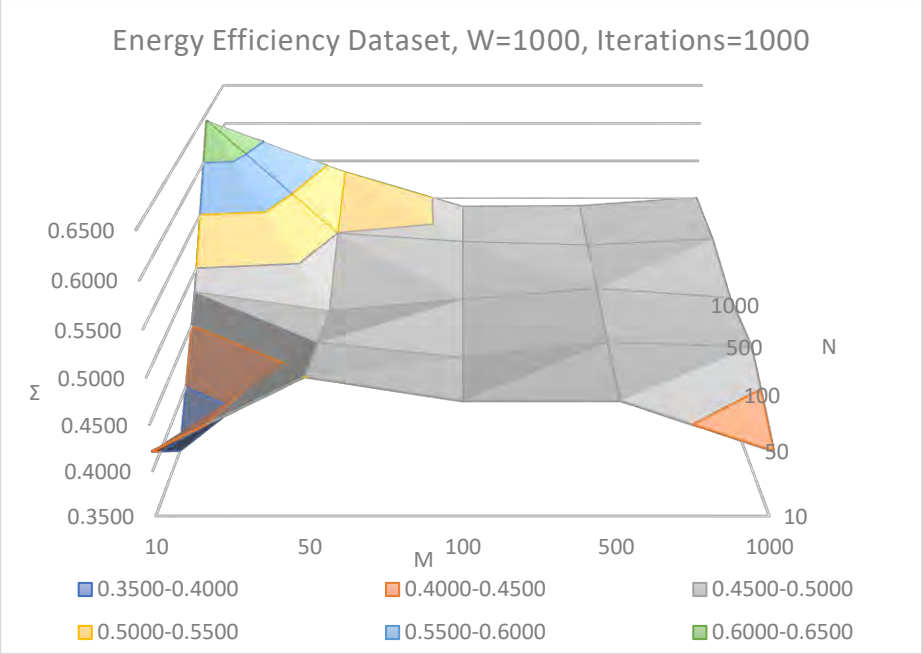


Figure 5.33 A 3D point of view on the Σ metric (N vs M vs Σ) regarding the Energy Efficiency Dataset when W = 1000 & Iterations = 1000

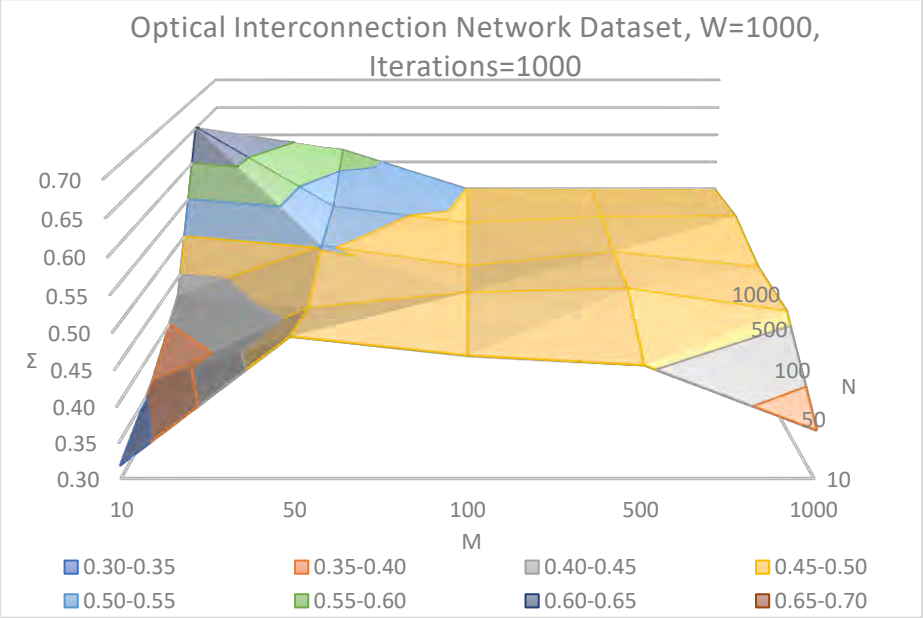


Figure 5.34 A 3D point of view on the Σ metric (N vs M vs Σ) regarding the Optical Interconnection Network Dataset when W = 1000 & Iterations = 1000

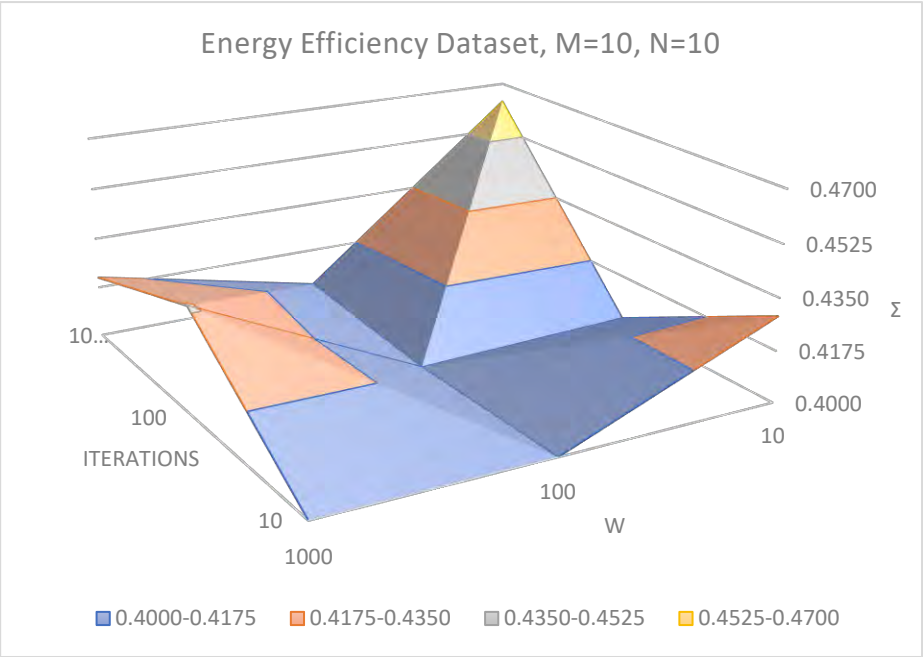


Figure 5.35 A 3D point of view on the Σ metric (W vs Iterations vs Σ) regarding the Energy Efficiency Dataset when N = 10 & M= 10

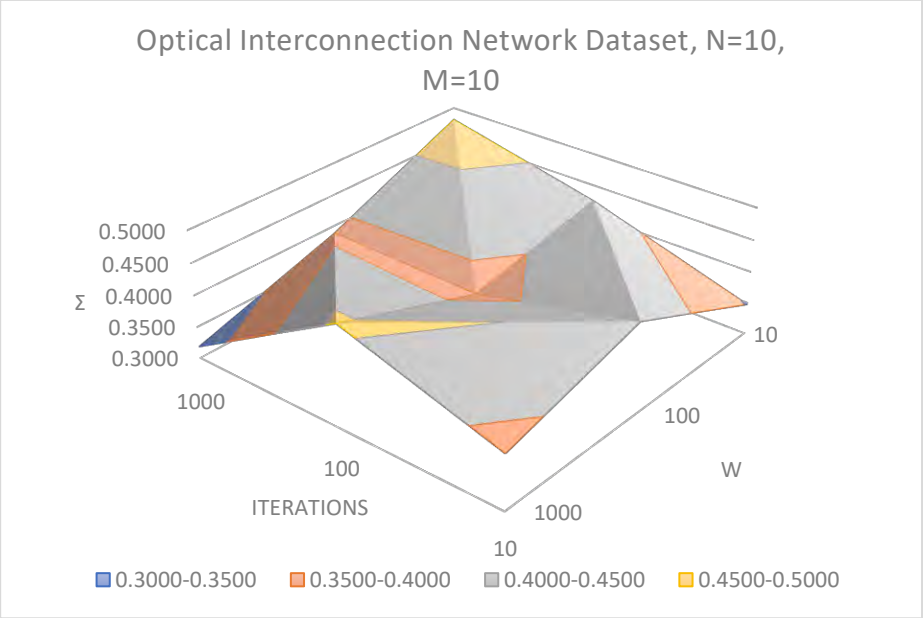


Figure 5.36 A 3D point of view on the Σ metric (W vs Iterations vs Σ) regarding the Optical Interconnection Network Dataset when N = 10 & M= 10

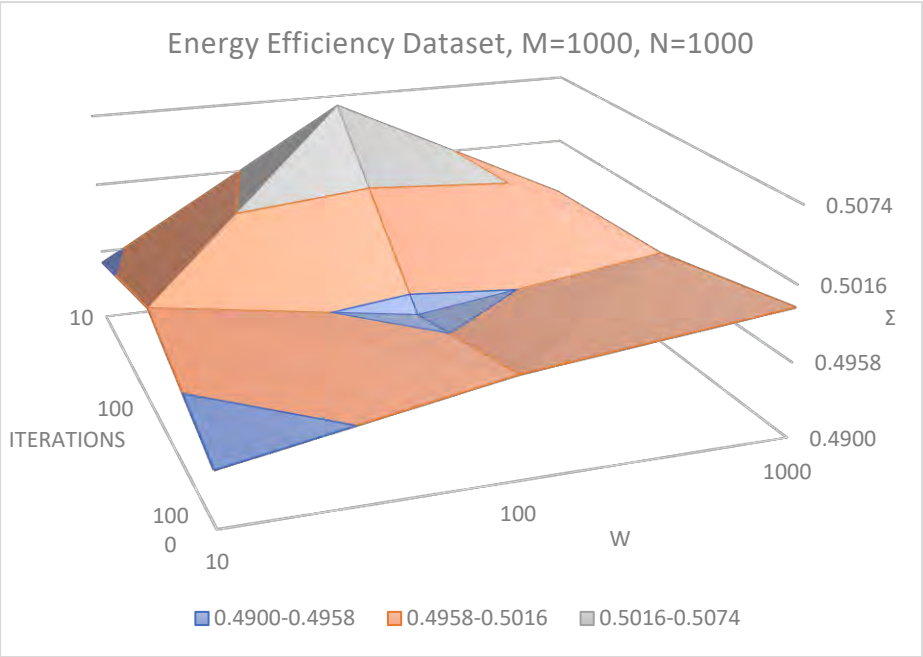


Figure 5.37 A 3D point of view on the Σ metric (W vs Iterations vs Σ) regarding the Energy Efficiency Dataset when N = 1000 & M= 1000

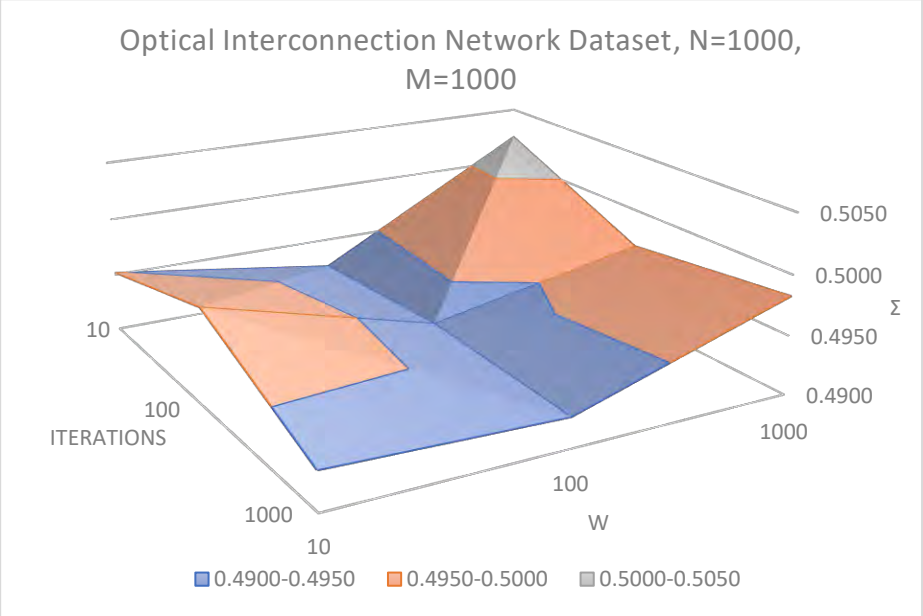


Figure 5.38 A 3D point of view on the Σ metric (W vs Iterations vs Σ) regarding the Optical Interconnection Network Dataset when N = 1000 & M= 1000

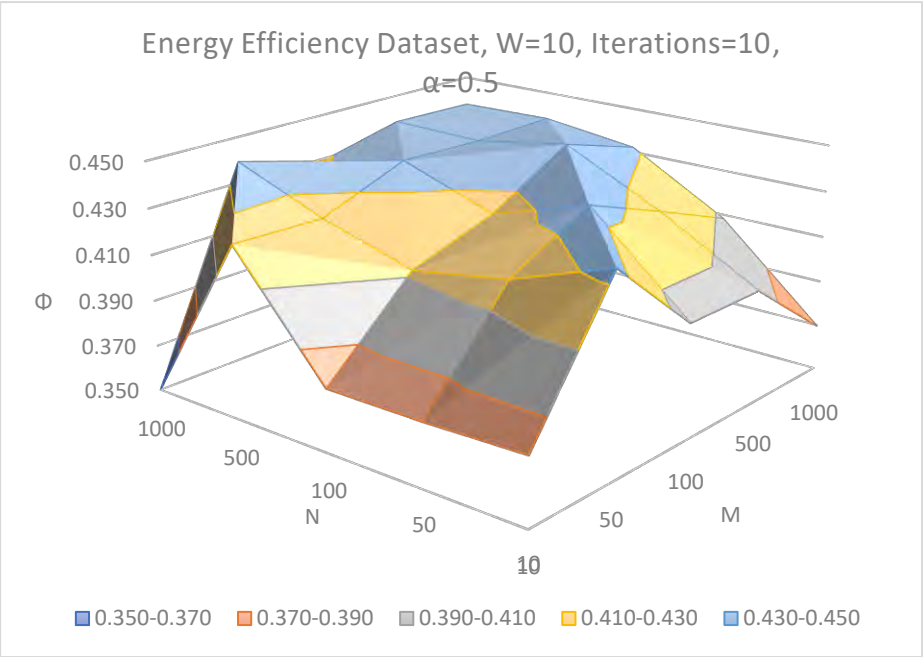


Figure 5.39 A 3D point of view on the Φ metric (N vs M vs Φ) regarding the Energy Efficiency Dataset when W = 10, Iterations = 10 & α = 0.5

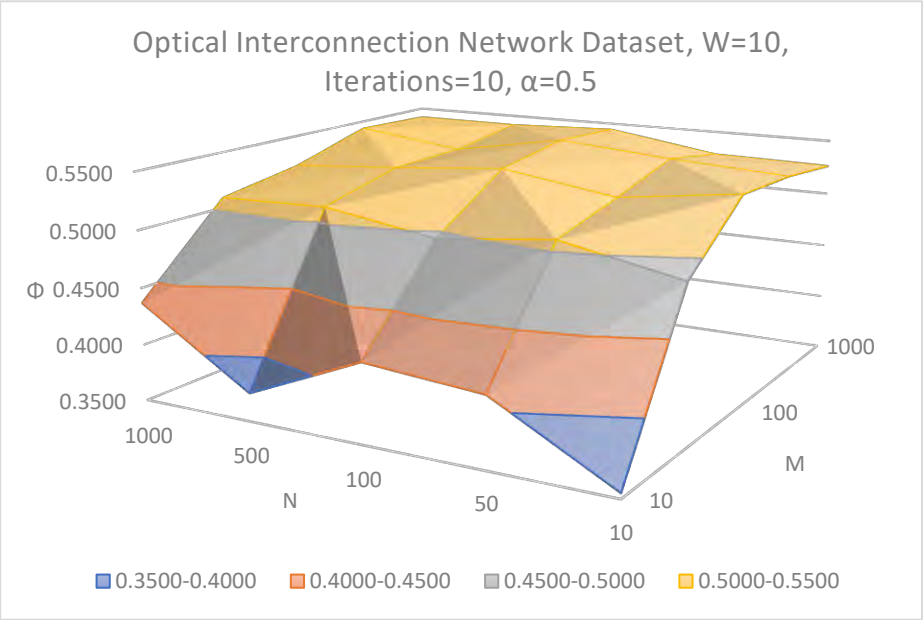


Figure 5.40 A 3D point of view on the Φ metric (N vs M vs Φ) regarding the Optical Interconnection Network Dataset when W = 10, Iterations = 10 & α = 0.5

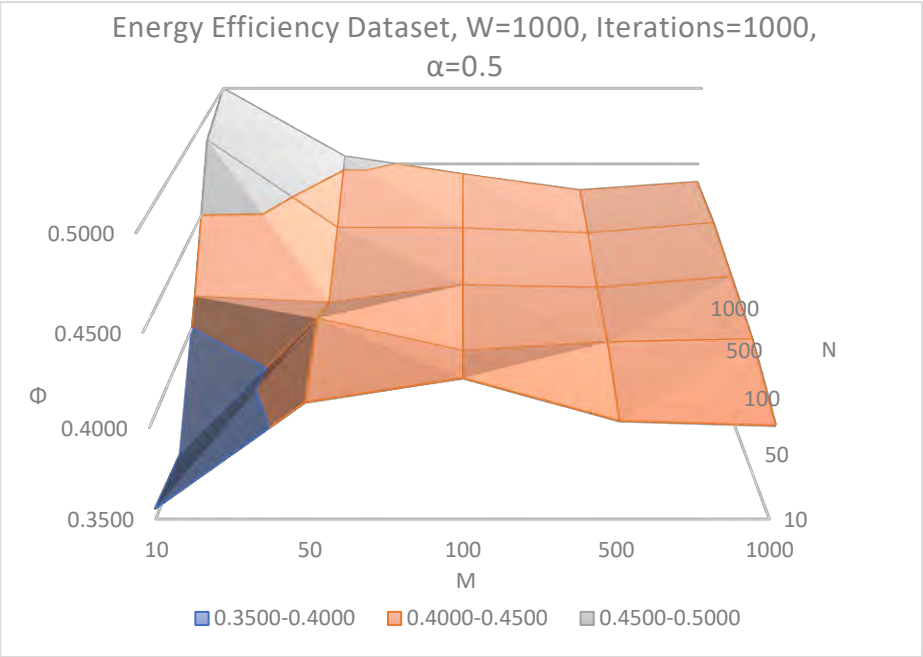


Figure 5.41 A 3D point of view on the Φ metric (N vs M vs Φ) regarding the Energy Efficiency Dataset when W = 1000, Iterations = 1000 & α = 0.5

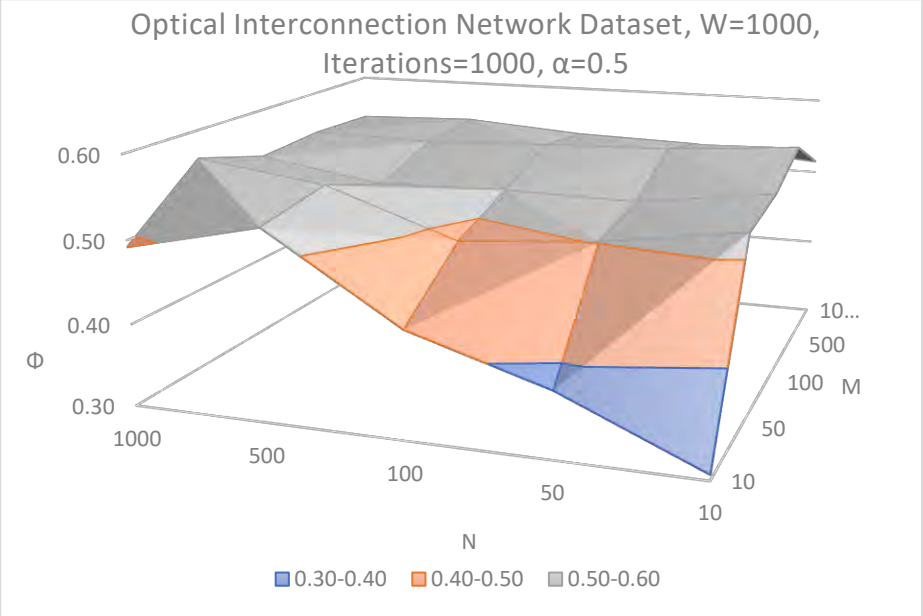


Figure 5.42 A 3D point of view on the Φ metric (N vs M vs Φ) regarding the Optical Interconnection Network Dataset when W = 1000, Iterations = 1000 & α = 0.5

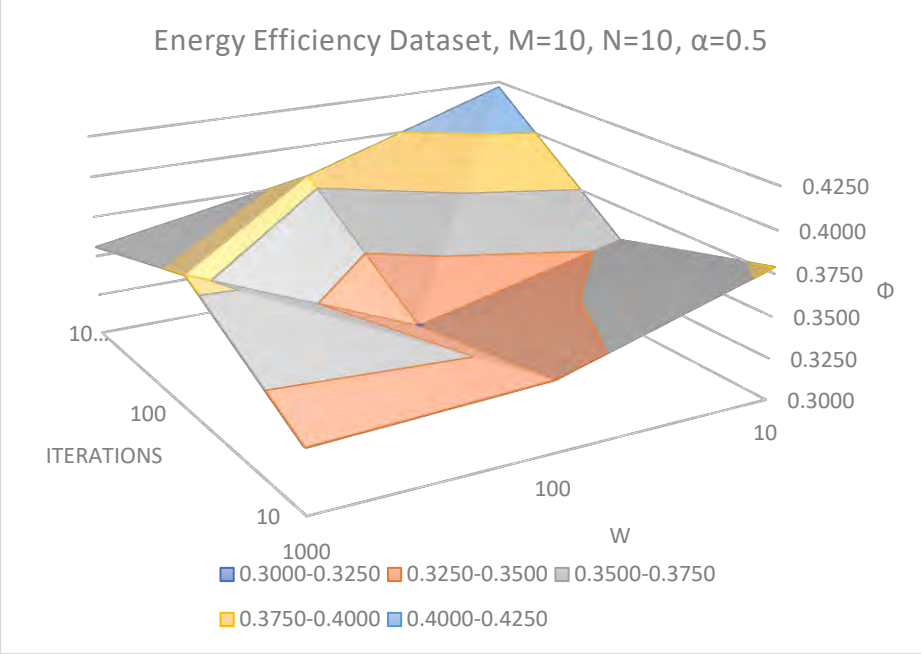


Figure 5.43 A 3D point of view on the Φ metric (W vs Iterations vs Φ) regarding the Energy Efficiency Dataset when N = 10, M= 10 & α = 0.5

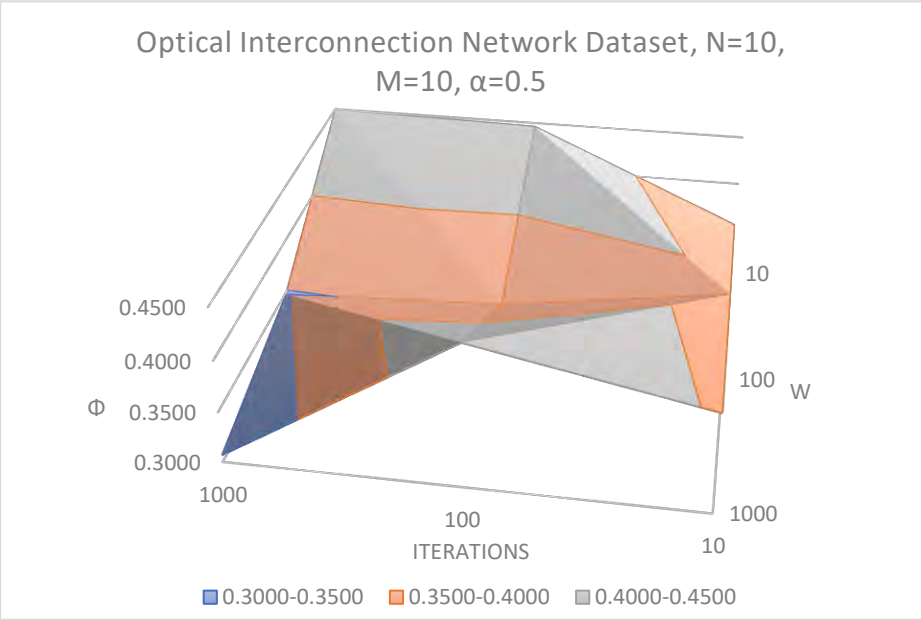


Figure 5.44 A 3D point of view on the Φ metric (W vs Iterations vs Φ) regarding the Optical Interconnection Network Dataset when N = 10, M= 10 & α = 0.5

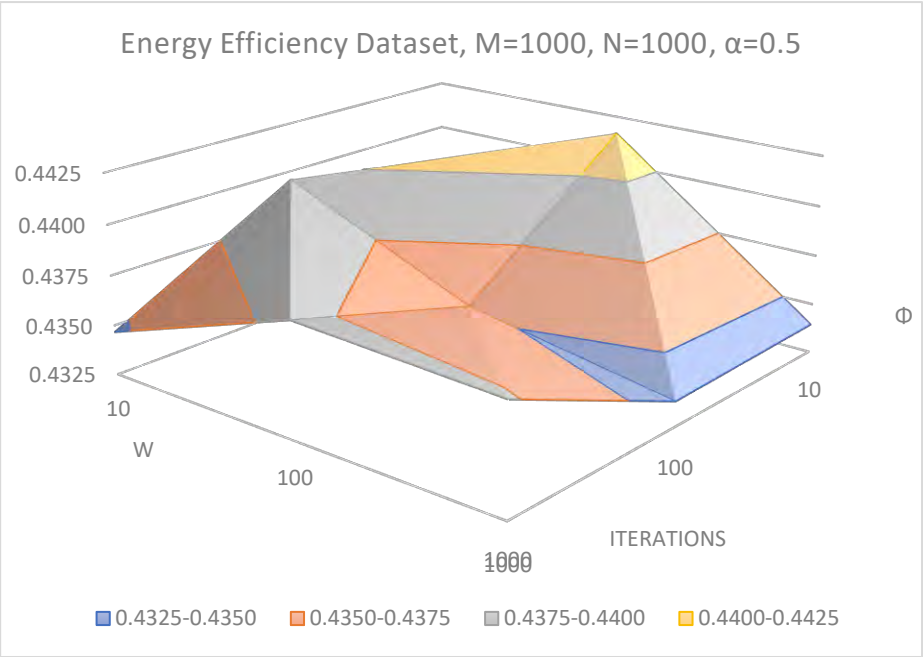


Figure 5.45 A 3D point of view on the Φ metric (W vs Iterations vs Φ) regarding the Energy Efficiency Dataset when N = 1000, M= 1000 & α = 0.5

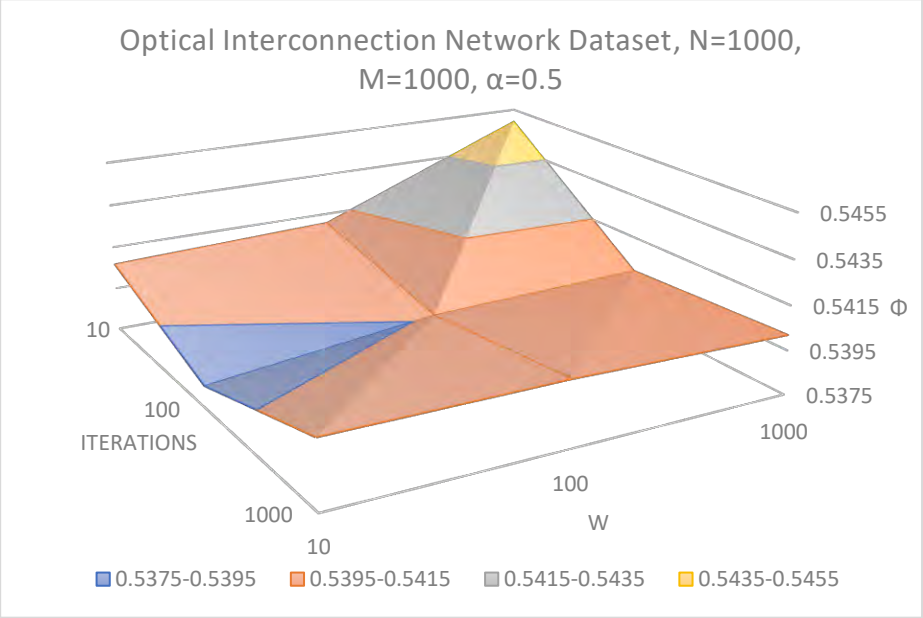


Figure 5.46 A 3D point of view on the Φ metric (W vs Iterations vs Φ) regarding the Optical Interconnection Network Dataset when N = 1000, M = 1000 & α = 0.5

5.2.4 Comparison of the Algorithms' Performance

The performance of the time T that each algorithm needs in order to allocate a query to a node is presented as follows based on Figures 5.47 through 5.53. As far as the HM is concerned, for small N values the time cost is low but as the number of queries-nodes N increases, T increases linearly. The CTA and SSO-TA algorithms' T values, when the number of queries equals the number of nodes, remain mainly constant with the CTA

exhibiting the best results among all algorithms when $N = 1000$. In case the number of nodes exceeds the number of queries, the CTA shows the best results while the SSO-TA's behavior is the worst, whereas for the opposite case the SSO-TA provides better time results than the CTA.

In Figures 5.54 through 5.60, we present our performance assessment results regarding the Φ metric for all three algorithms. The algorithms are compared based on the Φ metric for $\alpha = 0.5$, which means that Λ and Σ have the same weight. When the numbers of queries and nodes are equal, the three algorithms' Φ values follow a virtually steady course, since there are small increases which may be considered negligible. As the number of nodes increases in relation to the number of queries, the CTA's Φ values are reduced, whereas the SSO-TA's respective values increase. Lastly, in case $N > M$, we observe that the Φ metric's value decreases for the CTA while it increases for the SSO-TA.

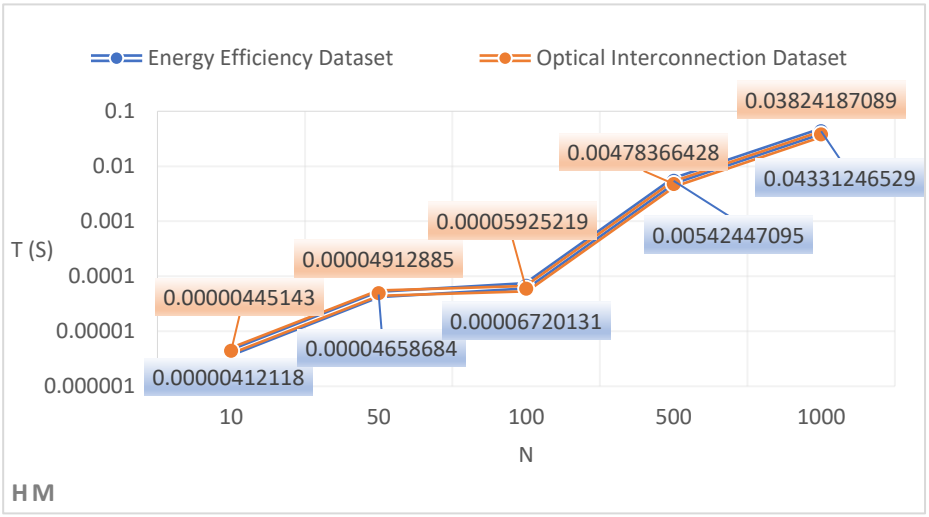


Figure 5.47 Experimental results for the T metric (T vs N) for HM

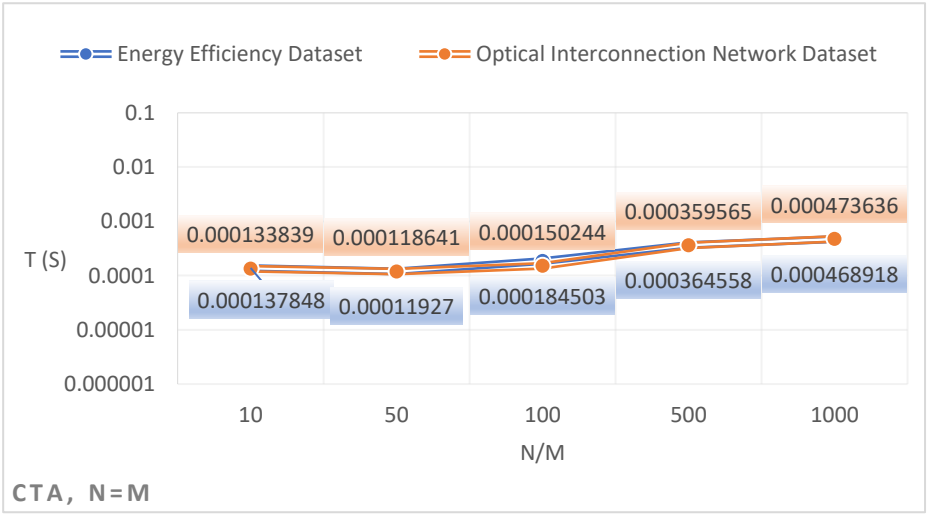


Figure 5.48 Experimental results for the T metric (T vs N/M) when N = M for CTA

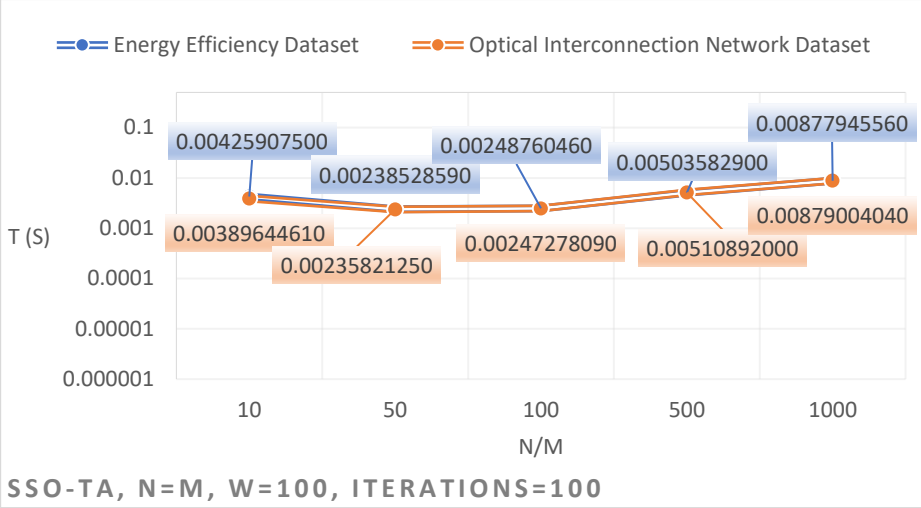


Figure 5.49 Experimental results for the T metric (T vs N/M) when N = M, W = 100 and Iterations = 100 for SSO-TA

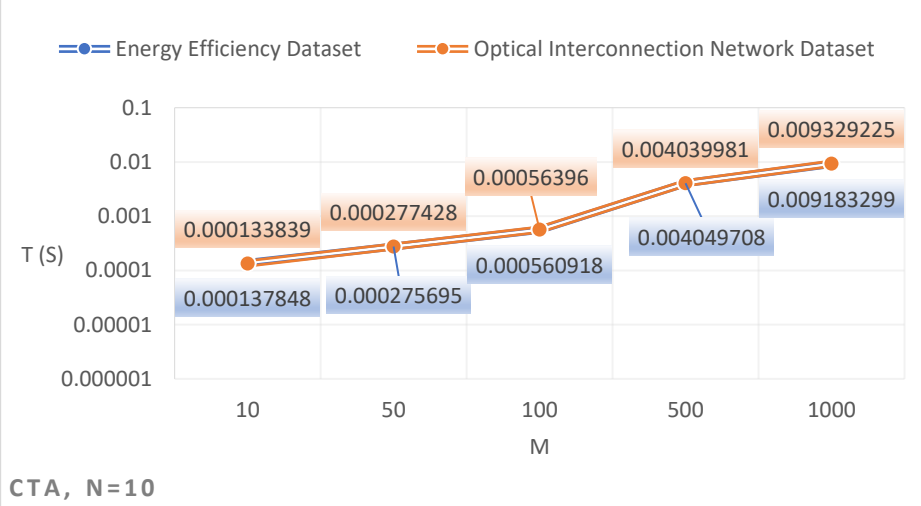


Figure 5.50 Experimental results for the T metric (T vs M) when N = 10 for CTA

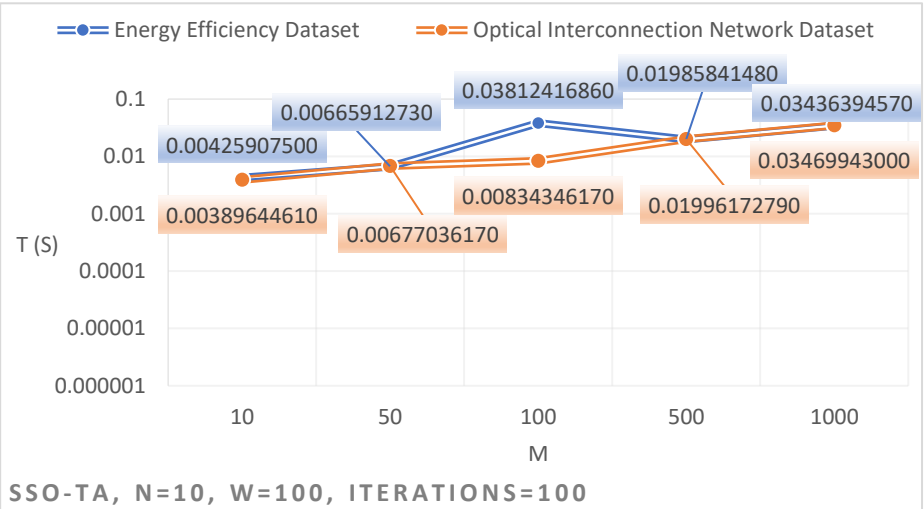


Figure 5.51 Experimental results for the T metric (T vs M) when N = 10, W = 100 and Iterations = 100 for SSO-TA

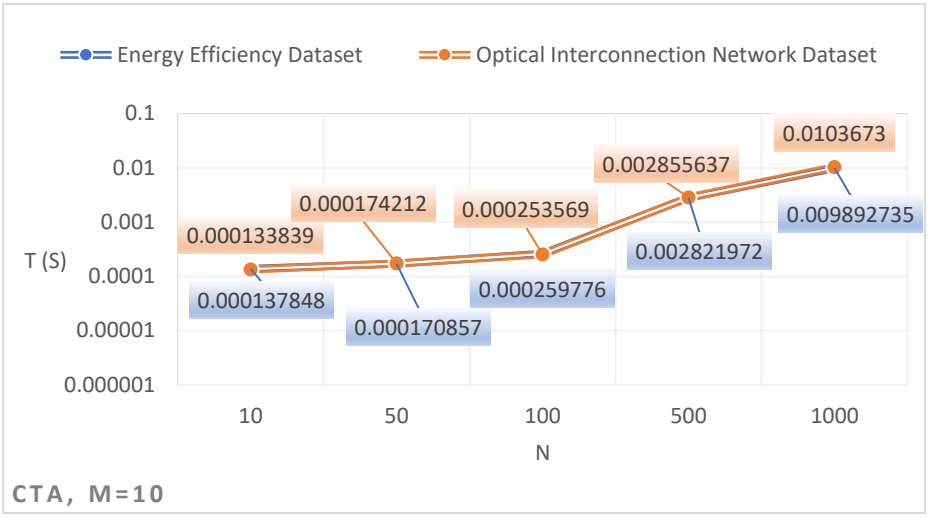


Figure 5.52 Experimental results for the T metric (T vs N) when M = 10 for CTA

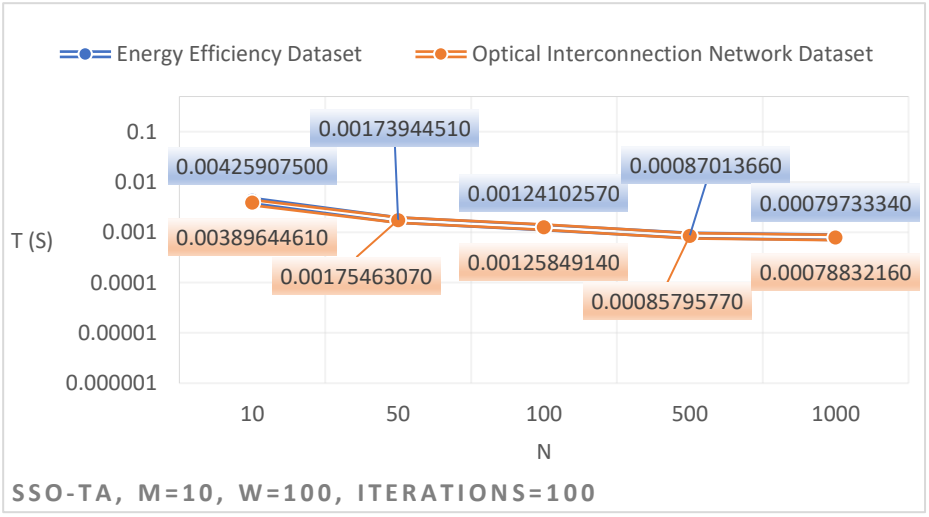


Figure 5.53 Experimental results for the T metric (T vs N) when M = 10, W = 100 and Iterations = 100 for SSO-TA

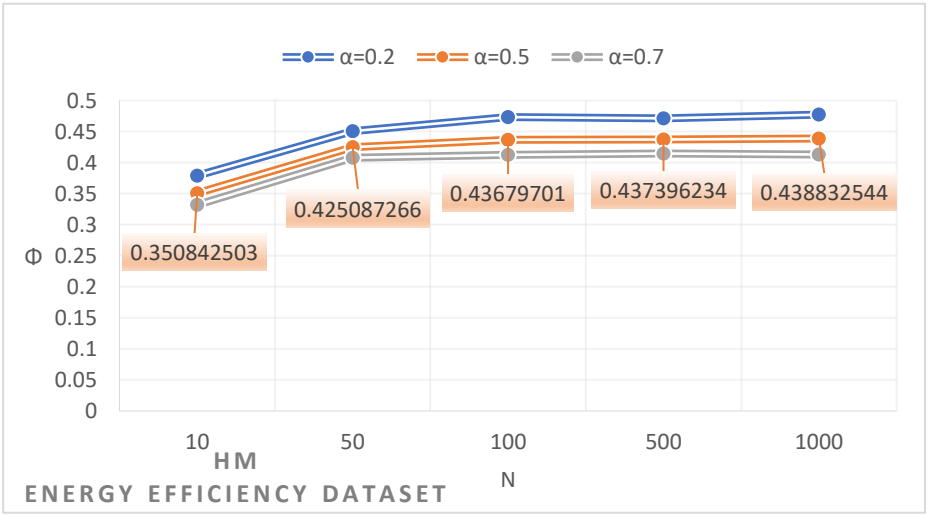


Figure 5.54 Experimental results for the Φ metric (Φ vs N) for HM regarding the Energy Efficiency Dataset

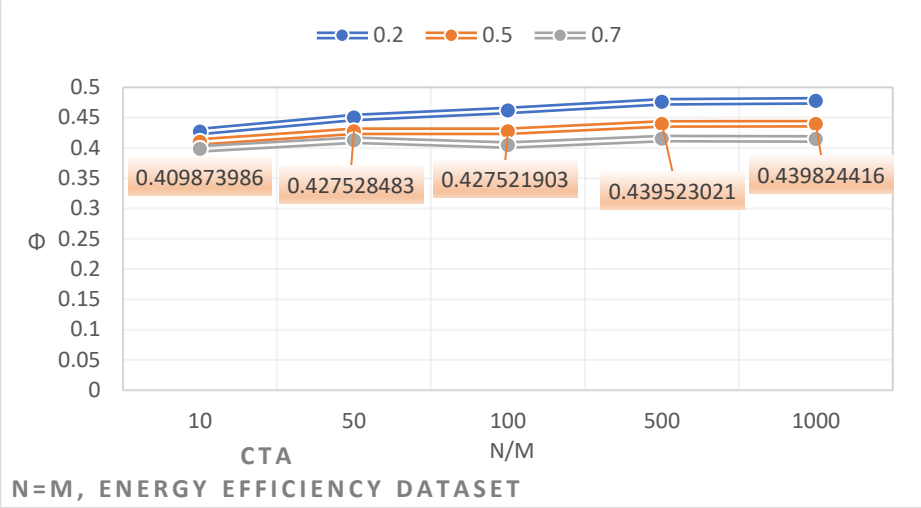


Figure 5.55 Experimental results for the Φ metric (Φ vs N/M) when N = M for CTA regarding the Energy Efficiency Dataset

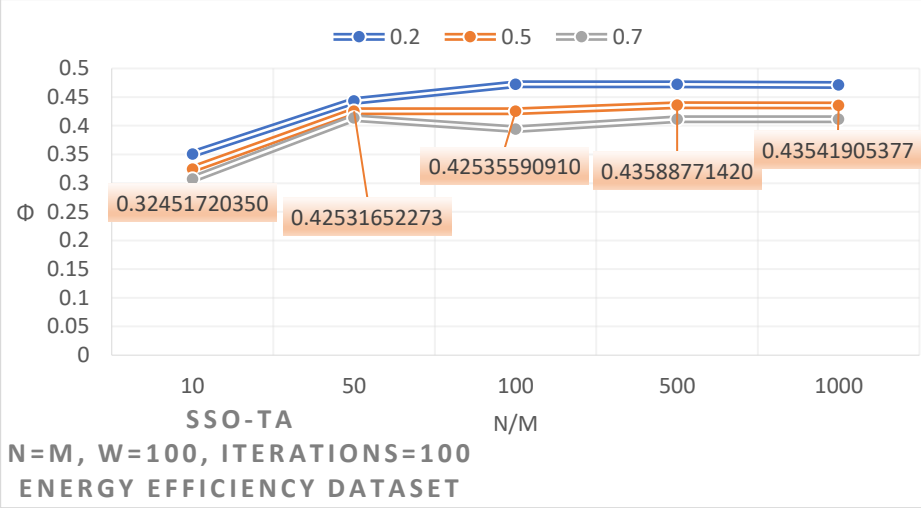


Figure 5.56 Experimental results for the Φ metric (Φ vs N/M) when N = M, W = 100 and Iterations = 100 for SSO-TA regarding the Energy Efficiency Dataset

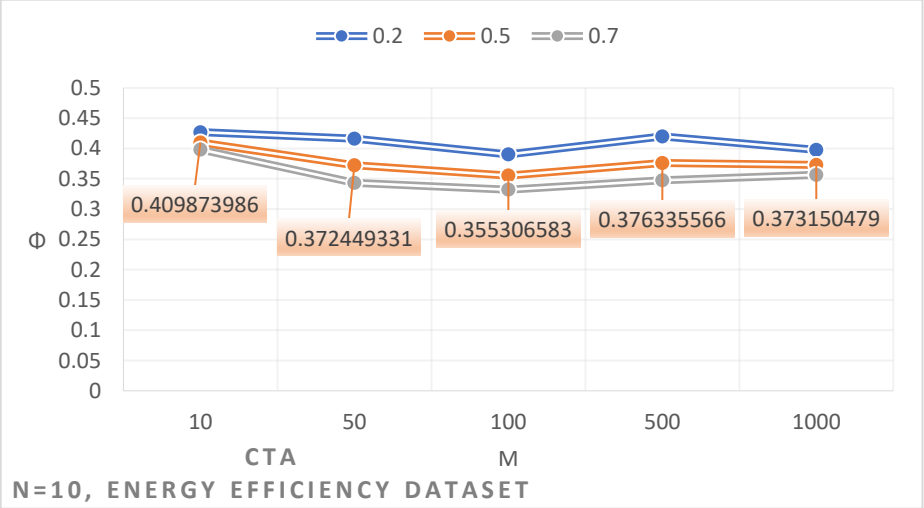


Figure 5.57 Experimental results for the Φ metric (Φ vs M) when N = 10 for CTA regarding the Energy Efficiency Dataset

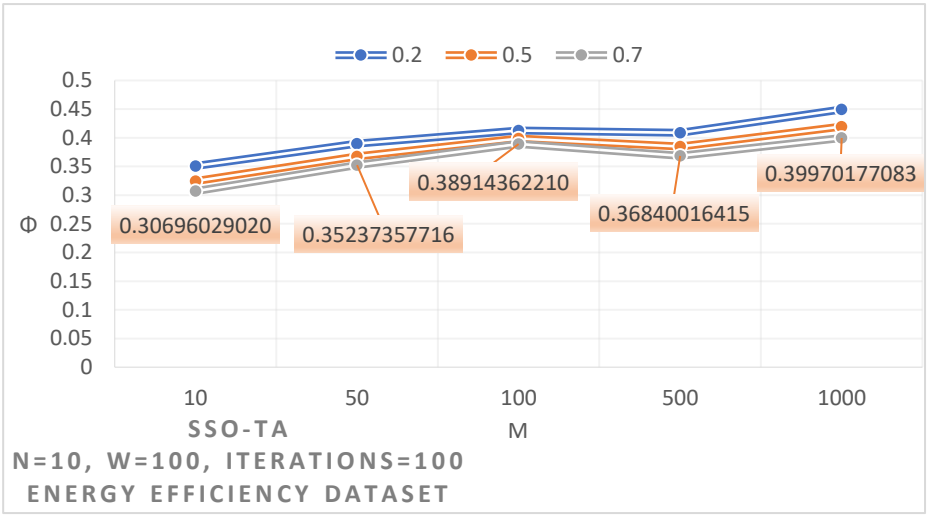


Figure 5.58 Experimental results for the Φ metric (Φ vs M) when N = 10, W = 100 and Iterations = 100 for SSO-TA regarding the Energy Efficiency Dataset

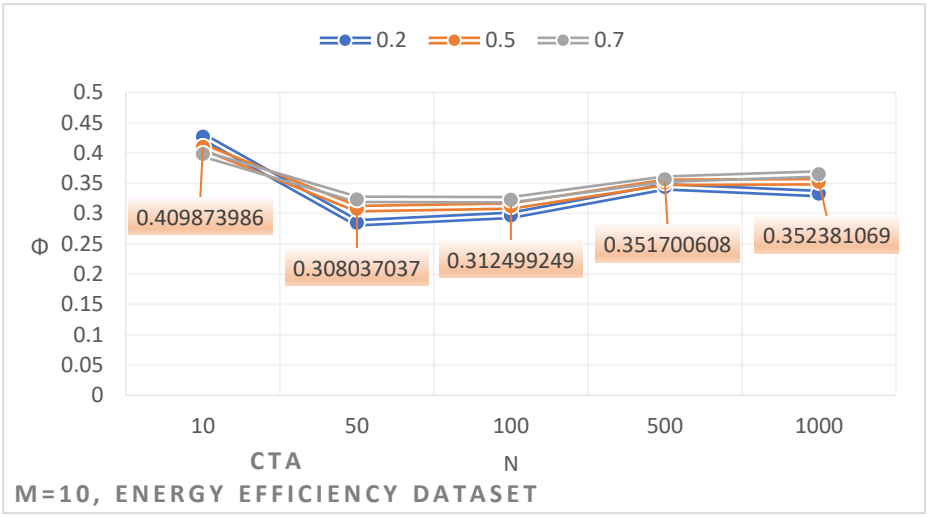


Figure 5.59 Experimental results for the Φ metric (Φ vs N) when M = 10 for CTA regarding the Energy Efficiency Dataset

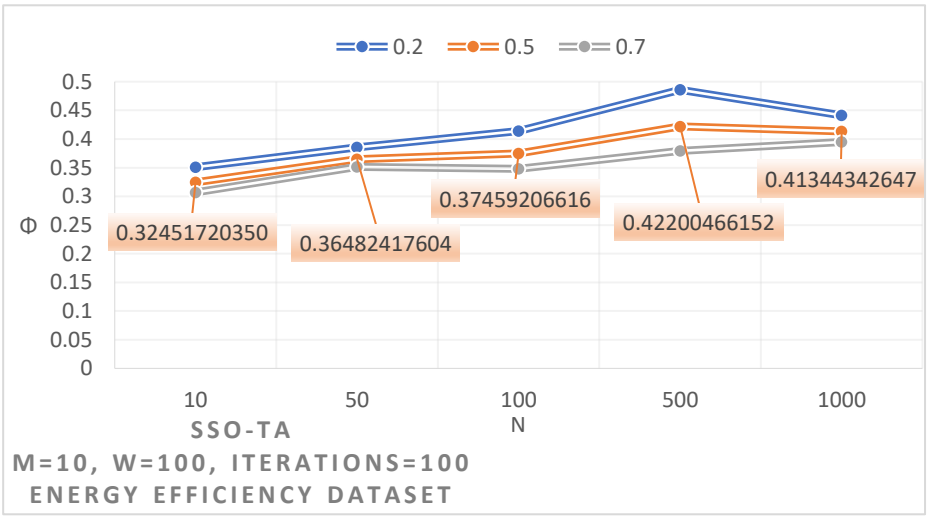


Figure 5.60 Experimental results for the Φ metric (Φ vs N) when M = 10, W = 100 and Iterations = 100 for SSO-TA regarding the Energy Efficiency Dataset

6 CONCLUSIONS AND FUTURE DIRECTIONS

The management of the huge amounts of data generated in a daily basis constitutes a challenge to be overcome in order to serve real time applications in the best way possible. It is a fact that the IoT applications which serve the Smart Cities need an intelligent scheme to process the continuously incoming queries. The proposed model adopts each of the three algorithms presented in Chapter 4 so as to decide which one gives the best solution to the problem. We define the notion of the Query Controller (QC), a module which orchestrates the query's allocation based on the nodes' characteristics — load and speed— aiming to reduce the time costs and achieve load balance among the nodes. Our experimental results show that different algorithms would best answer the problem's different cases, based on the results they exhibit. It seems that in a network where the number of queries exceeds the number of the available nodes, a scheme which is built on top of the SSO-TA algorithm provides solutions in the fastest way possible albeit the query-node fit is slightly worse than the CTA's. In case the number of the network's available nodes is greater than or equal to the number of the incoming queries, a mechanism which adopts the CTA algorithm exhibits lower time costs and achieves better query-node fits than the other algorithms. It appears that the HM algorithm performs in linear time complexity, while its query-node fit is quite similar to that of the CTA and SSO-TA's; hence, it is suggested only for few resources, i.e. up to 100 queries/nodes.

Our future research plans involve the incorporation of more parameters into the decision-making process. One such parameter can be a new query characteristic which will indicate the information the query demands. Additionally, a new node characteristic, which will represent the data that a QP can provide, may be integrated into our scheme. These characteristics will give us the opportunity to somehow measure the similarity between the information requested and the information to be returned. Moreover, the definition of a more complex cost function —incorporating the aforementioned characteristics— for the allocation of the queries is another research issue. In this way, we aim to return the data that best answer the queries in the shortest time possible as well as create a network which will use its resources in the most efficient way.

BIBLIOGRAPHY

[1] J. Munkres, "Algorithms for the assignment and transportation problems," in *Journal of the society for industrial and applied mathematics, 5(1)*, 1957, pp. 32-38.

[2] E. Brown, "Who Needs the Internet of Things?," Linux.com, [Online]. Available: https://www.linux.com/news/who-needs-internet-things.

[3] "Internet of Things Global Standards Initiative," [Online]. Available: http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx.

[4] O. Vermesan and P. Friess, Internet of things: converging technologies for smart environments and integrated ecosystems, River publishers, 2013.

[5] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *From active data management to event-based systems and more*, Berlin, Springer, 2010, pp. 242-259.

[6] G. Santucci, "The internet of things: Between the revolution of the internet and the metamorphosis of objects," in *Vision and Challenges for Realising the Internet of Things*, 2010, pp. 11-24.

[7] J. Teicher, "The little-known story of the first IoT device," IBM, [Online]. Available: https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/.

[8] M. Weiser, " The computer for the 21st century," in *IEEE pervasive computing*, 2002, pp. 19-25.

[9] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," in *CISCO white paper*, 2011, pp. 1-11.

[10] "Internet of Things Wiki," [Online]. Available: https://internetofthingswiki.com/wp-content/uploads/2019/01/internet-of-things.jpg.

[11] A. Ometov, S. V. Bezzateev, J. Kannisto, J. Harju, S. Andreev and Y. Koucheryavy, "Facilitating the Delegation of Use for Private Devices in the Era of the Internet of

Wearable Things," in *IEEE Internet of Things Journal, vol. 4, no. 4*, 2017, pp. 843-854.

[12] "How IoT's Are Changing The Fundamentals Of "Retailing"," [Online]. Available: https://trak.in/tags/business/2016/08/30/internet-of-things-iot-changing-fundamentals-of-retailing/.

[13] W. M. Kang, S. Y. Moon and J. H. Park, "An enhanced security framework for home appliances in smart home," in *Human-centric Computing and Information Sciences, 7(1), 6*, 2017.

[14] A. Meola, "How IoT & smart home automation will change the way we live," [Online]. Available: https://www.businessinsider.com/internet-of-things-smart-home-automation-2016-8.

[15] R. Picard and J. Healey, "Affective wearables," in *Personal Technologies, 1(4)*, Springer-Verlag, 1997, pp. 231-240.

[16] A. Gatouillat, Y. Badr, B. Massot and E. Sejdić, "Internet of Medical Things: A Review of Recent Contributions Dealing With Cyber-Physical Systems in Medicine," in *IEEE Internet of Things Journal, 5(5)*, 2018, pp. 3810-3822.

[17] C. A. da Costa, C. F. Pasluosta, B. Eskofier, D. B. da Silva and R. da Rosa Righi, "Internet of Health Things: Toward intelligent vital signs monitoring in hospital wards," in *Artificial Intelligence In Medicine*, 2018.

[18] K. Mahmud, G. E. Town, S. Morsalin and M. J. Hossain, "Integration of electric vehicles and management in the internet of energy," in *Renewable and Sustainable Energy Reviews, 82*, 2018, pp. 4179-4203.

[19] X. F. Xie and Z. J. Wang, "Integrated In-Vehicle Decision Support System for Driving at Signalized Intersections: A Prototype of Smart IoT in Transportation," 2017.

[20] J. Conway, "The Industrial Internet of Things: An Evolution to a Smart Manufacturing Enterprise," in *Schneider Electric Whitepaper*, 2015, p. 2.

[21] J. Lee, "Keynote Presentation: Recent Advances and Transformation Direction of PHM," in *Roadmapping Workshop on Measurement Science for Prognostics and Health Management of Smart Manufacturing Systems Agenda*, NIST, 2014.

[22] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati and G. P. Hancke, "Smart Grid Technologies: Communication Technologies and Standards," in *IEEE Transactions on Industrial Informatics*, 2011, pp. 529 - 539.

[23] Amazon AWS, "What is Cloud Computing?," [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/.

[24] Amazon AWS, "Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta," [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/.

[25] A. Regalado, "Who Coined 'Cloud Computing'?," MIT Technology Report, 31 October 2011. [Online]. Available: https://www.technologyreview.com/s/425970/who-coined-cloud-computing/.

[26] D. Hoffman, "What Is The Cloud," AT&T, [Online]. Available: https://www.youtube.com/watch?v=_a7hK6kWttE.

[27] "The Evolution of Consumer IoT," [Online]. Available: https://www.reply.com/en/topics/internet-of-things/the-evolution-of-the-consumer-internet-of-things.

[28] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," in *NIST Special Publication 145*, U.S. National Institute of Standards and Technology ITL Technical Report, 2011, pp. 6-7.

[29] Wikipedia, "Infrastructure as a service," [Online]. Available: https://en.wikipedia.org/wiki/Infrastructure_as_a_service.

[30] Wikipedia, "Platform as a service," [Online]. Available: https://en.wikipedia.org/wiki/Platform_as_a_service.

[31] Wikipedia, "Software as a service," [Online]. Available: https://en.wikipedia.org/wiki/Software_as_a_service.

[32] "Top 10 Cloud Computing Examples and Uses," 2017.

[33] "Backing it up: Cloud backup and recovery," 2017.

[34] "Edge Computing," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Edge_computing.

[35] MIT MTL, "Trends, Opportunities and Challenges Driving Architecture and Design of Next Generation Mobile Computing and IoT Devices," [Online]. Available: https://www.mtl.mit.edu/seminars/trends-opportunities-and-challenges-driving-architecture-and-design-next-generation-mobile#paragraphs-responsive-tabs-parent--field-seminar-tabs2.

[36] "Mobile Computing Opportunities, Challenges and Technology Drivers," [Online]. Available: http://www2.dac.com/events/videoarchive.aspx?confid=170&filter=keynote&id=170-103--0&#video.

[37] "Edge-Fog-Cloud Computing," [Online]. Available: https://it.kisspng.com/kisspng-6zoqt3/preview.html.

[38] B. Butler, "What is edge computing and how it's changing the network," Network World, 21 September 2017. [Online]. Available: https://www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html.

[39] W. Shi and S. Dustdar, "The promise of edge computing," in *Computer, 49(5)*, 2016, pp. 78-81.

[40] M. Satyanarayanan, "The emergence of edge computing," in *Computer, 50(1)*, 2017, pp. 30-39.

[41] O. Mäkinen, "Streaming at the edge: Local service concepts utilizing mobile edge computing," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, IEEE, 2015, pp. 1-6.

[42] K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, IEEE, 2017, pp. 68-73.

[43] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal and H. Flinck, "Mobile edge computing potential in making cities smarter," in *IEEE Communications Magazine, 55(3)*, IEEE, 2017.

[44] J. Liu, J. Wan, D. Y. Jia, B. Zeng, D. Li, C. H. Hsu and H. Chen, "High-efficiency urban-traffic management in context-aware computing and 5G communication," in *IEEE Communications Magazine, 55(1*, IEEE, 2017, pp. 34-40.

[45] E. D. Dickmanns, B. Mysliwetz and T. Christians, "An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles," in *IEEE Transactions on Systems, Man, and Cybernetics, 20(6)*, IEEE, 1990, pp. 1273-1284.

[46] Q. Z. N. Li and H. Cheng, "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection," in *IEEE Transactions on Intelligent Transportation Systems, 5(4)*, IEEE, 2004, pp. 300-308.

[47] W. Li, X. Jiang and Y. Wang, "Road recognition for vision navigation of an autonomous vehicle by fuzzy reasoning," in *Fuzzy Sets and Systems, 93(3)*, 1998, pp. 275-280.

[48] F. Thomanek, E. D. Dickmanns and D. Dickmanns, "Multiple object recognition and scene interpretation for autonomous road vehicle guidance," in *Proceedings of the Intelligent Vehicles' 94 Symposium*, IEEE, 1994, pp. 231-236.

[49] D. M. Edison, F. S. Marton and J. N. Ucovich, "Remote monitoring and controlling system for subsea oil/gas production equipment". Washington, DC: U.S. Patent and Trademark Office Patent U.S. Patent No. 4,138,669, 6 February 1979.

[50] S. O. Johnsen, M. A. Lundteigen, H. Fartum, J. Monsen and N. Hydro, "Identification and reduction of risks in remote operations of offshore oil and gas installations," in *Advances in Safety and Reliability, 1*, 2005, pp. 957-964.

[51] S. Singh and N. Singh, "Big Data Analytics," in *Proc. of the Int. Conf. on Communication*, Information and Computing Technology, 2012.

[52] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis and K. Taha, "Efficient Machine Learning for Big Data: A Review," in *Big Data Research, vol. 2(3)*, 2015, pp. 87-93.

[53] A. Cooper, "What is analytics? Definition and essential characteristics," in *CETIS Analytics Series, 1(5)*, 2012, pp. 1-10.

[54] A. Van Barneveld, K. E. Arnold and J. P. Campbell, "Analytics in higher education: Establishing a common language," in *EDUCAUSE learning initiative, 1(1), l-ll*, 2012.

[55] T. H. Davenport, "Competing on Analytics," in *Harvard business review*, 2006, pp. 98-107.

[56] T. K. Das and P. M. Kumar, "Big data analytics: A framework for unstructured data analysis," in *International Journal of Engineering Science & Technology, 5(1), 153*, 2013.

[57] Wikipedia, "Data Acquisition," [Online]. Available: https://en.wikipedia.org/wiki/Data_acquisition.

[58] Techopedia, "Data Organization," [Online]. Available: https://www.techopedia.com/definition/30624/data-organization.

[59] H. H. D. Erhard Rahm, "Data Cleaning: Problems and Current Approaches," in *Bulletin of the Technical Committee on Data Engineering*, 2000, p. 5.

[60] J.-C. F. Heiko Müller, Problems, methods, and challenges in comprehensive data cleansing, 2003.

[61] A. N. Eiman, A. N. Hind, M. Nader and A. J. Jameela, "Applications of big data to smart cities," in *Journal of Internet Services and Applications, vol. 6(25)*, 2015.

[62] Y. Sun, H. Song, A. Jara and R. Bie, "Internet of Things and Big data Analytics for Smart Cities and Connected Communities," in *IEEE Access, vol. 4*, 2016, pp. 766-773.

[63] D. H. Spatti and L. H. Bartocci Lobini, "Computational Tools for Data Processing in Smart Cities," in *Smart Cities Technologies*, InTech, 2016.

[64] D. Bonino, F. Rizzo, C. Pastrone, J. A. C. Soto, M. Ahlsen and M. Axling, "Block-based realtime big-data processing for smart cities," in *Proceedings of the IEEE International Smart Cities Conference (ISC2)*, Trento, Italy, 2016.

[65] B. N. Silva, M. Khan and K. Han, "Big Data Analytics Embedded Smart City Architecture for Performance Enhancement through Real-Time Data Processing and Decision-Making," in *Wireless Communications and Mobile Computing, vol. 2017*, 2017.

[66] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari and I. Stoica, "Knowing When You're Wrong: Building Fast and Reliable Approximate Query Processing Systems," ACM SIGMOD, USA, 2014.

[67] C. Bishop, "Pattern Recognition and Machine Learning," Springer, 2009.

[68] J. M. Hellerstein and R. Avnur, "Informix under control: Online query Processing," in *Data Mining and Knowledge Discovery Journal*, 2000.

[69] N. Pansare, V. R. Borkar, C. Jermaine and T. Condie, "Online aggregation for large MapReduce jobs," in *PVLDB*, 2011.

[70] H. Kahn, "Use of different Monte Carlo sampling techniques," 1955.

[71] K. Kolomvatsos, C. Anagnostopoulos and S. Hadjiefthymiades, "An efficient time optimized scheme for progressive analytics in big data," in *Big Data Research, 2(4)*, 2015, pp. 155-165.

[72] B. Chandramouli, J. Goldstein and A. Quamar, "Scalable Progressive Analytics on Big Data in the Cloud," in *Proceedings of the VLDB Endowment, vol. 6(14)*, 2013.

[73] S. Chaudhuri, G. Das and U. Srivastava, "Effective use of block-level sampling in statistics estimation," in *SIGMOD*, 2004.

[74] A. Doucet, M. Briers and S. Senecal, "Efficient block sampling strategies for sequential Monte Carlo methods," in *Journal of Computational and Graphical Statistics*, 2006.

[75] V. Raman, B. Raman and J. M. Hellerstein, "Online dynamic reordering for interactive data processing," in *VLDB*, 1999.

[76] S. Erevelles, N. Fukawa and L. Swayne, "Big Data consumer analytics and the transformation of marketing," in *Journal of Business Research 69*, Elsevier, 2016, pp. 897-904.

[77] Wikipedia, "Customer Analytics," [Online]. Available: https://en.wikipedia.org/wiki/Customer_analytics.

[78] A. Dev Mishra and Y. Beer Singh, "Big Data Analytics for Security and Privacy Challenges," in *International Conference on Computing, Communication and Automation (ICCCA2016)*, IEEE, 2016, pp. 50-53.

[79] A. A. Cárdenas, P. K. Manadhata and S. P. Rajan, "Big data analytics for security," in *IEEE Security & Privacy, 11(6)*, 2013, pp. 74-76.

[80] M. Banday and J. Qadri, "SPAM – Technological and Legal Aspects," in *Kashmir University Law Review, Vol. 8, No. 8*, 2006.

[81] L. Lu, R. Perdisci and W. Lee, "Surf: Detecting and Measuring Search Poisoning," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 467-476.

[82] Q. Gu and P. Liu, "Denial of Service Attacks," in *Handbook of Computer Networks: Distributed Networks, Network Planning, Control, Management, and New Trends and Applications, 3*, 2007, pp. 454-468.

[83] M. Jakobsson and S. Myers, Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft, John Wiley & Sons, Inc., 2006.

[84] J. Shi and S. Saleem, "Phishing," [Online]. Available: https://www. cs. arizona. edu/~ collberg/Teaching/466-566/2014/Resources/presentations/2012/topic5-final/report. pdf.

[85] "Symantec," [Online]. Available: http://www.symantec.com/index.jsp.

[86] T. Mahmood and U. Afzal, "Security analytics: Big data analytics for cybersecurity: A review of trends, techniques and tools," in *2013 2nd national conference on Information assurance (ncia)*, IEEE, 2013, pp. 129-134.

[87] D. Zeng, H. Chen, R. Lusch and S. H. Li, "Social Media Analytics and Intelligence," in *Commun. Acm, 57(6)*, 2014, pp. 74-81.

[88] W. Fan and M. D. Gordon, "The Power of Social Media Analytics," in *Commun. Acm, 57(6)*, 2014, pp. 74-81.

[89] W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential," in *Health information science and systems, 2(1)*, 2014, p. 3.

[90] C. S. M. S. C. Margaret Rouse, "database management system (DBMS)," [Online]. Available: https://searchsqlserver.techtarget.com/definition/database-management-system.

[91] Studytonight, "DBMS Database Models," [Online]. Available: https://www.studytonight.com/dbms/database-model.php.

[92] Wikipedia, "Relational Database Model," [Online]. Available: https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjjsYzptoTiAhVJJ1AKHfH4AYEQjRx6BAgBEAU&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FRelational_model&psig=AOvVaw1mvDL7_o5a5i17voznuv84&ust=1557147015858377.

[93] Wikipedia, "Hierarchical Database Model," [Online]. Available: https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjPq6LdtoTiAhXOZ1AKHaZrA3YQjRx6BAgBEAU&url=https%3A%2F%2Fwww.studytonight.com%2Fdbms%2Fdatabase-model.php&psig=AOvVaw1NLq8fSpBzBcsqG0eYDhoe&ust=1557146991335329.

[94] B. J. Cragun, D. R. Fish, C. T. Rath and D. A. Wall, *Federated query management,* 2007.

[95] S. Katariya, *Layered Query Management,* 2001.

[96] Oracle, "Oracle Partitioning," [Online]. Available: https://www.oracle.com/technetwork/database/options/partitioning/overview/index.html.

[97] "Particionamiento," [Online]. Available: http://vendara-oratun.blogspot.com/2011/11/particionamiento.html.

[98] "Table Partitioning In SQL Server," [Online]. Available: https://www.c-sharpcorner.com/article/table-partitioning-in-sql-server/.

[99] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul and S. Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," in *VLDB Journal, 12(2)*, 2003.

[100] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava and J. Widom, "STREAM: The Stanford Data Stream Management System," Springer, 2004.

[101] S. Chandrasekaran and M. J. Franklin, "PSoup: a system for streaming queries over streaming data," in *VLDB Journal 12(2)*, 2003, pp. 140-156.

[102] C. Cranor, T. Johnson, O. Spataschek and V. Shkapenyuk, "Gigascope: A Stream Database for Network Applications," in *Proc. of SIGMOD*, 2003.

[103] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. S. Manku, C. Olston, J. Rosenstein and R. Varma, "Query Processing, Approximation, and Resource Management in a Data Stream Management System," in *CIDR*, 2003.

[104] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," in *SIGMOD Record, 31(3)*, 2002.

[105] K. Kolomvatsos and C. Anagnostopoulos, "Reinforcement learning for predictive analytics in smart cities," in *Informatics (Vol. 4, No. 3, p. 16)*, Multidisciplinary Digital Publishing Institute, 2017.

[106] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," in *Naval research logistics quarterly, 2(1-2)*, 1955, pp. 83-97.

[107] J. Edmonds, "Maximum matching and a polyhedron with 0 − 1 vertices," *Journal of Research of the National Bureau of Standards, (B) 69,* pp. 125-130, 1965.

[108] J. Edmonds and R. Karp, "Theoretical improvements in algorithmic efficiency," *ACM, 19,* pp. 248-264, 1972.

[109] M. &. T. N. Iri, "An algorithm for finding an optimal" independent assignment"," *Journal of the Operations Research Society of Japan, 19(1),* pp. 32-57, 1976.

[110] E. Dinits, "Algorithm for solution on a problem on maximum flow in a network with power estimation," *Soviet Mathematics Doclady, 11,* pp. 1277-1280, 1970.

[111] L. Ford and D. Fulkerson, Flows in Networks, Princeton NJ.: Princeton Univ. Press, 1962.

[112] [Online]. Available: http://www.lix.polytechnique.fr/~ollivier/JACOBI/jacobiEngl.htm.

[113] C. Annamalai, "Finding perfect matchings in bipartite hypergraphs," in *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2016, pp. 1814-1823.

[114] H. E. Driver and A. L. Kroeber, "Quantitative Expression of Cultural Relationships," in *American Archaeology and Ethnology*, University of California Publications , 1932, pp. 211-256.

[115] J. Zubin, "A technique for measuring like-mindedness," *The Journal of Abnormal and Social Psychology 33 (4),* p. 508–516, 1938.

[116] R. C. Tryon, Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality, Ann Arbor: Edwards Brothers, 1939.

[117] R. B. Cattell, "The description of personality: Basic traits resolved into clusters," *Journal of Abnormal and Social Psychology,* vol. 38, no. 4, p. 476–506, 1943.

[118] H. Meireles Valadares, Clustering task assignment: an algorithm for time critical task assignment problems, UNIVERSIDAD POLITÉCNICA DE MADRID, 2017.

[119] A. K. Jain, M. N. Murty and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR),* vol. 31, no. 3, pp. 264-323, 1999.

[120] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995.

[121] J. Kennedy and R. Eberhart, Swarm Intelligence., Morgan Kaufmann, 2001.

[122] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1998.

[123] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications,* 2008.

[124] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary Computation,* 2017.

[125] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1997, p. 303–308.

[126] D. Bratton and T. Blackwell, "A Simplified Recombinant PSO," *Journal of Artificial Evolution and Applications,* pp. 1-10, 2008.

[127] M. Pedersen, Tuning & Simplifying Heuristical Optimization, University of Southampton, School of Engineering Sciences, Computational Engineering and Design Group, 2010.

[128] M. Pedersen and A. Chipperfield, "Simplifying particle swarm optimization," *Applied Soft Computing,* vol. 10, no. 2, pp. 618-628, 2010.

[129] X. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, 2008.

[130] J. Kennedy, "Bare Bones Particle Swarms," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, 2003, pp. 80-87.

[131] X. S. Yang, S. Deb and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," in *NDT 2011, Springer CCIS 136*, Springer, 2011, pp. 53-66.

[132] W.-C. Yeh, C.-M. Lai, Y.-C. Huang, T.-W. Cheng, H.-P. Huang and Y. Jiang, "Simplified Swarm Optimization for Task Assignment Problem in distributed computing system," in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, IEEE, 2017, pp. 773-776.

[133] P. R. Dickson and J. L. Ginter, "Market segmentation, product differentiation, and marketing strategy," in *Journal of marketing, 51(2)*, 1987, pp. 1-10.

[134] Wikipedia, "Virtual machine," [Online]. Available: https://en.wikipedia.org/wiki/Virtual_machine.

[135] Wikipedia, "Application programming interface," [Online]. Available: https://en.wikipedia.org/wiki/Application_programming_interface.

[136] Wikipedia, "Indirection," [Online]. Available: https://en.wikipedia.org/wiki/Indirection.

[137] Wikipedia, "Unstructured data," [Online]. Available: https://en.wikipedia.org/wiki/Unstructured_data.

[138] Wikipedia, "Data model," [Online]. Available: https://en.wikipedia.org/wiki/Data_model.

[139] Wikipedia, "Tuple," [Online]. Available: https://en.wikipedia.org/wiki/Tuple.

[140] Wikipedia, "Combinatorial Optimization," [Online]. Available: https://en.wikipedia.org/wiki/Combinatorial_optimization.