UNIVERSITÉ DE MONTRÉAL

OPTIMIZATION AND MINING METHODS FOR EFFECTIVE REAL-TIME EMBEDDED SYSTEMS

RABEH AYARI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
MARS 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

OPTIMIZATION AND MINING METHODS FOR EFFECTIVE REAL-TIME EMBEDDED SYSTEMS

présentée par : AYARI Rabeh
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

Mme BOUCHENEB Hanifa, Doctorat, présidente
Mme NICOLESCU Gabriela, Doctorat, membre et directrice de recherche
M. BELTRAME Giovanni, Ph. D., membre et codirecteur de recherche
M. GUÉHÉNEUC Yann-Gaël, Doctorat, membre
M. PASRICHA Sudeep, Ph. D., membre externe

DEDICATION

I would like to dedicate this thesis to : My
parents : Hannoud and Salih
My parents in law : Majda and Maryouma

And last but not least,
My beloved wife Ameni, for all of her constant love, support and inspiration. I can't even in imagination pay the immense debt of gratitude I owe her. We did it !

ACKNOWLEDGEMENTS

I would like to thank all those who have supported me during my doctoral studies. I am particularly grateful to my supervisor, Professor Gabriela Nicolescu, not only for her sincere support during my studies but also for her continuous encouragement and understanding. It was a great honor to work with such a wonderful supervisor. Without her precious advice, guidance and persistent help, this thesis would not have been possible.

I would like to extend my gratitude to my co-supervisor, Professor Giovanni Beltrame, for his continuous guidance and insights.

I also acknowledge all my committee members : Professor Hanifa Bouchenab, Professor Sudeep Pasricha, and Professor Yann-Gael Guéhéneuc for sparing their precious time in order to review and evaluate my research work.

This thesis was partially supported by CAE Inc. which provided insight and expertise that greatly assisted the research. We specially thank Patricia Gilbert, Michel Galibois, Aurelain Constantinescu and Jean-Pierre Rousseau.

A special thought goes to my colleagues at HESLab with whom I shared the office, for great support and friendship during these years.

ABSTRACT

The Internet of things (IoT) is the network of interrelated devices or objects, such as self-driving cars, home appliances, smart-phones and other embedded computing systems. It combines hardware, software, and network connectivity enabling data processing using powerful cloud data centers. However, the exponential rise of IoT applications reshaped our belief on the cloud computing, and long-lasting certainties about its capabilities had to be updated. The classical centralized cloud computing is encountering several challenges, such as traffic latency, response time, and data privacy. Thus, the trend in the processing of the generated data of IoT inter-connected embedded devices has shifted towards doing more computation closer to the device in the edge of the network. This possibility to do on-device processing helps to reduce latency for critical real-time applications and better processing of the massive amounts of data being generated by these devices.

Succeeding this transition towards the edge computing requires the design of high-performance embedded systems by efficiently exploring design alternatives (i.e. efficient Design Space Exploration), optimizing the deployment topology of multi-processor based real-time embedded systems (i.e. the way the software utilizes the hardware), and light mining techniques enabling smarter functioning of these devices.

Recent research efforts on embedded systems have led to various automated approaches facilitating the design and the improvement of their functioning. However, existing methods and techniques present several major challenges. These challenges are more relevant when it comes to real-time embedded systems. Four of the main challenges are : (1) The lack of online data mining techniques that can enhance embedded computing systems functioning on the fly ; (2) The inefficient usage of computing resources of multi-processor systems when deploying software on ; (3) The pseudo-random exploration of the design space ; (4) The selection of the suitable implementation after performing the otimization process ;

In this context, the overall objective of our research is to propose a set of new models, methods and algorithms helping designers to efficiently design and use real-time embedded systems. The main contributions of this thesis include but not limited to :

1. The proposition of a new online clustering algorithm to classify streaming data on the fly. The proposed algorithm enables smarter functioning of these real-time embedded devices.
2. The definition and implementation of a fully automated multi-objective exploration

process to map critical real-time application to heterogeneous multi-processor systems.

3. The introduction of an improved evolutionary algorithm incorporating problem structure during the exploration process.
4. The definition of a new approach to assist designers during the selection of the proper system implementation.

RÉSUMÉ

L'Internet des objets (IoT) est le réseau d'objets interdépendants, comme les voitures autonomes, les appareils électroménagers, les téléphones intelligents et d'autres systèmes embarqués. Ces systèmes embarqués combinent le matériel, le logiciel et la connection réseau permettant le traitement de données à l'aide des puissants centres de données de l'informatique nuagique. Cependant, la croissance exponentielle des applications de l'IoT a remodelé notre croyance sur l'informatique nuagique, et des certitudes durables sur ses capacités ont dû être mises à jour. De nos jours, l'informatique nuagique centralisé et classique rencontre plusieurs défis, tels que la latence du trafic, le temps de réponse et la confidentialité des données. Alors, la tendance dans le traitement des données générées par les dispositifs embarqués interconnectés consiste à faire plus de calcul au niveau du dispositif au bord du réseau. Cette possibilité de faire du traitement local aide à réduire la latence pour les applications temps réel présentant des fortes contraintes temporelles. Aussi, ça permet d'améliorer le traitement des quantités massives de données générées par ces périphériques. Réussir cette transition nécessite la conception de systèmes embarqués de haute performance en explorant efficacement les alternatives de conception (Exploration efficace de l'espace des solutions), en optimisant la topologie de déploiement des applications temps réel sur des architectures multi-processeurs (la façon dont le logiciel utilise le matériel) , et des algorithme d'exploration permettant un fonctionnement plus intelligent de ces dispositifs.

Des efforts de recherche récents ont conduit à diverses approches automatisées facilitant la conception et l'amélioration du fonctionnement des système embarqués. Cependant, ces techniques existantes présentent plusieurs défis majeurs. Ces défis sont fortement présents sur les systèmes embarqués temps réel. Quatre des principaux défis sont : (1) Le manque de techniques d'exploration de données en ligne permettant l'amélioration des performances des systèmes embarqués. (2) L'utilisation inefficace des ressources informatiques des systèmes multiprocesseurs lors du déploiement de logiciels là dessus ; (3) L'exploration pseudo-aléatoire de l'espace des solutions (4) La sélection de la configuration appropriée à partir de la listes des solutions optimales obtenue.

Dans ce contexte, l'objectif global de cette thèse est de proposer de nouveaux modèles, méthodes et algorithmes aidant les concepteurs à concevoir efficacement des applications temps réel complexes et déployées sur des architectures multi-processeurs modernes. Les principales contributions de cette thèse comprennent, mais sans s'y limiter à :

1. La proposition d'un algorithme de clustering en ligne qui permet de classer un flux de

données à la volée. L'algorithme proposé permet un fonctionnement plus intelligent de ces dispositifs.

2. La définition d'un processus d'exploration multi-objectif automatisé pour mapper des applications temps réel à un système multiprocesseurs hétérogène.
3. L'introduction d'un algorithme évolutif amélioré incorporant la structure du problème d'ordonnancement au cours du processus d'exploration.
4. La définition d'une nouvelle approche pour aider les concepteurs lors de la sélection de la bonne solution à la fin du processus d'exploration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SIGNS AND ABBREVIATIONS

CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DBT	Dynamic Binary Translation
DSE	Design Space Exploration
EDF	Earliest Deadline First
FPS	Fixed priority scheduling
FMS	Full-Mission Simulator
GA	Genetic Algorithm
ISS	Instruction Set Simulator
ML	Machine Learning
MCS	Mixed Criticality System
MOEA	Multi-Objective Evolutionary Algorithms
MOOA	Multi-objective optimization algorithms
MPSoC	Multi-processor System-on-chip
NoC	Network on Chip
NSGA	Non-dominated Sorting Genetic Algorithm
NSGAii	Non-dominated Sorting Genetic Algorithm II
NSPSO	Non-dominated Sorting Particle Swarm Optimization
PCA	Principal Component Analysis
PE	Processing Element
RM	Rate Monotonic
RR	Round Robin
RTOS	Real-time operating system
SA	Simulated Annealing
SPEAii	Strength Pareto Evolutionary Algorithm
IoT	Internet of Things
NTSA	National Training Systems Association
DeDaSC	Delaunay triangulation-based Data Stream Clustering algorithm
HypAp	Hypervolume-Based Approach for Refining the Design of Embedded Systems

ImGA	An Improved Genetic Algorithm for Partitioned Scheduling on Heterogeneous Multi-core Systems
DBSCAN	Density Based Spatial Clustering of Applications with Noise
OPTICS	Ordering Points To Identify the Clustering Structure
DENCLUE	DENsity-based CLUstEring
CLARANS	A Method for Clustering Objects for Spatial Data Mining
CURE	Clustering Using Representatives
CHAMELEON	A Hierarchical Clustering Algorithm Using Dynamic Modeling
BIRCH	balanced iterative reducing and clustering using hierarchies
STING	STatistical INformation Grid-based method
WaveCluster	A Multi-Resolution Clustering Approach for Very Large Databases
WCET	Worst Case Execution Time
SGX	Schedulability Guided Crossover Operator
MMM	Min-Max Mutation Operator
NP-hard	Non-deterministic Polynomial-time hard
PF	Pareto Front
TPF	True Pareto Front
RPF	Reduced Pareto Front
QoS	Quality of Service
HDMS	Heterogeneous Distributed Multi-processor System
CCM	Communication Cost MATRIX
MCV	Memory Consumption Vector
TGFF	Task Graphs For Free
DT	Delaunay Triangulation
DS	Data Set

CHAPTER 1 INTRODUCTION

1.1 Research Context : Internet of Things and Edge Computing

In recent years, the Internet of Things (IoT), attracted the attention of researchers from academia and industry throughout the world. Introducing IoT-based products and services has been a growing trend thanks to rapid advances of numerous technologies, including ubiquitous sensors, self-driving cars, bio-medical systems, and the list goes on. According to Gartner, Inc. (Fenn and Raskino, 2011), more than 20 *billion* connected objects will be in use worldwide by 2020. This huge expansion in terms of connected devices will raise the scale of produced, as well as consumed, data to an unprecedented level. Cisco reported that IoT networks will generate more than 400 *zettabytes* (*trillion gigabytes*) of data a year in 2018 (Index, 2015). The huge size of generated data elucidate the key role of cloud computing in IoT success.

Cloud computing allows devices to perform complex computing tasks using data managed remotely on data centers. Cloud computing has virtually unlimited capabilities in terms of data processing, storage and control. By taking advantage of cloud computing capacities, IoT manufacturers did not foresee any trouble. However, due to the exponential growth of IoT networks, in the next few years, cloud will face increasing difficulties to meet the streaming data processing requirements and did not live up to the expectations of the IoT community (Aazam et al., 2014). Thus, these limitations reverse the trend towards doing more computation and analysis on the devices themselves instead of sending all the generated data to the cloud for processing. This on-device computation enables lower response times, as well as reduced traffic and directly benefiting IoT scaling. The ability to do advanced on-device processing (i.e. near the source of the data) is referred to as edge computing (Bonomi et al., 2012). Edge computing pushes applications, data and computing power away from centralized data centers to the edge of the network, closer to the source of data (Shi et al., 2016) as depicted in Figure 1.1.

Edge computing will have a huge impact on IoT Quality of Service (QoS), fueling strong ecosystem growth as end devices become more powerful and capable of running sophisticated applications such as object detection, face recognition, language processing, and obstacle avoidance (Shi et al., 2016). For that it is crucial that each device must be able to handle the data and activate appropriate responses quickly and safely, while consuming little power on platforms with minimal footprints. Therefore, the key success factor of IoT network enabling edge computing is the design of powerful devices and light mining of the generated data.

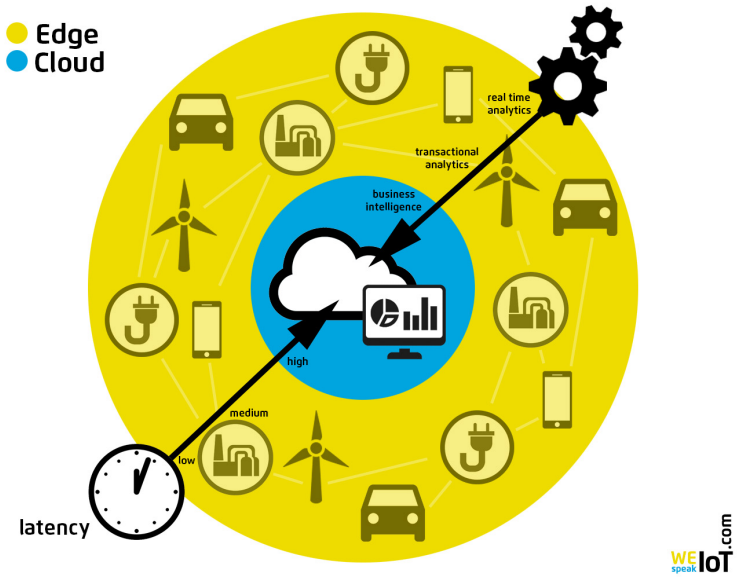


Figure 1.1 Edge computing allows data from internet of things devices to be analyzed on-device at the edge of the network and avoids the cloud as a possible bottleneck (local processing reducing the backhaul traffic to the central repository). (Manhart, 2017).

Advances in the design of embedded systems have given rise to many commercial devices that are powerful enough to run on-device complex real-time applications. These advances in integrated circuits and embedded systems pursue new opportunities but also several challenges. To succeed in this dramatic shift to the edge computing paradigm, IoT manufacturers need help from research community to tackle these issues.

1.2 Research Motivation

Most of the devices belonging to IoT network might require short response time, private data, produce big quantities of data, and have specific concerns with the privacy of transferred data. Thus, there is a strong belief among IoT manufacturers that the transition to edge computing is no longer an option but a necessity. The success of the edge computing compulsory transition is tightly coupled with the design of high-performance embedded devices.

An embedded system is a dedicated computing system combining software and hardware. A particular class of embedded systems are real-time embedded systems (hereafter named RTES). The output correctness of a given RTES depends, not only, on the functional result, but also on the time at which this result has been produced. These complex computing systems require sophisticated hardware, software, and design methodologies. One of the main focuses of this dissertation goes to the optimization of the hardware design and the way the

software is deployed on while considering its timing requirements. When considering the limitations associated with the design of RTES based on a single processor where increasing clock frequency is prohibitive after reaching the wall of 4 Ghz, companies and researchers were trying to find an alternative. Multi-processor was, therefore, the natural evolutionary step to keep up with the ever-increasing performance. In this context and given the key role of RTES, the problem is twofold ; suggest a bench of tools helping designers during the exploration of design alternatives, prototyping and build time-efficient data mining techniques allowing smarter functioning of these devices.

Given the complex specification of RTES based on multi-processor architectures and the plethora of design choices, design decisions need to be based on a systematic exploration process by tuning the design parameters. This process is performed in order to maximize system performances (e.g. slack time, response time) while minimizing non functional costs (e.g. power consumption, energy dissipated). We call this process Design Space Exploration (hereafter named DSE). Intuitively, identifying the optimal solution(s) in such space requires a brute-force approach that evaluates each platform configuration separately. This is impracticable. Owing to the complexity of the exploration process and performance evaluation cost, researchers have proposed automated DSE where exploration methods are able to take decisions and come up with the optimal solution using abstracted models able to cater to larger and complex systems. Even if automating the DSE represents an effective technique to reduce prototyping cost and time-to-market, novel exploration techniques need to be explored when larger design spaces are faced.

In addition to the efficient design of RTES devices, it is essential to consider how efficiently to manage and analyze the generated data to transform the information into knowledge for smarter decisions in the future. The mining task requires advanced and computationally effective cognitive computing techniques. Fortunately, data mining techniques avail of these voluminous data to improve their learning capabilities to discover hidden patterns. This real-time knowledge extraction process makes the mining task more challenging. In this context, several data stream clustering algorithms have been proposed to perform unsupervised learning. Data stream clustering is a useful and ubiquitous tool in data analysis that assigns on the fly incoming data into groups whose members present higher degree of similarity than others.

1.3 Problem Statement

In the context of this dissertation, our aim is to investigate, define, and implement a set of robust optimization and mining techniques to sustain the design and functioning of real-time

embedded systems.

In doing so, our research intends to provide solutions able to mitigate multiple challenges, including, but not limited to :

— The need of incremental mining approaches that help embedded systems designers to detect functioning anomalies and improve device autonomy.
— The lack of efficient and fully online clustering algorithms able to incrementally construct groups of design implementations under tight timing and memory constraints.
— The inefficient usage of computing resources when deploying software on.
— The design space defined as the set of possible partitioning and schedules is likely to be very large and checking whether each candidate satisfies the software's timing constraints in addition to evaluating its fitness is unrealistic.
— Conventional search heuristics were initially defined as a general exploration algorithm based on blind and pseudo-random operators. It is commonly admitted that the use of these operators is quite poor for an efficient exploration of vast design spaces.
— At the end of the optimization process, designers (decision makers) always face Pareto Fronts including a large number of sub-optimal solutions from which selecting the most proper system implementation is potentially tough.
— Designer's preference, if taken into account, can help to push the search engine to focus on specific regions of the design space.

Tackling these challenges has the potential to substantially advance of the frontier of knowledge in the fields of embedded systems design benefiting IoT network performances.

1.4 Research Objectives

To alleviate the challenges described in the previous section, the global objective of this thesis is to propose new techniques and methods helping designers to efficiently design complex applications on modern parallel architectures and ameliorate their behavior once in the market by exploiting the streaming data generated by IoT devices.

The main objectives of the thesis are :

— Conceive a novel fully online data stream clustering algorithm allowing better usage of embedded real-time devices.
— Accelerating the design stage and improving the exploration of the solutions space.
— The evaluation of the efficiency of several heuristics to choose the most appropriate one depending on the problem structure and the trade-off between conflicting metrics.
— Define an improved evolutionary algorithm for code mapping optimization that will

guide the search for the optimal parameter set for optimized code deployment.
— Introduce an automated approach to systematically help designers (decision makers) to efficiently choose their preferred solutions after the optimization process from the obtained Pareto Fronts.

1.5 Thesis

The main contribution of this dissertation is in : (1) online data stream clustering algorithm allowing smarter functioning and design of IoT devices ; helping designers to select the suitable implementation resulting from the optimization process and ; (3) developing automated and intelligently guided design space exploration engine based on evolutionary algorithms.

1.6 Thesis Contributions

This dissertation contributes to the area of real-time embedded systems design and functioning from various perspectives. Specifically, it introduces novel approaches and techniques to help designers during the design process by automation, optimization, improvement and data mining for smarter functioning.

As depicted in Figure 1.2, the main contributions of this thesis are :

1. The definition and implementation of a new fully online data stream clustering algorithm.
2. The implementation of a fully automated design space exploration process based on multi-objective evolutionary algorithm to dispatch the execution of real-time applications on heterogeneous multi-processor systems.
3. The proposition of a hypervolume-based approach to help designers choose the preferred implementation from the obtained Pareto front.
4. The definition of a schedulability guided exploration engine based on an improved evolutionary algorithm.

These contributions are briefly described in the next section dedicated to the list of publications.

1.7 List of Publications

The core chapters of this dissertation are based on a set of published and submitted papers listed and briefly described in this section. The publications are divided into two parts :

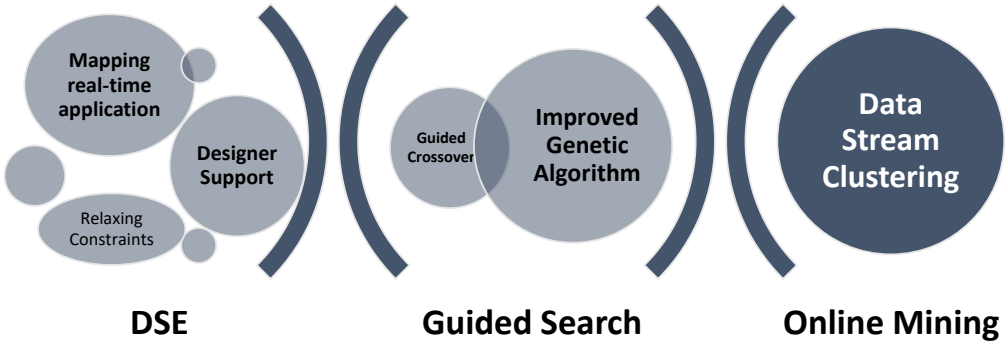


Figure 1.2 Overview of the main contributions in this dissertation.

(1) Six journal papers containing the main contributions. Four of these paper represents the basics of this dissertations. Four secondary papers presented in international conferences are given later.

1.7.1 Journal Papers

— Ayari, R., Hafnaoui, I., G. Beltrame and G. Nicolescu (2017).DeDaSC : Delaunay Triangulation-based Data Stream Clustering Algorithm for the Internet of Things IEEE Transactions on Emerging Topics in Computing).

In this paper (Ayari et al., 2017a), we introduce a new Delaunay triangulation-based Data Stream Clustering algorithm (DeDaSC) able to construct arbitrary shaped clusters on the fly without any prior knowledge of the number of clusters and their shapes. DeDaSC requires only the newly arrived data, not the entire dataset, to be saved in memory. An incremental Delaunay triangulation is introduced to guarantee the effectiveness of the neighborhood discovery with limited computation time and memory space. DeDaSC has been developed with the aim of smarter and autonomuos working of IoT devices based on the generated data. 2

— Ayari, R., Hafnaoui, I., G. Beltrame and G. Nicolescu (2017). ImGA : An Improved Genetic Algorithm for Partitioned Scheduling on Heterogeneous Multi-core Systems. Journal of Design Automation for Embedded Systems (DAEM).

The endeavor of this paper (Ayari et al., 2017b) is to propose an Improved Genetic Algorithm, named ImGA, to guide the exploration process and extensively ameliorate the conventional GA findings. To do this, we are proposing a climbing hill repairing strategy for the population initialization. This strategy aims on adjusting the randomly generated solutions by incorporating a local search algorithm. Later, we integrated the schedulability-guided crossover operator proposed in (Ayari et al., 2016a) to be part of our improved algorithm. Then, maintaining a vast population diversity is crucial to ensure that the design space was appro-

priately explored and avoid premature convergence by stagnating on sub-optimal solutions. To face this problem, we propose two main contributions ; (i) a circular mutation operator switching the workload of the heaviest and lightest processors (ii) and a restart technique based on injecting randomly generated solutions at advanced stages of the optimization engine giving a new impetus to the remaining generations.

— Ayari, R., Nikdast, M., Hafnaoui, I., G. Beltrame and G. Nicolescu (2017). HypAp : a Hypervolume-Based Approach for Refining the Design of Embedded Systems. IEEE Embedded Systems Letters (ESL).

This paper presents HypAp (Ayari et al., 2017c), a hypervolume-based automated approach to systematically help embedded systems designers choose their preferred solutions after the optimization process. HypAp first clusters the Pareto Front solutions, and then seeks a Reduced Pareto Front that maximizes the hypervolume. Furthermore, several quality indicators, including hypervolume, non-uniformity, concentration, and outer diameter, are defined to assess the effectiveness of HypAp when it is applied to various types of Pareto Fronts obtained from different multi-objective optimization problems. We further apply HypAp to a case study of an NoC mapping problem with three competing objectives, and, employing the defined quality indicators. The result of this work is critical for the design of complex embedded systems, in which designers require to choose from Pareto Fronts including a large number of near-optimal solutions.

— Ayari, R., I. Hafnaoui, A. Aguiar, P. Gilbert, M. Galibois, JP. Rousseau, G. Beltrame, G. Nicolescu (2016). Multi-objective mapping of full-mission simulators on heterogeneous distributed multi-processor. Journal of Defense Modeling and Simulation (JDMS).

In this paper (Ayari et al., 2016b), a methodology based on a multi-objective evolutionary algorithm, NSGA-II, was developed to search the design space for a set of optimized mapping configurations that trade-off three performance metrics ; makespan, communication cost and memory balancing. The target architecture considered is a heterogeneous distribute multi-processor platform. This work demonstrated that the methodology is able to find a set of optimized solutions that will help in the system integration.

— I. Hafnaoui, Ayari, R., G. Nicolescu and G. Beltrame (2017). Execution Order Assignment Through Data Criticality in Hard Real-Time Dataflow Graph-based Systems. Journal of Design Automation for Embedded Systems (DAEM).

In this paper (Hafnaoui et al., 2017a), we propose an approach that will facilitate the process of assigning execution order to different components without inherent knowledge of the

function and behaviour of components. Thus, there is a need to transform the DFG into a DAG. To that endeavour, we propose a method based on the idea of propagation of error to eliminate the cycles by taking the dependencies between the components and their in uence on the system into consideration. We introduce the concept of criticality of data that will aid in deciding which edges to discard to transform the graph into a DAG.

— I. Hafnaoui, Ayari, R., G. Nicolescu and G. Beltrame (2018). Time is of the Essence : Spreading Information among Interacting Groups. Proceedings of the National Academy of Sciences (PNAS).

Animal behavior is greatly influenced by interaction between peers as well as with the environment. Understanding the flow of information between individuals can help deciphering their behavior. This applies to both the microscopic and macroscopic levels, from cellular communication to coordinated actions by humans. The aim of this work is to provide a simple but sufficient model of information propagation to learn from natural coordinated behavior, and apply this knowledge to engineered systems. We develop a probabilistic model to infer the information propagation in a network of communicating agents with different degrees of interaction affinity. We focus especially on the concept of consensus, and estimate the time needed to reach an agreement between all agents. We experiment using swarms of robots to emulate the communication of biological and social media groups.

1.7.2 Conference Papers

— Ayari, R., Hafnaoui, I., G. Beltrame and G. Nicolescu (2016). Simulation-based Schedulability Assessment for Real-Time Systems. Summer Computer Simulation Conference (SCSC).

The aim of this paper (Ayari et al., 2016c) is to reduce the pessimism introduced by the use of schedulability tests in the context of a real-time task allocation problem. We introduce a simulation-based approach that reduces the pessimism to achieve better findings. Our experimental results show that the proposed simulation-based approach can identify a considerable amount of valid task sets that would otherwise have to be treated as unschedulable by existing methods.

— Ayari, R., Hafnaoui, I., G. Beltrame and G. Nicolescu (2016). Schedulability-Guided Exploration of Multi-core Systems. Rapid Systems Prototyping (RSP), published.

In this paper (Ayari et al., 2016a), we propose a guided search strategy based on a novel crossover operator for partitioned scheduling which incorporates certain knowledge of the rate monotonic schedulability test. The work can be applied to any other schedulability test

and even for dynamic algorithms such as the earliest deadline first algorithm. With the help of the guided search strategy, is capable of finding near-optimal solutions for the mapping problem on heterogeneous multi-core systems. hence, it can act as a powerful tool for the solution space exploration in the context of partitioned scheduling.

— I. Hafnaoui, C. Chao, Ayari, R., G. Nicolescu and G. Beltrame (2017). An Analysis of Random Cache Effects on Real-Time Multicore Scheduling Algorithms. Rapid Systems Prototyping (RSP).

In this paper (Hafnaoui et al., 2017b), we offer a framework that incorporates contention consciousness in the timing analysis as well as scheduling algorithm design and hence a tool to optimize the hardware/software structures. To do this, we introduce a compact random cache model for multilevel caches on real-time multicore systems to estimate interleaving reuse distance in shared caches. Then we calculate the cache hit probability and the Cycle Per Instruction (CPI) of tasks sharing a cache which makes our model able random cache model into the scheduling simulator. This integration helps to evaluate scheduling algorithms in the presence of shared random caches.

— I. Hafnaoui, Ayari, R., G. Nicolescu and G. Beltrame (2016). Regression-based Model Generator for Software Performance Estimation. Summer Computer Simulation Conference (SCSC).

With complex real-time systems development, a gap is created between system developers and integration experts due to a limited knowledge in software engineering. The lack of data needed to integrate the system properly, ensure the timing requirements and optimize performances was motivation to propose the work presented in this paper (Hafnaoui et al., 2016). A model generator based on regression analysis is introduced to estimate the speed-up expected when migrating from a reference architecture, such as a component developer's machine, to a final target architecture to which the integrated system is deployed.

1.8 Thesis Outline

This thesis is divided into 10 chapters. Chapter 2 critically reviews the important background material and related works that are used in this thesis. In chapter 3, we formally model the scheduling problem tackled in the different stages of this thesis. In Chapter 4, we present and synthesize the set of articles presented in Chapters 4, 5, 6 and 7 in terms of both methodology and results.

In addition to these two chapters, other parts of this thesis are presented as four journal publications (research papers) which are included in Chapters 5, 6, 7 and 8. The chapters

are organized as follows.

In Chapter 5, we introduce a new Delaunay triangulation-based Data Stream Clustering algorithm, called DeDaSC, able to construct arbitrary shaped clusters on the fly. DeDaSC requires only the newly arrived data, not the entire dataset, to be saved in memory.

Chapter 6 presents a multi-objective optimization approach based on the model-driven design paradigm allowing us to map a set of inter-communicating real-time tasks making up the Full-Mission Simulator model onto a heterogeneous distributed multi-processor system model.

Chapter 7 introduces a new posteriori technique, called HypAp, to help system designers to effectively choose their preferred solutions, from a refined representation of the Pareto Front optimal solutions.

Chapter 8 illustrates a guided DSE approach. A guided exploration prioritizes fitter solutions to be part of next generations and avoids exploring unpromising configurations by transmitting a set of predefined criteria from parents to children.

Finally, chapter 9 discusses the techniques proposed in previous chapters, the conclusions, and possible avenues for future work.

CHAPTER 2 LITTERATURE REVUE

It is now more than a quarter of a century since researchers started publishing papers on DSE and how to efficiently dispatch real-time computation across the execution resources of parallel architectures. This section includes a broad overview of the basic concepts applied in this thesis and a survey of most relevant related literature. This chapter will be devised in three main sections. First, we start by introducing embedded real-time systems from software and hardware perspectives. The focus in this part goes to the definitions and theoretical background of real-time systems and multi-processor architectures. This section is therefore of crucial importance and will provide the basis for better understanding of the following parts and get familiarized with. Then, after a brief introduction of multi-objective optimization techniques, we summarize the results of work related to the scheduling of real-time applications with different criticalities on mono-/multi-processor execution platforms. Since partitioned scheduling strategy is adopted in this dissertation, a special focus has been laid on mapping strategies. Lastly, since we strongly believe that succeeding the clustering of the generated implementations on the fly, sorely helps to guide the design space exploration towards specific (i.e. based on designer preferences) regions of the solution space. Also, online mining of the generated data enhances these systems functioning. We discuss the main classes of online (i.e. data-stream) clustering techniques.

2.1 Embedded Real-time Systems

2.1.1 Real-time Systems

Systems are referred to as real-time when their correct behavior depends not only on the proper functioning of the operations they perform, but also on the time at which they are performed by respecting the system's deadline (Davis and Burns, 2011a). A response that occurs too late to meet its deadline will be useless or even result in catastrophic consequences. Unlike other traditional systems that have a separation between timing correctness and performance, real-time systems try to make a compromise between the two where timing correctness and performance are tightly coupled. Today, real-time computing plays a primordial role in our society as an increasing number of complex systems rely, partially or completely, on computer control.

2.1.1.1 Timing Constraints

Real-time systems can be categorized as either hard or soft depending on the criticality of timing constraints and the consequence of missing a deadline (Davis and Burns, 2011a). We can describe hardness using a value function of time to indicate solution validity evolution.

— We consider a real-time system as hard when it is compulsory that an system's response must be performed within a strict deadline i.e. missing a deadline is considered as a total system failure.
Figure 2.1 shows the task value function of a hard real-time system.

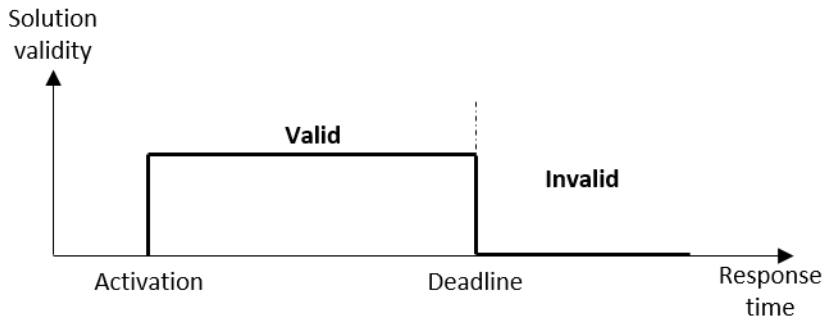


Figure 2.1 Hard real-time task value function. Once a deadline is missed, the real-time system switches immediately to an invalid state. This event may have catastrophic consequences.

— A real-time system is considered as soft when missing deadlines cause only some performance degradation. Figure 2.2 shows the task value function of a soft real-time system.

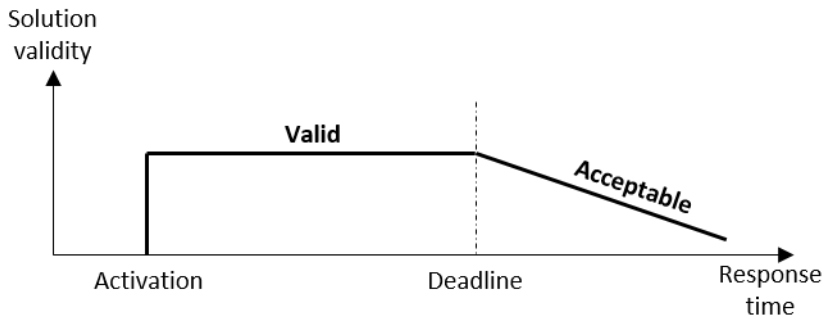


Figure 2.2 Soft real-time task value function. After missing a deadline, the real-time system still in an acceptable state during a given time laps.

2.1.1.2 Priority and Preemption

Task's priority defines the execution order of tasks in priority-based scheduling policies. We distinguish two priority assignment approaches ;

- Static priority-driven scheduler where priorities are assigned statically to each task and do not change over time.
- A dynamic priority-driven scheduler can assign, and possibly also redefine, process priorities at run-time.

Preemption is a very helpful mechanism to support priority-based scheduling. Regardless of the tasks' behavior, it is important to state that ;

- Some of them, once elected for execution, may be interrupted to allocate the processor to another one. Due to such behavior, they are called preemptive.
- On the other hand, when an elected task should not be interrupted before the end of their execution, it is called non-preemptive.

It is worth to mention that schedulers involving preemption can achieve better performance than non-preemptive scheduling algorithms (e.g. preemptive Earliest Deadline First (EDF) can reach a utilization rate equal to 1 while non-preemptive EDF cannot (Stankovic et al., 2012)).

2.1.1.3 System Periodicity

Arrival patterns of real-time tasks determine whether those tasks will be treated as periodic, aperiodic or sporadic.

- A periodic task has ready times for its jobs separated by a constant rate. The amount of time between each iteration of two consecutive activations is called period.
- Aperiodic tasks respond to randomly arriving events. Aperiodic tasks characterized by a minimum inter-arrival time are called sporadic.
- Sporadic tasks are real-time tasks which are activated irregularly with some known bounded rate. The bounded rate is characterized by a minimum inter-arrival period.

2.1.1.4 Schedulability Analysis

The schedulability analysis of a real-time system verifies its temporal correctness under a specific scheduling policy. They are concerned with deciding whether it is possible to allocate to each task a processor time equal to its execution requirement. In a real-time system, a process or task has schedulability ; tasks are accepted by a real-time system and completed as specified by the task deadline depending on the characteristic of the scheduling policy on

a given hardware architecture.

Definition 2.1.1 *A taskset τ is considered schedulable if all tasks meet their deadlines.*

A set of schedulability tests have been developed and widely used in literature to verify schedule validity. These schedulability tests are classified in three categories ; sufficient, necessary and exact.

Definition 2.1.2 *Sufficient test : If test is passed, then tasks are definitely schedulable. If test is failed, tasks may be schedulable, but not necessarily*

Definition 2.1.3 *Necessary Test : If test is passed, tasks may be schedulable, but not necessarily. If test is failed, tasks are definitely not schedulable*

Definition 2.1.4 *Exact Test (Necessary + Sufficient) : The task set is schedulable if and only if it passes the test.*

2.1.2 Embedded Multi-processor Systems

The trend in the design of modern hardware platforms has shifted towards increasing the number of processing elements. The rise of multi-processor architectures reshaped computer engineering, and long-lasting certainties had to be updated. Moore's law (Davis and Burns, 2011a) shifted its focus from the number of transistors to the number of processors that could be integrated on a chip and Amdahl's law (Hill et al., 2008) was no longer sufficient to describe speed-ups gained by parallelization. Multi-processor systems are being accepted in a wide spectrum of disciplines. As a result, the design trend from single-processor real-time systems to multi-processor real-time systems is inevitable. Multi-processor design offers a number of advantages over the traditional single-processor design. One of them is that multi-processor can exploit parallelism, which is one of the most effective ways to address the power issue, while maintaining high performance with lower voltage and frequency.

Among multi-processor architectures, we distinguish three main classes :

— Homogeneous : All the processors of the parallel execution platform have identical characteristics and are therefore perfectly interchangeable.
— Uniform : Each processor is characterized by its computing capacity or relative speed : when a job is running on a computing capacity s processor during t time units, it performs $s \times t$ work units.

— Heterogeneous : Heterogeneous computing refers to systems that use more than one kind of processing elements. These systems gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar coprocessors, usually incorporating specialized processing capabilities to handle particular tasks. This type of architecture was developed to meet the high computation and timing requirements of real-time systems and have been widely used in industry, covering different domains such as aerospace, automotive and avionics. One recent new opportunity in this multi-processor era is provided by heterogeneous systems : in (Hill et al., 2008), Hill and Marty claimed that "asymmetric multiprocessor chips can offer potential speedups that are much greater than symmetric multiprocessor chips (and never worse)". In (Van Craeynest et al., 2012a) authors stated that a heterogeneous architecture composed by a single big processor and few smaller processors "can enable higher performance and reduced energy consumption".

With the shift from single processor to multi-processor design come along new challenges. One such challenge is directly dependent on how to efficiently utilize the hardware resource on multi-processor platforms. Therefore, software plays a more important role in the multi-processor era. The improvement in performance gained by the use of a multi-processor systems depends widely on the way software algorithms will exploit it and efficiently distribute and schedule the workloads between its processing elements.

2.1.3 Multi-objective optimization

Problems with more than one objective have the distinction of being much more difficult to treat than their mono-objective equivalent. The difficulty lies in the absence of a ranking criteria to compare solutions. A solution may be better than another on some objectives and worse on others (Yuanlong Chen and Ma, 2012). The solutions found by a multi-objective optimization approach have to be optimal with respect to distinct objectives, typically conflicting. It is not possible to find an optimal solution that satisfies all objectives but rather a pool of efficient solutions characterized by the fact that their cost cannot be improved in one dimension without being worsened in another as depicted in Figure 2.3. That is why the concept of optimal solution becomes less relevant in multi-objective optimization. These solutions form the Pareto optimal front referring to the economist Vilfredo Pareto (Ehrgott, 2012).

Mathematically, the multi-objective optimization problem is defined as follows :

Ω = Decision Space ; Ψ = Solution Space
$X = (x_1, x_2, ..., x_n) \in \Omega$; X is a vector of n decision variables

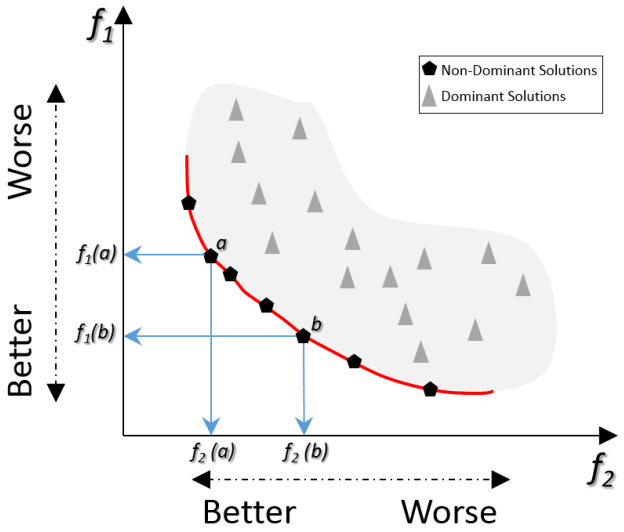


Figure 2.3 Example of a Pareto frontier. The triangle points represent feasible choices, and smaller values are always preferred to bigger ones. Triangle points are not on the Pareto front because they are dominated by pentagon points. Points A and B are not strictly dominated by any other, and hence do lie on the Pareto front.

$F = (f_1, f_2, ..., f_m)$; m is number of objective functions

$$Y = F(X) = (f_1(X), f_2(X), ..., f_m(X)) = (y_1, y_2, ..., y_m) \in \Psi \tag{2.1}$$

In the literature, many approaches have been developed to address this problem and could be classified into two main categories :

Scalar or weight-based approach. Weight-Based approach consists of formulating a single-objective optimization problem such that its optimal solutions are optimal solutions to the multi-objective optimization problem. This technique is one of the oldest techniques in multi-objective optimization using heuristics such as Genetic Algorithms (GAs) (Syswerda and Palmucci, 1991), (Jakob et al., 1992), (Yang and Gen, 1994) and Simulated Annealing (Serafini, 1992). Since setting a weight vector leads to a single point, to find different solutions with various trade-offs, the optimization process is performed with different weight vectors which produce an extensive computational cost and the decision maker have to set the most suitable weight combinations to reproduce a representative part of these Pareto solutions. Furthermore, a main technical shortcoming of this approach is that the non-convex points of the pareto front are unreachable.

Pareto approach. The Pareto approach directly uses the concepts of dominance in the solutions generation. Therefore, the Pareto optimum gives more than a single solution, but rather a set of solutions called non-dominant solutions. The main advantage of these approaches

is the simultaneous optimization of conflicting objectives. NSGA (Srinivas and Deb, 1994), NSGA-II (Deb et al, 2002), SPEA (Zitzler and Thiele, 1999) and SPEA-II (Guliashki et al., 2009) are among the most known multi-objective algorithms based on this technique.

2.1.4 Real-time Scheduling

In real-time environments, scheduling algorithms ask for an order according to which the tasks are executed such that various constraints (deadlines) are satisfied (met). Efficient scheduling policies look for higher utilization of available resources to achieve a target quality of service (Sha et al., 2004). Depending on the number of processing units of the execution platform, we distinguish two classes of scheduling policies.

2.1.4.1 Uni-processor Scheduling

Many researchers have been extensively conducted on single processor real-time scheduling algorithms. Two classic real-time scheduling policies which are considered as fundamentals for uni-processor scheduling and and also play a key role in implementing multi-processor scheduling algorithms.

— Rate Monotonic (RM) (Lehoczky et al., 1989), (Klein et al., 2012) is among the most effective uni-processor real-time scheduling algorithms. It belongs to the class of fixed-priority preemptive scheduling algorithms on uni-processor. The static priorities are assigned according to the cycle duration of the job, so a shorter cycle duration results in a higher job priority. RM is one of the most widely studied and used in practice (Halang and Stoyenko, 1994), (Mattai and Joseph, 1995), (Krishna, 1999), (Stankovic and Ramamritham, 1988). It is demonstrated by Liu and Layland (Liu and Layland, 1973) that RMS is the optimal scheduling policy among all fixed-priority scheduling algorithms, i.e., if a task set is schedulable, then RMS can successfully schedule that task set. They also formally proved that a feasible schedule by RM can be found if the processor utilization is less or equal to 0.7.

— Earliest Deadline First (EDF) algorithm is an optimal dynamic priority driven algorithm in which higher priority is assigned to the task's job that has earlier deadline, and a higher priority task always preempts a lower priority one(Halang and Stoyenko, 1994), (Mattai and Joseph, 1995), (Krishna, 1999), (Stankovic and Ramamritham, 1988). Due to those characteristics, the processor can achieve a utilization bound up to 1.

2.1.4.2 Multi-processors Scheduling

The development of appropriate scheduling algorithms for multi-processor platforms is not trivial and an algorithm which is optimal for uni-processor is not optimal anymore for multi-processor system (Baker, 2005). Efficient scheduling of real-time applications onto multi-processor systems is an NP-hard problem and solving it requires the use of meta-heuristics to find sub-optimal solutions instead of the optimal ones. Multiprocessor scheduling techniques fall into two main categories :

— Global scheduling

Global scheduling algorithms store the tasks that have arrived, but not finished their execution, in one queue which is shared among all processors as depicted in Figure 2.4. Specifically, the majority of the previous work on global scheduling have been focusing on job-level migration, where jobs of different tasks may preempt on one processor and later resume on another processor and jobs from the same task may execute on the same processor or different processors. Global strategies have a bench of disadvantages as compared with the partitioning strategies. Partitioning usually has a low scheduling overhead compared to global scheduling, because tasks do not need to migrate across processors. Furthermore, for global scheduling, a job that is preempted on one processor and later resumed on another can potentially result in additional communication costs and cache misses which is not the case for partitioned scheduling.

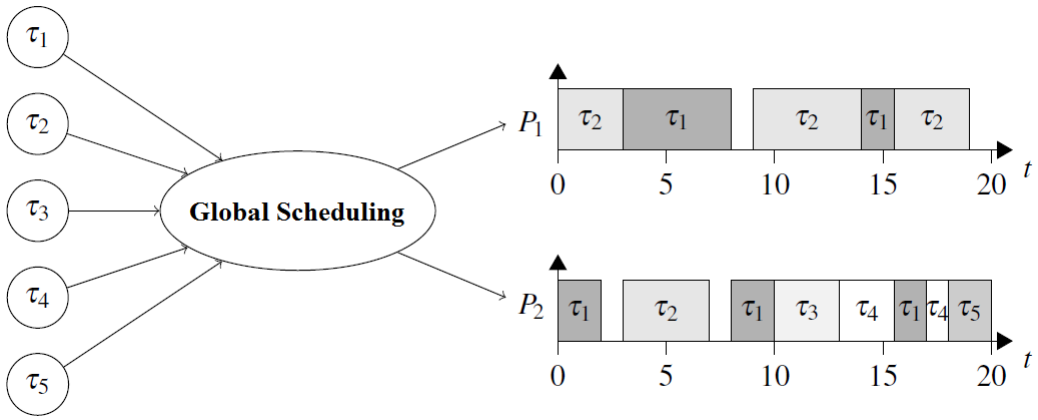


Figure 2.4 Graphic representation of global scheduling strategy (Cheramy, 2014). Only one queue is shared between all processors.

— Partitioned scheduling

For systems that contain more than one processor, we not only should decide about

the appropriate scheduling algorithm on each processor, but also we have to specify the allocation algorithm which assigns the tasks to the available processors. It is a partitioned algorithm in which migration is prohibited, i.e. tasks and all of their jobs are statically assigned to only one processor ; Hence the multiprocessor scheduling problem is transformed to many uni-processor scheduling problems. In such approach, as depicted in Figure 2.5, scheduling tasks in a multi-processor system can be seen as a two-stage problem (Baker, 2005), (Bastoni et al., 2010). For each job of every task one should be able to determine :

- on which processor the job should be executed (the so-called allocation problem) ;
- the time at which the job should be executed (scheduling problem), with respect to the other jobs executing on the same processor,

In the context of a partitioned algorithms, the allocation problem can be solved by mapping every task in the taskset to exactly one PE. In general, the number of possible partitionings of a taskset of size N on a M-processor processor is exponential in the number of tasks (M^N). Therefore, it is often impossible to find the best (with respect to a given performance metric) partitioning by exhaustive search in a constrained amount of time. This problem can be seen as a variant of the traditional bin packing problem (Johnson et al., 1974), which consists of deciding how to place N packages with fixed volumes v_i's in the minimum number of bins of volume V. This problem is known to be NP-hard (Coffman et al., 1997), (Johnson et al., 1974), and numerous heuristics have been proposed for its solution. Among the simplest of these heuristics, there are the *fist-fit*, the *worst-fit* and the *best-fit* algorithms (Johnson, 1973).

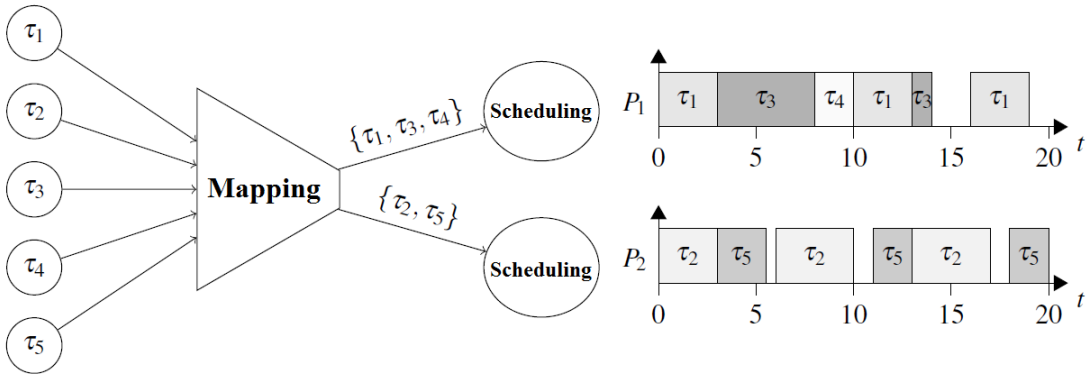


Figure 2.5 Graphic representation of partitioning scheduling strategy (Cheramy, 2014). Each processor has its own queue to schedule mapped tasks.

2.1.5 Scheduling on Multi-processor Systems

This section reviews some of the most important research works in the field of scheduling for multi-processor systems. In (Hou et al., 1994), authors use a genetic algorithm approach to implement a multiprocessor scheduler, showing that it can achieve an error in the range $0.6 - 17.5\%$ with respect to the optimal scheduling. Kwok and Ahmad describe 27 algorithms for the scheduling of programs represented by directed task graphs on identical multi-processors, a problem which is NP-complete in general (Kwok and Ahmad, 1999). The paper recognizes that the problem of extending these approaches to heterogeneous systems is nontrivial. Kianzad and Bhattacharyya review different methodologies for the partitioning and mapping of tasks to processors, discovering that no clustering algorithm strictly dominates the others but that the quality of the clustering is crucial to the overall performance of the scheduler (Kianzad and Bhattacharyya, 2006). Zhuravlev *et al.* extend the discussion about multi-processor scheduling with a survey of approaches taking into account shared resources, noting that multi-processor systems performance cannot be accurately predicted without considering contention (Zhuravlev et al., 2012a). In (Koufaty et al., 2010), authors propose "bias scheduling" as a way to cope with heterogeneous systems. The architecture considered in (Koufaty et al., 2010) is composed by a single large processor and three smaller processors. Bias scheduling affects which jobs should be migrated to which processor when the system is imbalanced, and it can be integrated on top of any other scheduling algorithm. In (Van Craeynest et al., 2012a), authors conducted their research on the same platform used in (Koufaty et al., 2010), introducing a more sophisticated model to predict the performance of a task on a specific processor without having to execute it. Braun *et al.* compared 11 heuristics for the scheduling of tasks on a heterogeneous set of machines, a problem whose optimal solution is, in general, NP-complete (Braun et al., 2001). Experimental results showed that genetic algorithms consistently performed best. Other heuristic and biologically-inspired methodologies for scheduling in heterogeneous multi-processors can be found in (Tumeo et al., 2008) and (Sih and Lee, 1993). Davis and Burns (Davis and Burns, 2011b) provide an excellent survey of real-time scheduling algorithms for multi-processor systems. They review both partitioned and global approaches, noting that the use of partitioned scheduling allows to exploit a wealth of results on schedulability and optimality for single-processor systems. However, heterogeneous systems are explicitly excluded in this survey.

In recent years, however, we have seen an increasing number of publications dealing with the real-time scheduling of tasks over a particular type of multi-processor architectures : uniform multi-processor systems. In uniform architectures, according to the taxonomy given by Davis and Burns (Davis and Burns, 2011b), each task receives an equal speed-up s moving from

one processor to another. Funk and Baruah provide an approximate utilization bound for the partitioned scheduling of periodic real-time tasks in such systems (Funk and Baruah, 2005). Niemeier *et al.*, extend the analysis of approximate solutions to tasksets composed by independent jobs and shared memory multiprocessors (Niemeier et al., 2011). In (Raravi and Nelis, 2012) and (Raravi et al., 2012),authors focus on an even more specific family of architectures : two-type heterogeneous multiprocessors. They provide polynomial complexity algorithms to solve approximately the task-to-processor assignment problem in this context.

In literature, few works deal with the scheduling problem of fully heterogeneous multi-processor systems, which is addressed by our work. In (Moreira et al., 2007a), authors present a real-time scheduling algorithm for heterogeneous multi-processors mixing time-division multiplexing and static-order scheduling. This is a global static-priority algorithm that requires the solution of a linear-programming optimization problem. Qin and Jiang (Qin and Jiang, 2005) propose a heuristic-based, reliability-driven real-time scheduling (global and dynamic-priority) algorithm for heterogeneous multi-processor that "reduces reliability cost by up to 71.4%".

2.1.6 Mapping Strategies

Since the introduction of multiprocessor systems into the market, a lot of effort was put into exploring new techniques to fully exploit these hardware platforms. As systems grow in complexity, different performance requirements are needed of the hardware. To cater to this, heterogeneous multi-processor systems have become increasingly involved and methodologies that could efficiently allocate software components to these types of platforms highly demanded. Here, we classify these techniques into (1) static ; methodologies that perform the allocation at design time and cannot therefore be altered afterwards, and (2) dynamic, which involve techniques that map the tasks at run-time.

Design-time mapping strategies are applicable to static scenarios where the system behavior such as computational load and communication are known ahead of time and the hardware platform considered is static. These types of systems are not expected to deal with new applications at run-time as is the case with multimedia applications.

Dynamic techniques, on the contrary, face the challenge of mapping applications onto a hardware platform at run-time while considering new incoming tasks or a change in system parameters and adjusting the mapping to satisfy the resource requirements. Although our research scope focuses on systems with static behaviour, we mention a few works that tackled the mapping problem at run-time.

Heuristics were proposed in (Mehran et al., 2008), (ter Braak et al., 2010), (Hong et al., 2009), (Carvalho, 2010) to map tasks at run-time when new applications arrive and no prior knowledge of the behaviour of these applications. In (Mehran et al., 2008)(ter Braak et al., 2010), authors propose a dynamic spiral mapping that searches the allocation of tasks in a spiral way. Mehran proposed a new strategy that aims on optimizing energy by considering a homogeneous 2D-mesh topology and placing the highest communicating task at the center of the mesh (Mehran et al., 2008) . An improvement of 29% was observed in (Hong et al., 2009) when migration of tasks to other processors in a homogeneous architecture to balance workload variation was performed. In (Brião et al., 2008), authors proposed a method that relies on Dynamic Voltage Scaling and turned off idle processors to reduce energy as well as reduce execution time. A heuristic was proposed in (Carvalho, 2010) to reduce the communication overhead on the NoC and hence decrease execution time over a heterogeneous multiprocessor architecture.

These techniques, having no prior knowledge of the new applications, can not guarantee schedulability. For such, other methods were proposed for run-time mapping that rely on static DSE where knowledge of potential future additions are considered in exploring mapping configurations and stored for run-time situations. For further reading, these works (Ykman-Couvreur, 2011), (Singh et al., 2011), (Siegel et al., 2006), (Jia and Pimentel, 2010) can be examined.

When it comes to more critical systems, static mapping techniques are widely preferred. Graph theory concepts were employed by (Khoroshevsky and Kurnosov, 2009) to partition the global task graph and map partitions onto a multiprocessor clusters system with heterogeneous network channels and differing topologies. (Xu et al., 2015) proposed a methodology based on streaming graph partitioning that is aware of the heterogeneity of the system to reduce the overall execution time of partitions. (ZHOU and XIONG, 2012) proposed a weighted Round Robin to allocate tasks and reduce power consumption. The works in (Ascia et al., 2004)(Hu and Marculescu, 2005)(Marcon, 2008) focused on reducing both the energy consumption and execution time. (Hu and Marculescu, 2005)(Marcon, 2008) concentrated on communication by reducing energy in communication and including the communication times in the optimization algorithms.

(Orsila et al., 2007), (Lin and Wang, 2005), (Ascia et al., 2004), (Choi et al., 2012), (Piscitelli and Pimentel, 2012) proposed techniques that searched a near-optimal mapping solution while optimizing a set of parameters. Simulated Annealing was used to reduce execution time, memory consumption in (Orsila et al., 2007) and throughput in (Lin and Wang, 2005) considering a homogeneous architecture. Genetic algorithms were extensively explored in

this concept. (Ascia et al., 2004) based their approach on a GA to explore pareto fronts for efficient energy consumption and optimized execution times. (Choi et al., 2012) also proposed a GA-based approach to efficiently execute Synchronous Data-flow systems on heterogeneous multi-processor architectures. (Piscitelli and Pimentel, 2012) proposed a hybrid technique that combined analytical analysis with simulation results to estimate the system throughput and employed a multi-objective NSGAii to explore the DSE and optimize execution time and system cost. NSGAii was also used in the proposed approach by (Devi and Anju, 2014) to map dependent tasks in a heterogeneous environment to minimize makespan and reliability cost. A hybrid GA integrating Particle Swarm Optimization was proposed in (Bergamini et al., 2009)(Kang and Zhang, 2012) that considered the precedence of tasks in the system. Precedence was taken into account in the works of (Miryani and Naghibzadeh, 2009a) as well where a multi-objective GA was proposed to map tasks on a heterogeneous multiprocessor system in order to optimize execution time and reduce the number of processors in the system.

2.2 Data Stream Clustering

Even if our aim at the beginning was to propose an online clustering algorithm allowing the classification of generated design implementations on the fly to incorporate designer preferences in the search engine, we went one step further. Thus we proposed a fully online clustering algorithm that have the capability discovering arbitrary shaped clusters with reduced timing and memory cost. Data stream clustering is motivated by emerging applications involving massive datasets (Guha et al., 2003). An exhaustive survey of data stream clustering algorithms and relevant applications is given in (Silva et al., 2013). It includes, for instance, bearing prognostics, forest cover, grid computing, sensor networks, network intrusion detection, stock market analysis, etc. When designing data stream clustering algorithms, several requirements have to be considered and various issues need to be addressed, mainly, in terms of run-time and memory needs. Due to the wide spectrum of application domains and model constraints, the research in data stream clustering has gained a high attraction. In below we review some of the most important related research works. Broadly speaking, we can classify the related work on data stream clustering into the following four clustering techniques, namely density-based algorithms, partitioning algorithms, hierarchical algorithms, and grid-based algorithms.

2.2.1 Density-based Algorithms

Density-based algorithms aim mainly on finding arbitrary shaped clusters and noise filtering based on the density. The basic idea behind this technique is to keep increasing cluster

cardinality i.e. number of members as long as its local density reaches a predefined certain threshold. Density reachability and density connectivity concepts were introduced to refine the algorithm's process. In the context of density-based algorithms, clusters are dense regions (i.e. maximal set of density connected data points), separated by regions of lower density. A series of density-based algorithms have been developed in literature. Hereafter, we give a brief overview of these algorithms ; DBSCAN (Density Based Spatial Clustering of Applications with Noise) has been proposed by Martin Ester, Hanz-Peter Kriegel's group in (Ester et al., 1996). DBSCAN is considered as the prototypical density-based clustering approach and the most used one. DBSCAN aims on estimating the density surrounding each data point by counting the number of members in a predefined *eps* neighborhood compared to a certain threshold to identify processor region and outliers. Later, core samples join clusters if they are density-reachable and border points are assigned to clusters. In (Ester et al., 1998), authors proposed an incremental version of DBSCAN allowing gradual modifications of the dataset. After each iteration micro-cluster connections are created and broken according to the changes. The main disadvantage of this incremental version is that the whole dataset is required to be available for each iteration or update. Another well-known algorithm is OPTICS (Ankerst et al., 1999) (Ordering Points To Identify the Clustering Structure). The basic idea behind OPTICS is similar to DBSCAN while tackling one of the most important limitations of DBSCAN consisting of the way meaningful clusters with varying density is detected. In (Hinneburg et al., 1998), authors proposed DENCLUE (DENsity-based CLUstEring) which is one of the most effective unsupervised clustering algorithms allowing the classification of voluminous data. DENCLUE is based on the concept of density closeness and the hill-climbing algorithm.

2.2.2 Partitioning Algorithms

Partitioning algorithms (a.k.a. iterative relocation algorithms) categorize the data samples into a pre-defined fixed number k of clusters. These algorithms aims on optimizing a given criterion by iteratively relocating data points by moving them from one cluster to another. The clustering is based on the following conditions : (1) each cluster contains at least one data point and (2) each data point belongs to only one cluster. To improve the partitioning, it uses an iterative relocation procedure which moves objects from one group to another. A common measure of partitioning quality is the degree of similarity (e.g. Squared Euclidean distance, Mahalanobis distance, Minkowski distance) between data samples belonging to the same cluster, as data samples of different clusters. The main drawback of such simple technique is unsuitable for discovering complex non-linear (arbitrary) shaped clusters of voluminous datasets. An incremental version of k-means has been proposed in (Macqueen, 1967). This

sequential version was adapted for streaming data and generated hyper-spherical clusters. This algorithm adds cluster centroids gradually as clusters are being formed. In (Ng and Han, 1994), authors proposed CLARANS which is an efficient medoid-based clustering algorithms based on randomized search. CLARANS draw samples of neighbours dynamically. The clustering process searches on the graph where every node is a potential solution, that is, a set of k-medoids. Generally, partitioning algorithms are time consuming and cannot be proficiently applied to the voluminous datasets.

2.2.3 Hierarchical Algorithms

Hierarchical algorithms are based on constructing a hierarchy of clusters (a.k.a. dendrogram) which iteratively divides the dataset into smaller subsets until each subset contains single data sample. This technique involves creating clusters that have a predetermined ordering from top to bottom. In this hierarchy, each node of the tree represents a group of similar data. The cluster hierarchy can be formed from leaves to root in an agglomerative approach or the opposite in divisive approach (Murtagh, 1983). The merge or split process should be stopped whenever the pre-defined stopping criterion is met. Among these algorithms we cite CURE (Clustering Using Representatives) (Guha et al., 1998) which is designed to handle large datasets that is more robust to outliers and identify clusters having non-spherical shapes. To do this, CURE employs a mixture of random sampling and partitioning. In (Karypis et al., 1999) authors proposed CHAMELEON that measures the degree of similarity between two clusters is based on a dynamic model. To group data samples, CHAMELEON aims on maximizing the intra-cluster similarity and minimizes the inter-cluster one. CHAMELEON is applicable to all types of data as long as a similarity matrix can be constructed and requires the prior knowledge of the number of clusters to be created. In (Zhang et al., 1996) authors proposed BIRCH which is an unsupervised data mining algorithm that has been applied for data stream mining. BIRCH introduced the use of micro-clustering and macro-clustering. It scans the database to build an in-memory tree before applying clustering algorithms to cluster the leaf nodes. A major limitation of hierarchical algorithms is that as soon as two data points are grouped together, they cannot move to other groups in the tree. Thus, migration is prohibited between the clusters.

2.2.4 Grid-based Algorithms

Grid-based algorithms explore multi-resolution grid data structure in clustering. It subdivides the data space into a limited number of cells to form a grid structure. This partitioning of the data space is based on the prior knowledge of data granularity. It allows the detection of

dense regions from the cells in the grid. The main advantage of this method is its scalability given that grid-based algorithms are typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space. Wang et al. proposed a STING (STatistical INformation Grid-based method) for spatial data mining (Wang et al., 1997) in the grid structure. The quality of the resulting clustering depends on the granularity of the lowest level of the grid structure. Finer granularity of the grid cells leads to higher quality of the obtained clustering and the cost of processing will increase substantially. Another well-known grid-based clustering approach is WaveCluster (Sheikholeslami et al., 1998). WaveCluster the method is not sensitive to the order of the number of input data samples to be processed. WaveCluster uses a wavelet transformation to transform the original feature space. WaveCluster is well capable of finding arbitrary shape clusters.

2.3 Conclusion

In this chapter, we presented a revision of fundamental concepts and an overview of the related research contributions that are necessary to address the dissertation objectives. This research contributions addresses two complementary areas : embedded real-time systems design optimization and online data-stream clustering. Not to weigh down the discussion with unnecessary repetition and in order to improve the quality of the synthesis brought forward, the listed articles provide extensive discussions of the aforementioned related work and recapitulatory tables comparing them to our propositions.

CHAPTER 3 PARTITIONING PROBLEM FORMULATION

In this section, we define the formulation dedicated to mapping and scheduling of real-time applications on multi-processor systems. We start by providing a formulation for real-time applications. After that, we introduce the way we model the execution platform. Then, we define a formulation for schedulability analysis. Finally, we finish with a particular formulation of the mapping problem.

3.1 Real-time Application Model

Real-time systems can be easily represented as a graph, where sub-systems (tasks) are nodes and their data dependencies are edges. We formally define the graph modeling as $G =< T, E >$, where $T = \{t_i | i \in \{1, ..., n\}\}$ is a set of n vertices, each one representing a real-time task and $E = \{e_{ij} | (i, j) \in \{1, ..., n\} \times \{1, ..., n\}\}$ is a set of edges that represents the dependencies among these tasks. The edges are weighted by the amount of data exchanged between each pair of connected tasks. Such graphs are commonly referred to as task graphs. We assume that there are n tasks and m processors belonging to different machines. Also, we assume that all the tasks are preemptive. Each processing unit can execute one task at a time, so that the system can process m tasks concurrently. Each task is defined as a tuple $\tau_i =< C_i, T_i, D_i, \Pi_i, \Phi_i >$, where : C_i is the task execution time vector modeling the expected execution time c_{ij} to complete task t_i on processor p_j, D_i is the deadline of the task t_i, Π_i its priority and Φ_i its phase. At run-time, the processor executes the task with the highest priority that has work pending. Each task τ_i generates an infinite sequence of jobs. The inter-activation times of τ_i are characterized by a fixed period T_i.

We start by defining the tasks expected execution time on the execution platform. This information is defined by specifying a $(n \times m)$ matrix named Execution Time Matrix (C), where c_{ij} is the expected execution time of the task i on processor j. This model expresses the execution heterogeneity among the execution platform resources. The elements along a row indicate the execution times of a specific task on the different processing units, and those along a column indicate the expected execution time of all tasks on a single node. For this, we represent execution times of modules with a matrix.

We also need to define the amount of data exchanged between each couple of tasks. This information is defined in a square matrix $(n \times n)$ that we called Communication Cost Matrix (CC), where $CC(i, j)$ is the the weight of the edge connecting t_i to t_j and defined as e_{ij}. This

matrix is known as the adjacency matrix. CC is an upper triangular matrix since the data exchanged between tasks t_i and t_j are the same as the amount of data exchanged between tasks t_j and t_i. The main diagonal entries are equal to zero.

Another important feature to consider in our model is the memory consumption of each task. This information is defined in a Memory Consumption Vector (M) with n elements where each term m_i describes the amount of static memory needed to execute task t_i.

The utilization rate of a task represents the fraction of time that the task consumes on a processor to complete its execution. This ratio is widely used in schedulability analysis and as performance metrics to a evaluate a partitioning and/or a schedule. A task with high utilization rate will be considered as heavy task, and instead a task which the utilization rate is low will be qualified as a light one.

— Utilization rate of task i

$$u_i = \frac{C_i}{T_i} \tag{3.1}$$

— Processor utilization

$$U_{tot} = \sum_{j=1}^{k_j} u_i \tag{3.2}$$

— Average utilization rate per processor

$$\bar{u} = \frac{u_{max}}{m} \tag{3.3}$$

— Maximum utilization rate

$$u_{max} = \max_{i=1}^{m}(u_i) \tag{3.4}$$

— Minimum utilization rate

$$u_{min} = \min_{i=1}^{m}(u_i) \tag{3.5}$$

All the aforementioned parameters shown in this section are summarized in Table 3.1

3.2 Execution Platform Model

A Heterogeneous multi-processor system is a complex yet powerful architecture widely used in industry, covering different domains such as aerospace, automotive and avionics. The multi-processor systems used in this work is *heterogeneous* in its broadest terms, i.e. a given task τ_i might execute faster on processor m_s than on PE m_t while the opposite might be true for another task τ_j. In the following, we will indicate the WCET of task τ_i on PE m_s as $C_{i,s}$. This is a generalization of both the homogeneous model, where all tasks have the same execution

Table 3.1 Real-time System Model Symbols with their Descriptions

Symbole	Description
n	Number of tasks
m	Number of processors
τ	Task set $\tau_1, \tau_2, ..., \tau_n$
τ_i	i^{th} task of the taskset τ
k_i	Number of tasks mapped to processor i
T_i	Period of task i
D_i	Deadline of task i
C_i	Worst-case execution time vector of task i
c_{ij}	Worst-case execution time of task i on processor j
Π_i	Priority task i
Φ_i	Phase of task i
u_i	Utilization of processor i
U_{tot}	Total processor utilization i
u_{min}	Minimum processor utilization
u_{max}	Maximum processor utilization
e_{ij}	Amount of data exchanged between τ_i and τ_j

time one each processor (i.e. $\forall i \in [1..N]\ \forall s,t \in [1..M]\ , C_{i,s} = C_{i,t}$), and the uniform model, where all tasks have the same speed-up going from one processor to another (Davis and Burns, 2011b) as depicted in equation 3.6 :

$$\forall i,j \in [1..n]\ \forall s,t \in [1..m]\ , C_{i,s}/C_{i,t} = C_{j,s}/C_{j,t} \tag{3.6}$$

3.3 Schedulability Analysis

In the partitioned scheduling adopted in this thesis, whenever the mapping phase is processed, the taskset assigned to each PE is scheduled accordingly using a uni-processor Rate-monotonic scheduling policy. RM is an optimal static scheduler and by far the most used real-time algorithm and it is one of the easiest policies to implement. RM is a static-priority scheduling algorithm for real-time systems (Mohammadi and Akl, 2005). It is a preemptive algorithm that assigns higher priorities to the tasks with shorter periods T_i.

$$T_i \leq T_j \Leftrightarrow \Pi_i \geq \Pi_j \tag{3.7}$$

The Liu & Layland bound (Deb et al., 2002) is a schedulability test that provides a sufficient condition for a taskset to be schedulable when using an RM scheduling policy. Liu & Layland proved that a feasible schedule (i.e. that will always meet deadlines) exists if the total processor utilization U is below a specific bound. For each task-set ζ_j composed of k_j tasks assigned to processor j, the task-set is guaranteed to be schedulable if this test yields. The necessary condition to verify the schedulability of ζ_j based on the processor utilization bound, is defined as :

$$U_j = \sum_{i=1}^{k_j} \frac{c_{ij}}{T_i} \leq k_j.(2^{\frac{1}{k_j}} - 1) \tag{3.8}$$

Alongside the Liu & Layland bound test, another sufficient schedulability test for RM scheduling was proposed by Bini et al. (Bini et al., 2003), who introduced the hyperbolic bound :

$$\prod_{i=1}^{k_j} \left(\frac{c_{ij}}{T_i} + 1 \right) \leq 2 \tag{3.9}$$

In our work, we rely on the more restrictive Liu & Layland bound defined in Equation (3.8).

3.4 Mapping Problem Formulation

Heterogeneous multiprocessor systems are growing in importance in parallel architectures (Hill et al., 2008) and have been drawing increasing attention from both industrial and research communities. These systems can guarantee a significant increase in performance by providing a large number of unbalanced parallel powerful execution resources. The improvement in performance gained by the use of a heterogeneous multi-processor system depends widely on the way software algorithms will exploit it and efficiently distribute workloads between its processing elements. Therefore, the step consisting on mapping tasks to processing elements has been extensively addressed and is in the middle of the storm for hardware and software communities. In concrete words, this step consists on finding the optimal assignment of n tasks modeling the application on the m processors of the target architecture. Since the proposed methodology rely on partitioned static scheduling, the mapping phase is a fundamental step in the optimization process. Thus, we can rely on complex, yet powerful algorithms to get closer as much as possible to the optimal solution.

This problem of assigning tasks to processors can be seen as a variant of the quadratic assignment problem, Bin packing. In this classic problem, objects of different sizes must be packed in a finite number of boxes with fixed capacities in a way that minimizes the

necessary number of boxes. In our case, the items to store are the tasks and boxes are processors that have limited capacity beyond which valid scheduling can not be assured. Sufficient schedulability tests are used in our work to verify that the chosen algorithm will be able to correctly schedule tasks on each processor. Stirling numbers of the second kind count the number of ways to partition a set of n elements into k nonempty subsets. They are denoted by $S(n, k)$.

Finding a global solution to the partitioning problem through exhaustive methods is unfeasible. The order of complexity of this approach would be $O(m^n)$, where n is the number of applications and k the desired partitions. The combination of all possible solutions is giving by the Stirling Numbers of the Second Kind as depicted in equation 3.10 :

$$S(n,k) = \frac{1}{k!} \times \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \tag{3.10}$$

Based on this equation, the mapping problem permits to leave unused processors which gives us the number of possible solutions defined by $S'(n, k)$ where :

$$S'(n,k) = \sum_{k=0}^{m} S(n,k) = \sum_{k=0}^{m} \frac{1}{k!} \times \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \tag{3.11}$$

Using a number of tasks relatively low, exact methods have been widely discussed in literature to solve this problem. However, with bigger tasksets it becomes unrealistic to rely on exhaustive search to find the best partitioning and it is better to use a heuristics, although faster, but could possibly not find a valid partition while it exists or not the best valid partitioning.

In a mathematical formulation of the mapping problem, we define the function χ that assigns each of n tasks to any of the m processors :

$$\forall i \in [1, n], \exists j \in [1, m] \ , \chi(t_i) = p_j \tag{3.12}$$

3.5 Conclusion

In this chapter we presented the formulation related to the problem of designing real-time systems (i.e. mapping and scheduling). In the next chapter, we will recap the central research questions by introducing the research methodology of this dissertation.

CHAPTER 4 RESEARCH METHODOLOGY

This chapter presents the methodology, that we propose, to proceed towards the fulfillment of all research objectives, explaining the rationale for each research phase and address the challenges outlined in Chapter 1. This chapter provides a condensed but comprehensive view of thesis methodology and main contributions.

4.1 Global View

The contribution of this dissertation provides for three significant deficiencies in the existing research in embedded computing systems design : (i) the lack of efficient and effective online clustering techniques that can be included in the search engine to make it smarter ; (ii) the fact that most of automated DSE processes rely on the blunt application of existing heuristics without incorporation of problem structure or designer preferences and (iii) the need of sophisticated approaches to help designers to choose their preferred system implementation for prototyping. Among the aforementioned list of papers (in Chapter 1), our focus, in this dissertation, goes to four main contributions as depicted in Figure 4.1.

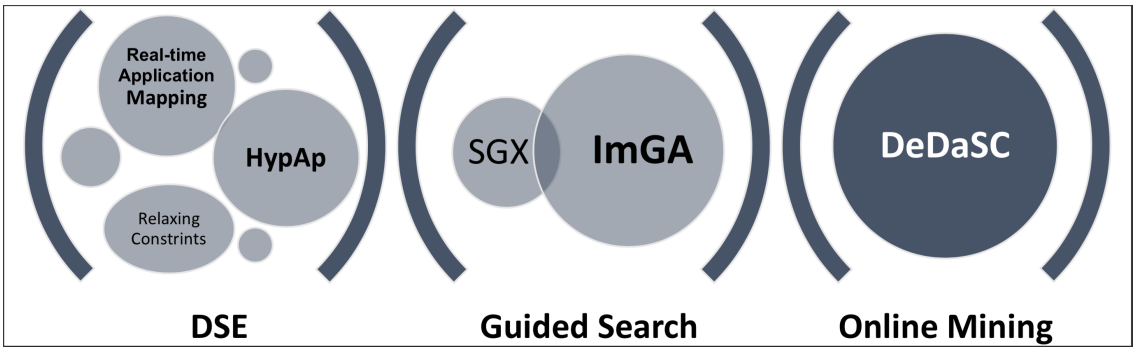


Figure 4.1 Overview of the main contributions in this dissertation.

1. At the beginning, a fully online clustering algorithm (DeDaSC) is proposed to classify IoT embedded devices' behavior. Every time DeDaSC receive a new data point, it performs a set of operations to assign it to an existing cluster or create a new one by its own on the fly.
2. Then, a multi-objective optimization evolutionary algorithm is implemented to dispatch the execution of real-time applications on heterogeneous distributed multi-processor systems while ensuring system's schedulability.

3. Later, we introduce a hypervolume-based approach (HypAp) to systematically refine the design of embedded systems. This approach helps embedded systems designers to pick the preferred implementation from a reduced Pareto front (in terms of cardinality).
4. Finally, an improved genetic algorithm (ImGA) is proposed to enhance and guide the search engine by taking into account the problem structure. The study focus on maximizing the ratio of schedulable tasks mapped to a heterogeneous multi-processor system.

The overview of the development flow of the proposed methodology is illustrated in Figure 4.2.

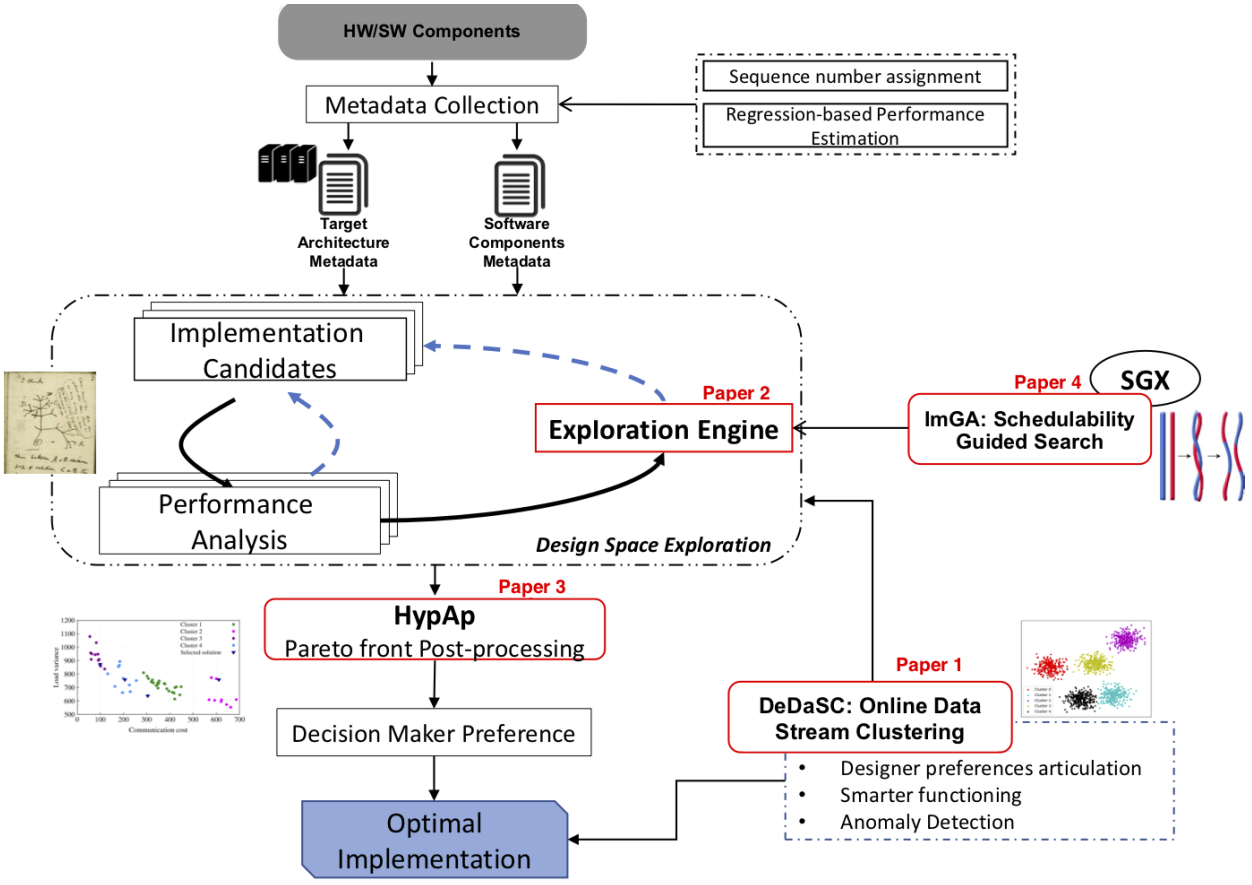


Figure 4.2 Global view of the developpemet flow of the main thesis contributions.

We divide the flow into five principle research steps : (a) The collection and estimation of metadata necessary to be able to efficiently explore the design space [1] ; (b) the techniques to explore the mapping implementations of the real-time system application onto a heterogeneous

1. This part has been developed in collaboration with the PhD student Imane Hafnaoui

multi-processor architecture ; (c) the improved search engine that guides the exploration of the design space ; (d) the decision making support incorporating designer preferences and (e) the online clustering of streaming data that can be used in different stages of embedded systems life cycle from design to usage.

4.2 Proposed Methodology

This section offers a survey of the two principal axes of research of this dissertaion. Firstly, we focus on the data-stream clustering enhancing real-time embedded systems functioning. Later, we will discuss in depth the various techniques proposed in this thesis offering designers a panoply of optimization techniques to ameliorate the design process.

4.2.1 Online Data Stream Clustering

Data stream clustering refers to the process of grouping sequence of data that is too large on the fly. Data points belonging to the same cluster presents higher degrees of similarity than those belonging to other clusters. Nowadays, streaming data are omnipresent. Examples include stock market analysis, bearing prognostics, forest cover, sensor networks, etc. The profusion of data streams in various engineering disciplines has led to new algorithmic paradigms for processing them, which often impose very stringent timing and memory requirements on the algorithm's resources.

4.2.1.1 Online Clustering Applications

In the context of embedded systems, developing efficient online clustering policies can hugely impact the quality of service (QoS) of these devices. Data-stream clustering algorithms take advantage of the previous computations and experiences to discover repeatable patterns in order to adaptively produce reliable and more autonomous devices. Tradionnaly, designers have analyzed data and adapted embedded systems to the changes in data patterns. However, as the volume of data surpasses the ability of humans to make sense of it and manually detect these hidden patterns and classes, designers cannot perform such dynamic process. The natural solution consists on automating techniques that can learn from the generated data to adapt to a shifting landscape. The use of data stream clustering algorithms is broadening to include various aspects of embedded computing systems. Consider the following :

— Smarter functioning

The huge size of streaming data generated by embedded devices represents their priceless output. In order to make these devices smarter and more adaptive, extracting

valuable knowledge from the non-stationary and unbounded streaming data is one of the principal challenges for embedded computing systems community. Data stream clustering serves as powerful tools to extract this hidden information in data stream patterns.

— Designer preferences articulation

In a priori optimization techniques, decision maker preferences are explicitly outline before launching the optimization process. A good practice is to express designer preferences as weights representing relative importance of the objectives. Once finished, the best solution according to the given preferences is found. During the exploration, online clustering can serve as a powerful tool to guide the search engine to specific regions of the design space. This approach favors the implementations belonging to clusters with higher weights to be selected for the subsequent generations.

— Aging detection

The results of behavioral changes obtained by direct application of online mining techniques can be used to compare aging to each other between successive time frames. Since embedded computing systems aging progresses over time, the occurrences of changes in signs of aging will appear gradually. Thus, detecting these changes on the fly allows the rapid discovering of suspicious behavior due to device's aging.

— Anomaly detection

Online detection of performance anomalies in embedded systems functioning. Data stream clustering helps to detect, at a fine granularity and during run-time, unknown anomalies that may still be observed in complex modern systems. Such clustering process provides designers the ability to perform root cause analysis and identify potential corrective actions.

— Security enhancement

Online clustering of generated streaming data can be used to identify suspicious behaviors. Thus, online mining is concerned with protecting these omnipresent embedded devices from corruption due to malicious software such as Trojan and viruses. The real-time detection is primordial to prevent some catastrophic situation.

4.2.1.2 DeDaSC Processing Stages

In order to cluster generated data points (i.e. solution), DeDaSC goes through a list of iterations. Our clustering algorithm addresses all these considerations and can be divided into three parts that are detailed in below : (1) neighborhood discovering using incremental Delaunay triangulation (2) online micro-clustering, and (3) macro-cluster construction. Hereafter, we give a summary of DeDaSC processing.

— Initialization

The first step consists in creating three micro-clusters to construct the first triangle of the Delaunay triangulation. For this purpose, the first three points are stored on primary memory as three different micro-clusters regardless their spatial proximity.

— Micro-cluster Assignment

Whenever a new data point is linearly scanned, the micro-clusters are updated in order to reflect the adjustments. Each data point either needs to be put in a new micro-cluster of its own, or needs to be absorbed by a pre-existing micro-cluster. To make such substantial decision with limited timing cost, we first refer to the most recent image of the Delaunay triangulation at time. In many cases, the data point p may not be sufficiently close to this micro-cluster which means that the distance is greater than the predefined magnetic attraction distance. Thus, p should be placed in a new micro-cluster of its own. If the data point p falls in a populated region, than the incoming data point p is absorbed by this micro-cluster. Subsequently, micro-cluster's metadata and the Delaunay triangulation is updated.

— Macro-clustering

The macro-clustering stage will use, only, the compactly stored mature micro-clusters. Therefore, it is not constrained by one-pass requirements. It is assumed that as input to the algorithm, the user supplies macro-cluster attraction distance to merge overlapped micro-clusters. Macro-clustering is a kind of a re-clustering process introducing a new scale at which any mature micro-cluster that will be within the macro-cluster attraction distance of the freshly updated micro-cluster will revise its membership to be the same.

By applying the aforementioned process, every incoming data sample is instantly clustered and outliers are easily identified. Such approach maintains arbitrary shaped data space regions of global clusters online. More details about the proposed data stream clustering algorithm is given in Chapter 5.

4.2.2 Design Space Exploration

It has been a decade since microprocessor companies realized that it was fruitless to work on improving processor technology by decreasing transistor size and hence increasing switching speed. Due to an increase in leakage power and thermal constraints, the technology hit what architects like to call the Power Wall (Guo et al., 2010). As a way out, manufacturers turned towards MPSoCs. This created a wave of scientific research focused on solving problems related to this topic.

Developing the expertise in the design of high-performing embedded systems hugely impact

the power and capacities of IoT netwok. These MPSoCs will have the capacity to process the generated data at the edge of the network in real time. Only a summary of these data or the most pertinent of them will pass to the higher-level computing environments (i.e. Cloud data centers). Therefore, offering new improved design techniques is of paramount importance to succeed the general trend of transition from a centralized cloud computing to a more local edge computing.

4.2.2.1 Automatic Design Space Exploration

A principal question, in the design of embedded systems, is how to efficiently and effectively dispatch the workload of the software application on the processing elements of the target execution platform. The problem becomes more complex when the system is a real-time system where system behavior need to respect timing constraints. The problem of mapping n jobs to m processing units is an NP-Hard problem. Finding a near optimal solution is acceptable since finding the best solution in sensible time is impossible. For such, meta-heuristics are widely used to solve this type of problem.

Design Space Exploration (DSE) pertains to the action of exploring design configurations, solutions or alternatives while sub-optimal decisions need to be taken. The exploration process can be performed by using mono or multi-objectives approaches. Due to the expansion in terms of number of integrated processors in embedded systems and real-time applications complexity, the number of possibile implementations that have to be considered increases exponentially. This leads to an extremely huge search space. Even if abstract models reduce simulation time by scaling down design complexity, inspecting all the possible solutions stills an NP-hard problem in which exhaustive search is unrealistic and smarter ways of doing exploration needs to be adopted. In this context, the use of meta-heuristics accelerates the exploration process. Techniques borrowed from graph theory, machine learning and data mining could be used. The aim of these enhanced approaches is to reach sub-optimal solutions with higher cost/performance ratio (i.e. quality) compared to those obtained with conventional approaches.

Owing to the complexity of the exploration process, a highly efficient option consists on automating the DSE in such a way that designer's intervention is restricted to the definition of the system model and the tuning of exploration parameters (i.e. meta-heuristic algorithm parameters). As depicted in Figure 4.3, automating the exploration process leads to a colossal reduction of processing time before getting the final sub-optimal solution(s). This significant time gain reduces, simultaneously, prototyping cost and the time-to-market of the of the final product.

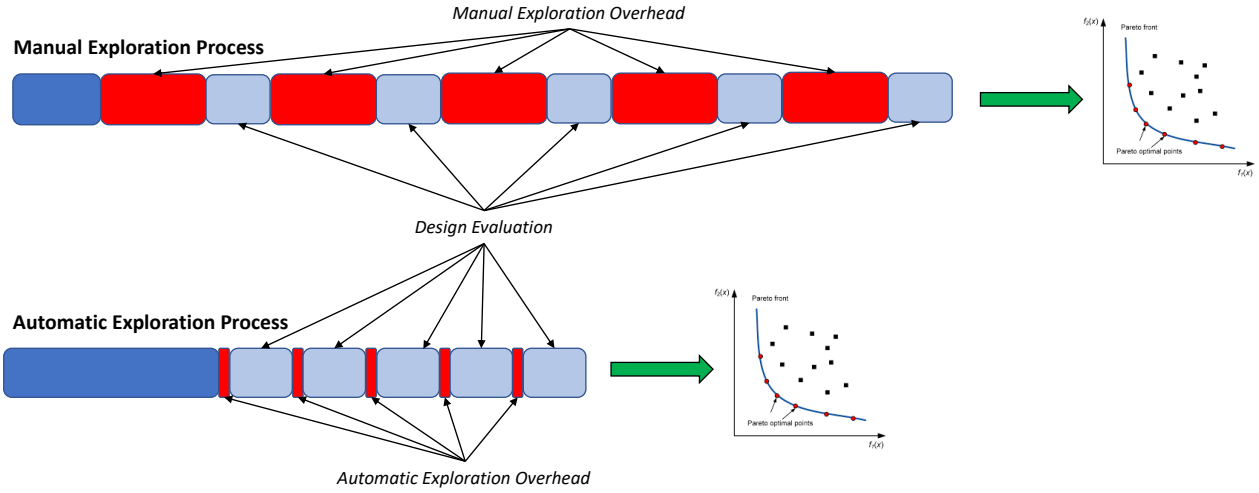


Figure 4.3 Comparison between a manual design space exploration approach and an automatic approach in terms of search engine setup and overhead. Red boxes stand for the time required to analyze the freshly evaluated solution and pick a new one for the next iteration. Blue boxes stand for the time needed to evaluate new solutions.

As highlighted earlier, both manual and automatic DSE start by defining the system model, implementation constraints to differentiate feasible and unfeasible solutions and finally optimization metrics. In the context of manual DSE, the exploration is performed with a human-in-the-loop. In this type of DSE, the designer is a principal player that dramatically influences the outcome of the search engine in such a way that it is difficult if not impossible to inspect as many solutions as in the automated DSE. In every iteration, the designer (a.k.a. decision maker), based on its knowledge, subjective assumptions and previous experience, will tune manually the implementation modifying a limited number of parameters per evaluation. This new solution is evaluated and, based on the simulation results, the same actions will be made. Thus, an enormous amount of time is wasted on rectifying model parameters after analyzing obtained results. In addition, decision maker needs a time lapse between the end of the evaluation process and taking control of its output to be analyzed. The quality of the resulting solutions obtained by manual DSE is highly correlated to the decision maker acquaintance, skills and past experience to efficiently and rapidly assess the results and to move towards the next iteration of the search engine.

An automated DSE engine utilizes a set of predefined optimization objectives combined with effective meta-heuristics to conduct the exploration. The main idea behind this automatic exploration is to conceive a configuration template which can be automatically handled to produce any instance of the target design space. In practice, this means that the initialization

phase of the automated DSE would last longer than, its opponent, the manual DSE. Given the configurable model template, the automated search engine will extensively modify all or most of the model parameters at each iteration before passing through the evaluation stage. This automatic generation of the next solution to be evaluated is widely faster than the manual approach by systematically increasing the overhead by dint of human intervention. The resulting implementations of the automated DSE flow is a set of sub-optimal configurations which are, probably, close to the real Pareto set. In Figure 4.4, we present an overview of the proposed automatic DSE Flow starting from designer's specifications to the final prototype and going through the evolutionary algorithm automating the search engine.

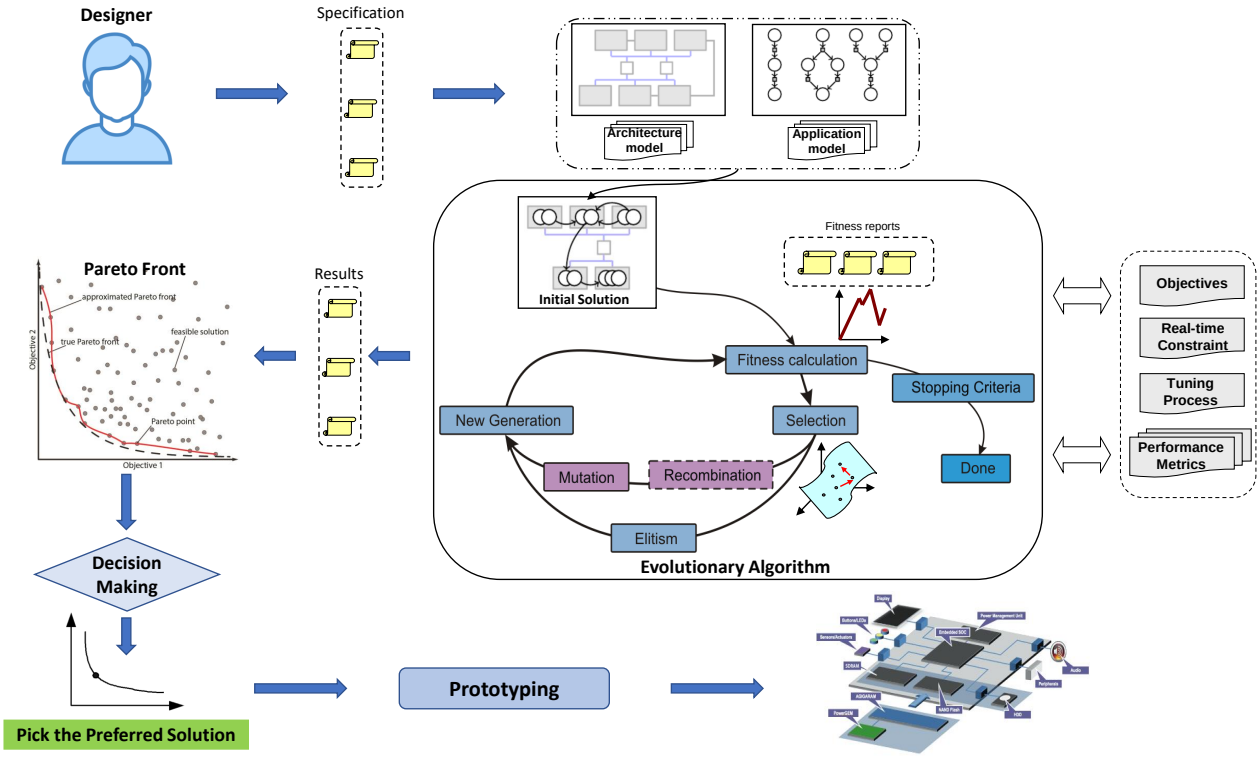


Figure 4.4 The autuomatic design space exploration work flow. The process starts with the specification of hardware, sofware and implementation models by the designer. The output of the multi-objective evolutionary algorithm is the Pareto front.

4.2.2.2 Multi-objective Partitioning

Multi-Objective Optimization Algorithms (MOOA) tend to be harder to deal with as opposed to their mono-objective counterparts since a final decision criteria is difficult to identify. Regardless, MOOAs are advantageous in their ability to find a trade-off between a set of

potentially conflicting metrics, especially with complex systems. We propose an approach that relies on an MOOA to allocate n jobs onto m heterogeneous PEs while optimizing a set of performance metrics and making sure the timing requirements are tended to as well. The proposed algorithm is based on Non-dominated Sorting Genetic Algorithm (NSGAii) (Deb and Agrawal, 2000).

4.2.2.2.1 Performance Metrics When working with complex systems such as FMSs, which involve a large set of parameters, choosing the appropriate performance metrics becomes crucial to obtain an efficient mapping solution. On this basis, given that every module in the system has its own real-time constraints and communication requirements, we define a set of performance metrics. These metrics represent the optimization objectives of the exploration process.

1. Slack Time
   Slack time is defined as the time that remains to a deadline of a workload after it has finished executing. Figure.4.5 represents a case where a workload of 10 tasks is mapped onto a platform with three PEs.

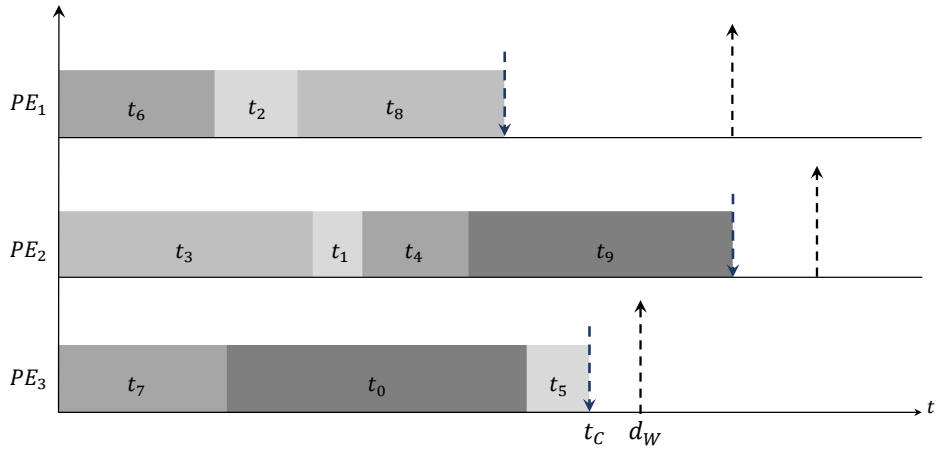


Figure 4.5 Total slack time within three (3) Processing Elements and ten (10) real-time tasks

The objective is to maximize the slack time by the fitness function f_1 in equation.4.1.

$$f_1 = maximize\{ \min_{i \in PE}(d_{Wi} - t_{Ci}) \} \tag{4.1}$$

Integrating the system while taking this property into consideration will be a safety measure for critical scenarios when the load is heavy and the workload takes longer to execute than usual.

2. Communication Cost
   In order to reduce the latency that the network traffic might increase, we add a fitness function, f_2, minimizing the traffic across the network. To do this, we consider the amount of data exchanged between workloads running in different PEs, referred to as inter-communication cost.

$$f_2 = minimize\{\sum_{i\in k}\sum_{j\in p} w_{ij}\} \mid k, p \in PE,\ k \cap p = \emptyset \tag{4.2}$$

3. Memory Consumption
   Our aim is to have a balanced memory load over the PEs. Thus, we define a fitness function f_3 to minimize the memory consumption of modules dispatched to the different PEs. m_{ij} refers to the memory that task t_j requires to execute on PE_i.

$$f_3 = minimize\{\max_{i\ inPE}(\sum_{j=1}^{m} m_{ij})\} \tag{4.3}$$

In the bio-inspired GA, a given population represents a set of solutions to the problem and new generations are created through genetic operators. According to the evolution theory, only the strongest individuals of the population are likely to survive and generate offspring, transmitting their biological inheritance to the next generation. NSGA-II varies from GA not only in the fact that it addresses multi-objective optimization problems but also in the selection operation. It also gives the non-dominated solutions belonging or near the Pareto front in one single run. Before selecting a number of individuals to apply the genetic operators on, the population is ranked on the basis of the non-dominance and crowding distance concepts. More details about the proposed implementation of NSGA-II is given in Chapter 6.

4.2.2.3 Decision Maker Support

In multi-objective optimization there is no unique optimal solution but rather a set of efficient solutions, also known as Pareto solutions. The set of all the Pareto solutions constitutes the True Pareto Front (TPF). Pareto-based exploration approaches, such as evolutionary algorithms, aims at finding the solutions approximating the TPF. However, the large size of the TPF constitutes the major shortcoming of such techniques : the main challenge for designers facing such a large TPF is how to effectively select their preferred solution.

From the designers' perspective, evaluating all the near-optimal solutions (i.e., non-dominated solutions) is quite unrealistic. On the other hand, selecting a preferred solution from a large

TPF is potentially unfeasible. Therefore, a possible solution is to provide a refined representation of the Pareto Front optimal solutions (i.e., a subset of solutions belonging to the TPF) to which we refer as the Reduced Pareto Front (RPF). Selecting the most significant and representative solutions from a TPF, the novel contribution of this work is in developing HypAp, a hypervolume-based automated approach generating a RPF that is much smaller in size compared to the TPF, and yet maintains the main characteristics of the TPF. HypAp helps refine the design of embedded systems through guiding the system designers effectively choose their preferred solutions, now from a RPF includes a few solutions, obtained after applying a multi-objective optimization.

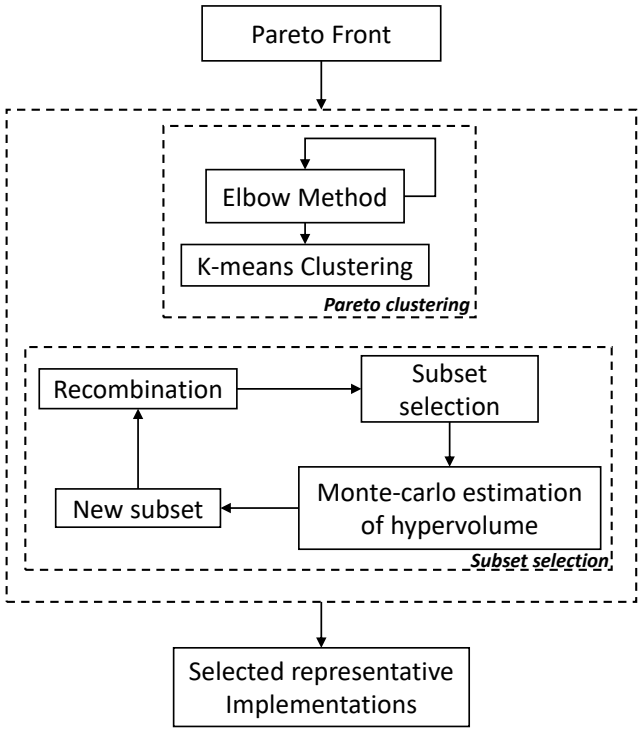


Figure 4.6 The work flow of HypAp. The proposed approach relies on two stages. (1) Pareto clustering and (2) subset selection via hypervolume maximization.

As depicted in Figure 4.6 HypAp is a two-stage approach : Pareto optimal solutions will be first clustered and then a subset of solutions that maximizes the hypervolume will be selected by employing a genetic algorithm. Please note that different performance metrics that need to be optimized during the DSE have a large size or great variability due to their different distributions and orders of magnitudes. Consequently, this kind of variability can lead to a knock-on effect on the clustering result. Therefore, prior to clustering, we normalize the fitness values of the performance metrics by adjusting them (on different scales) to notionally averages (a.k.a. feature scaling). The resulting fitness values lead to a better sym-

metry, and hence more accurate learning in the clustering. More details about the proposed Hypervolume-based approach is given in Chapter 7.

4.2.2.4 Schedulability Guided Exploration Engine

Evolutionary algorithms have already proven to be one of the most powerful and widely used stochastic tools for the implementation of partitioned scheduling. Standard or conventional genetic algorithms that mimic the process of natural selection were initially defined as a general evolutionary algorithm based on blind genetic operators (selection, crossover, mutation). While they are known by their ability to explore the solution space of small problems, this benighted exploration reveals a considerable weakness with bigger problems presenting sizable solution spaces. It is commonly admitted that the use of these operators is quite poor for an efficient exploration. Likewise, since exhaustive exploration of the solution space is unrealistic, a potent option is often to guide the exploration process by hints, derived by problem structure. This guided exploration prioritizes fitter solutions to be part of next generations and avoids exploring unpromising configurations by transmitting a set of predefined criteria from parents to children. Consequently, genetic operators, such as crossover, must incorporate specific domain knowledge to intelligently guide the exploration of the solution space. In order to find better solutions rapidly, exploration is often aided by a guided strategy.

The endeavor of this work is to propose an Improved Genetic Algorithm (ImGA) to guide the exploration process and extensively ameliorate the conventional GA findings. We find an overview of ImGA in Figure4.7. Thus, we are proposing a climbing hill repairing strategy for the population initialization. This strategy aims on adjusting the randomly generated solutions by incorporating a local search algorithm. This climbing hill technique iteratively apply local changes on the generated chromosomes to find a fitter one in its neighborhood. Later, crossover operator is considered as the fundamental search operator of evolutionary algorithms. This operation occurs during evolution according to a user predefined probability. The aim of this operator is to generate a new better chromosome than both of the parents by taking the best characteristics from each one of them. By adopting classical crossover operator, the new solutions are generated by simply swapping random sections of the two parents. However, this random choice is unable to guarantee the best choice. While standard crossover operators do not involve any intelligence during the exploration, in the recent past, a lot of effort has been put into the development of sophisticated crossover operators to guide the exploration in different problems. Thus, we integrated a schedulability-guided crossover operator to be part of our improved algorithm. Then, maintaining a vast population diversity is crucial to ensure that the design space was appropriately explored and avoid

premature convergence by stagnating on sub-optimal solutions. To face this problem, we propose two main contributions ; (1) a circular mutation operator switching the workload of the heaviest and lightest processors. (2), a restart technique based on injecting randomly generated solutions at advanced stages of the optimization engine giving a new impetus to the remaining generations.

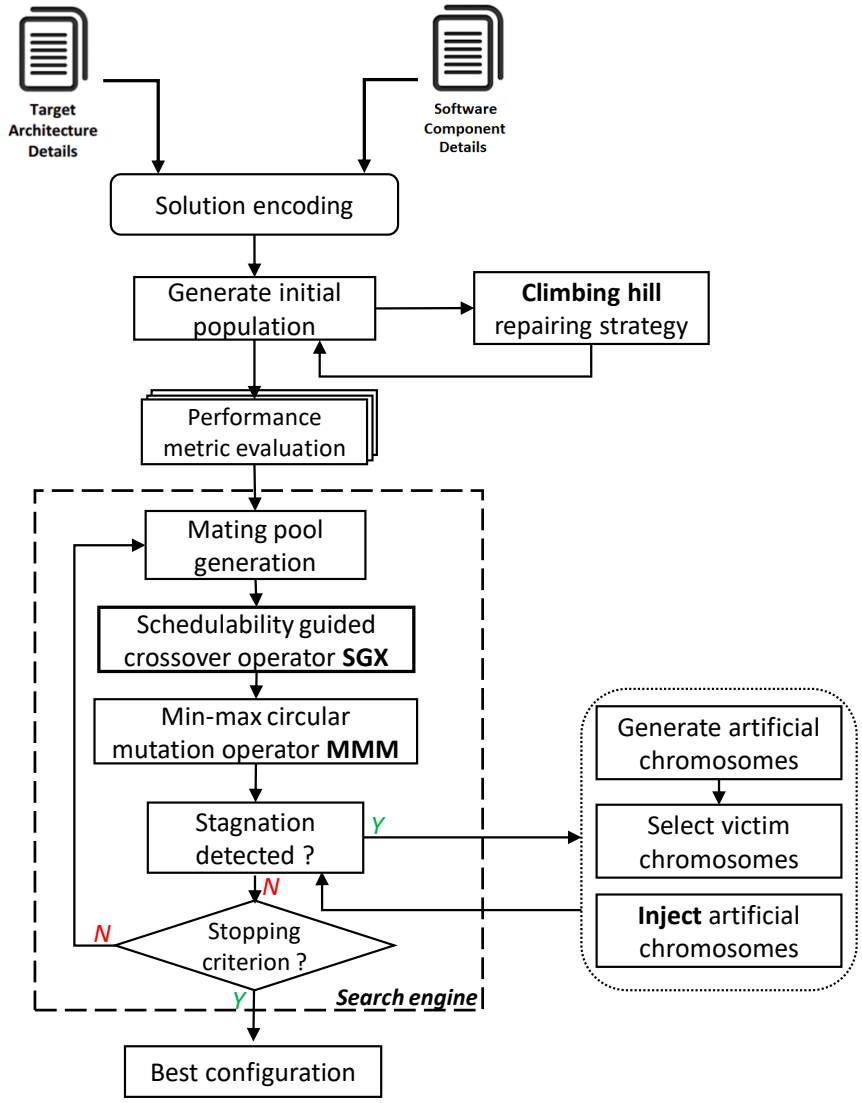


Figure 4.7 Improved Genetic Algorithm work flow. This hybrid version is based on a climbing hill repairing strategy for population initialization, schedulability guided crossover, min-max circular mutation operator and injection policy for better diversity.

Experimental results show that the guided search approach based on the proposed improvements leads to better findings compared to the conventional evolutionary algorithm. To the best of our knowledge, there is, as yet, no previous work in the literature tackling the problem

of guiding the solution space exploration process using a sophisticated evolutionary operator in the context of mapping real-time applications on heterogeneous multi-core systems. More details about the proposed improved genetic algorithm is given in Chapter 8.

4.3 Conclusion

In this chapter, we studied the main research axes of this dissertation and the methodologies put in place in order to tackle the thesis research questions. In the four next chapters, we will present four journal paper adressing in depth these questions. We start by introducing, in Chapter 5, the first paper entitled DeDaSC : Delaunay Triangulation-based Data Stream Clustering Algorithm for the Internet of Things. Then, in Chapter 6, we present the second paper entitled Multi-objective Mapping of Full-mission Simulators on Heterogeneous Distributed Multi-processor Systems in which we study the case of flight simulators. Later, HypAp : a Hypervolume-Based Approach for Refining the Design of Embedded Systems will be the subject of the Chapter 7. At last, Chapter 8 will be dedicated to the fourth paper entitled ImGA : An Improved Genetic Algorithm for Partitioned Scheduling on Heterogeneous Multi-core Systems.

CHAPTER 5 ARTICLE 1 : DEDASC : DELAUNAY TRIANGULATION-BASED DATA STREAM CLUSTERING ALGORITHM FOR THE INTERNET OF THINGS

Rabeh Ayari, Imane Hafnaoui, Sébastien le-Beux, Giovanni Beltrame, Gabriela Nicolescu
Submitted to Transactions on Emerging Topics in Computing

Abstract : The Internet of Things (IoT) is a worldwide network of interconnected objects. It is built on people, physical devices, vehicles, smart homes, data-gathering sensors, actuators, etc. Heterogeneous communication protocols enable these objects to collect, exchange data and update the pre-configured functioning. The huge size of streaming data generated by IoT devices represents its priceless output. In order to make these devices smarter and more adaptive, extracting valuable knowledge from the non-stationary and unbounded streaming data is one of the principal challenges for IoT community. Emerging data mining techniques, such as data stream clustering, represent an effervescent research issue and serve as powerful tools to extract this hidden information in data stream patterns. In information retrieval, the intrinsic nature of data stream clustering requires computationally and memory effective incremental processing of incoming data. In this paper, we introduce a new Delaunay triangulation-based Data Stream Clustering algorithm (DeDaSC) able to construct arbitrary shaped clusters on the fly without any prior knowledge of the number of clusters and their shapes. DeDaSC requires only the newly arrived data, not the entire dataset, to be saved in memory. An incremental Delaunay triangulation is introduced to guarantee the effectiveness of the neighborhood discovery with limited computation time and memory space. Our performance studies over a number of synthetic datasets demonstrate the efficiency of our algorithm to create arbitrary shaped clusters as they evolve in a fully online manner. Experimental results on these datasets show that DeDaSC can discover clusters that many existing state-of-the art clustering algorithms fail to find.

5.1 Introduction

In recent years, the Internet of Things (IoT), a.k.a. the Internet of Objects, attracted the attention of researchers from academia and industry throughout the world. Introducing IoT-based products and services has been a growing trend thanks to rapid advances of numerous technologies, including ubiquitous sensors, actuators, biomedical systems, and real-time embedded systems. Due to the exponential growth of IoT networks, in the next few years, more

devices will join that list. According to Gartner, Inc. (Fenn and Raskino, 2011), more than 20 *billion* connected objects will be in use worldwide by 2020. The aim of such tremendous heterogeneous network is that all these objects will be controlled, monitored, detected by other objects, communicate together, and can even make autonomous decisions.

This huge expansion in terms of connected devices will raise the scale of produced data to an unprecedented level. These massive amounts of data are generated unceasingly and in a highly fluctuating rate. Cisco reported that IoT networks will generate more than 400 *zettabytes* (*trillion gigabytes*) of data a year by 2018 (Index, 2015). Therefore, it is essential to consider how efficiently to manage and analyze these data to transform the information into knowledge for smarter decisions in the future. The mining task enables IoT manufacturers to aggressively leverage these data in key business decisions and processes, with impressive results.

Such process requires advanced and computationally effective cognitive computing techniques. Fortunately, data mining techniques avail of these voluminous data to improve their learning capabilities to discover hidden patterns. In practice, the generated data by IoT objects arrives continuously and needs to be processed on the fly. This kind of dataset is appropriately referred to as data streams. For ease of understanding, data stream can be seen as a river : data come in and come out as reported in (Suresh et al., 2014). Data points are scanned only once and hidden patterns are constructed progressively. This real-time knowledge extraction process makes the mining task more challenging. Therefore, using conventional clustering algorithms certainly lead to misleading output (Aggarwal, 2007; Muthukrishnan et al., 2005; Babcock et al., 2002; Golab and Özsu, 2003; Amini et al., 2014). Traditional offline methods are appropriate only for resident data stored in large data repositories and consequently cannot address the problem of a continuous supply of data with temporal locality (Aggarwal and Reddy, 2013).

Data mining is a computing process that covers a wide spectrum of data analysis and knowledge retrieval techniques, including classification, prediction, outlier analysis, and clustering (Fayyad et al., 1996). In this context, several data stream clustering algorithms have been proposed to perform unsupervised learning. Data stream clustering is a useful and ubiquitous tool in data analysis that assigns on the fly incoming data into groups whose members present higher degree of similarity than others. Prior clustering research has largely focused on static datasets. The nature of streaming data requires the development of sophisticated algorithms capable of performing fast and incremental processing of data objects, suitably addressing time and memory limitations. Likewise, data stream clustering algorithms should be able to detect whenever new clusters should appear, disappear i.e. being fused. Also, they need

to have the capacity to dynamically distinguish outliers a.k.a. noise from potential clusters (Guha et al., 2003).

Based on the aforementioned considerations, a rich body of fundamental research in clustering has emerged in the data stream model of computation. Most of the proposed algorithms are either hybrid (online/offline) methods, windowed offline methods, find only hyper-elliptical clusters, require prior knowledge of the number of clusters or their density. In real-life applications, such hypothesis is quite unrealistic, clusters are more likely of irregular shapes and the number of clusters keeps changing over time and cannot be fixed in advance. Clustering based on the regular-shaped class struggles when faced with data in which the existing clusters do not conform to the allowed shape of clusters.

In this paper, we are introducing a fully online and efficient data stream clustering algorithm using incremental Delaunay triangulation to address these issues and clusters streams of data on the fly.

The rest of the paper is structured as follows : Section 2 surveys the current state of the art with an exhaustive comparison of the related work. Section 3 lists a bench of definitions useful for DeDaSC comprehension. Section 4 describes the principles and methodology behind the DeDaSC algorithm and provides a description of its pseudo-code. Section 5 describes the datasets and the methodology of their use throughout the analysis parts of the paper and provides analysis of the performance of the proposed algorithm and comparisons to alternative techniques. Finally, we conclude the paper and consider some directions for future work in Section 6.

5.2 Related Work

Data stream clustering is motivated by emerging applications involving massive datasets (Guha et al., 2003). An exhaustive survey of data stream clustering algorithms and relevant applications is given in (Silva et al., 2013). It includes, for instance, bearing prognostics, forest cover, grid computing, sensor networks, network intrusion detection, stock market analysis, etc. When designing data stream clustering algorithms, several requirements have to be considered and various issues need to be addressed, mainly, in terms of run-time and memory needs. In Table 5.1, we make a list of the most significant restrictions to be considered with this class of data. Due to the wide spectrum of application domains and model constraints, the research in data stream clustering has gained high attraction. Below we review some of the most important research work in the field of data stream clustering.

Broadly speaking, we can classify the related work on data stream clustering into the fol-

Table 5.1 List of Data Stream Clustering Restrictions with their Definitions

Restriction	Description
Stream order	Data stream should be processed on the fly without any order constraint
Single pass	Data points are linearly scanned only once before being discarded and random access is prohibited.
Time constraint	Designing a clustering algorithm of low computational complexity without sacrificing performance.
Memory constraint	Reducing the space complexity of the clustering algorithm and storing only a summary of incoming data.
Prerequisites	No assumption or prior knowledge of the number of clusters to be created.
Cluster shape	Position and shape of micro/macro-clusters evolve oveR time. Data are grouped on arbitrary shaped clusters.
Outliers	Ability to handle outliers. In data stream random noise appears occasionally.
Scalability	Algorithm's capacity to handle a growing amount of incoming data.

lowing four clustering techniques, namely density-based algorithms, partitioning algorithms, hierarchical algorithms, and grid-based algorithms. A comparison of the surveyed clustering algorithms is summarized in Table 5.2. This comparison is based on a list of metrics discussed in details in Section 5.3.

5.2.1 Density-based Algorithms

Density-based algorithms aim mainly on finding arbitrary shaped clusters and noise filtering based on density. The base idea behind this technique is to keep increasing cluster cardinality i.e. number of members as long as its local density reaches a predefined certain threshold. Density reachability and density connectivity concepts were introduced to refine the algorithm's process. In the context of density-based algorithms, clusters are dense regions (i.e. maximal set of density connected data points), separated by regions of lower density. A series of density-based algorithms have been developed in literature. Hereafter, we give a brief overview of these algorithms ; DBSCAN (Density Based Spatial Clustering of Applications with Noise) has been proposed by Martin Ester, Hanz-Peter Kriegel's group in (Ester et al., 1996). DBSCAN is considered as the prototypical density-based clustering approach and the most used one. DBSCAN aims at estimating the density surrounding each data point by

Table 5.2 A Comparative Study of Data Stream Clustering Algorithms

	Clustering technique	Single pass requirement	Shape of the clusters	A-priori knowledge of the number of clusters	Outlier's handling	Variable density	Algorithm's time complexity
DBSCAN	Density-based	✗	Arbitrary	✗	✓	✗	$\mathcal{O}(n \log n)$
Incremental DBSCAN	Density-based	✓	Arbitrary	✗	✓	✗	$\mathcal{O}(n) + \mathcal{O}(n \log n)$
OPTICS	Density-based	✗	Arbitrary	✗	✓	✗	$\mathcal{O}(n \log n)$
DENCLUE	Density-based	✗	Arbitrary	✗	✓	✗	$\mathcal{O}(n \log n)$
Sequential K-means	Partitioning	✓	Hyper-Spherical	Required	✗	✗	$\mathcal{O}(n^{dk+1})$
CLARANS	Partitioning	✗	Hyper-Spherical	Required	✓	✗	$\mathcal{O}(n^2)$
STING	Grid-based	✓†	Arbitrary	✗	✓†	✓	$\mathcal{O}(c)$
Wave Cluster	Grid-based	✗	Arbitrary	No ‡	✓	✓	$\mathcal{O}(c)$
CURE	Hierarchical	✓	Non-spherical	✗	✓	✗	$\mathcal{O}(n^2 \log n)$
CHAMELEON	Hierarchical	✗	Arbitrary	Required	✗	✓	$\mathcal{O}(nk + n \log n + k^2 \log k)$
BIRCH	Hierarchical	✗	Hyper-Spherical	Required	✓	✓	$\mathcal{O}(n)$
DeDaSC (Our algorithm)	Density-based	✓	Arbitrary	✗	✓	✓	$d \leq 3 \Rightarrow \mathcal{O}(n) + \mathcal{O}(n^{1/2})$ $d \geq 4 \Rightarrow \mathcal{O}(n) + \mathcal{O}(n)$

† Partial solution.
‡ Useful if specified.
^{n} Number of data samples.
^{d} Number of data dimensions.
^{k} Number of clusters.
^{c} Number of grid cells.

counting the number of members in a predefined *eps* neighborhood compared to a certain threshold to identify core region and outliers. Later, core samples join clusters if they are density-reachable and border points are assigned to clusters. In (Ester et al., 1998), authors proposed an incremental version of DBSCAN allowing gradual modifications of the dataset. After each iteration micro-cluster connections are created and broken according to the changes. The main disadvantage of this incremental version is that the whole dataset is required to be available for each iteration or update. Another well-known algorithm is OPTICS (Ankerst et al., 1999) (Ordering Points To Identify the Clustering Structure). The base idea behind OPTICS is similar to DBSCAN while tackling one of the most important limitations of DBSCAN consisting of the way meaningful clusters with varying density are detected. In (Hinneburg et al., 1998), authors proposed DENCLUE (DENsity-based CLUstEring) which is one of the most effective unsupervised clustering algorithms allowing the classification of voluminous data. DENCLUE is based on the concept of density closeness and the hill-climbing algorithm.

5.2.2 Partitioning Algorithms

Partitioning algorithms (a.k.a. iterative relocation algorithms) categorize the data samples into a pre-defined fixed number k of clusters. These algorithms aim at optimizing a given criterion by iteratively relocating data points by moving them from one cluster to another. The clustering is based on the following conditions : (1) each cluster contains at least one

data point and (2) each data point belongs to only one cluster. To improve the partitioning, it uses an iterative relocation procedure which moves objects from one group to another. A common measure of partitioning quality is the degree of similarity (e.g. Squared Euclidean distance, Mahalanobis distance, Minkowski distance) between data samples belonging to the same cluster, as data samples of different clusters. The main drawback of such simple technique is that it is unsuitable for discovering complex arbitrary shaped clusters of voluminous datasets. An incremental version of k-means was proposed in (Macqueen, 1967). This sequential version was adapted for streaming data and generated hyper-spherical clusters. This algorithm adds cluster centroids gradually as clusters are being formed. In (Ng and Han, 1994), authors proposed CLARANS which is an efficient medoid-based clustering algorithms based on randomized search. CLARANS draw samples of neighbours dynamically. The clustering process searches on the graph where every node is a potential solution, that is, a set of k-medoids. Generally, partitioning algorithms are time consuming and cannot be effectively applied to the voluminous datasets.

5.2.3 Hierarchical Algorithms

Hierarchical algorithms are based on constructing a hierarchy of clusters (a.k.a. dendrogram) which iteratively divides the dataset into smaller subsets until each subset contains single data sample. This technique involves creating clusters that have a predetermined ordering from top to bottom. In this hierarchy, each node of the tree represents a group of similar data. The cluster hierarchy can be formed from the leaves to the root in an agglomerative approach or the opposite in divisive approach (Murtagh, 1983). The merge or split process should be stopped whenever the pre-defined stopping criterion is met. Among these algorithms we cite CURE (Clustering Using Representatives) (Guha et al., 1998) which is designed to handle large datasets, that is more robust to outliers and identify clusters having non-spherical shapes. To do this, CURE employs a mixture of random sampling and partitioning. In (Karypis et al., 1999) authors proposed CHAMELEON that measures the degree of similarity between two clusters is based on a dynamic model. To group data samples, CHAMELEON aims on maximizing the intra-cluster similarity and minimizes the inter-cluster one. CHAMELEON is applicable to all types of data as long as a similarity matrix can be constructed and requires the prior knowledge of the number of clusters to be created. In (Zhang et al., 1996) authors proposed BIRCH which is an unsupervised data mining algorithm that has been applied for data stream mining. BIRCH introduced the use of micro-clustering and macro-clustering. It scans the database to build an in-memory tree before applying clustering algorithms to cluster the leaf nodes. A major limitation of hierarchical algorithms is that as soon as two data points are grouped together, they cannot move to other groups in the

tree. Thus, migration is prohibited between the clusters.

5.2.4 Grid-based Algorithms

Grid-based algorithms explore multi-resolution grid data structure in clustering. They subdivide the data space into a limited number of cells to form a grid structure. This partitioning of the data space is based on the prior knowledge of data granularity. It allows the detection of dense regions from the cells in the grid. The main advantage of this method is its scalability given that grid-based algorithms are typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space. Wang et al. proposed a STING (STatistical INformation Grid-based method) for spatial data mining (Wang et al., 1997) in the grid structure. The quality of the resulting clustering depends on the granularity of the lowest level of the grid structure. Finer granularity of the grid cells leads to higher quality of the obtained clustering and the cost of processing will increase substantially. Another well-known grid-based clustering approach is WaveCluster (Sheikholeslami et al., 1998). The WaveCluster method is not sensitive to the order of the number of input data samples to be processed. WaveCluster uses a wavelet transformation to transform the original feature space. WaveCluster is well capable of finding arbitrary shape clusters.

5.3 Basic definitions of DeDaSC

Before detailing the DeDaSC process, we start by introducing a set of definitions. A summary of the used notations is given in Table 5.3.

Table 5.3 List of DeDaSC Notations with their Definitions

Symbole	Description
p	Data point received at instant t
n	Number of micro-clusters at instant t
η_i	Micro-cluster *id*
μ_i	Maturity index of micro-cluster η_i
μ_{th}	Maturity threshold of micro-cluster η_i
$C_i(t)$	Centroid co-ordinates of micro-cluster η_i at instant t
ϕ	Magnetic attraction distance
ζ_j	Macro-cluster *id*
$DT(n,t)$	Delaunay triangulation of micro-cluster centroids at instant t

Definition 5.3.1 *Micro-cluster*

A Micro-cluster at time t is defined as η_i where $1 \leq i \leq n$ for a set of data points residents within a user defined neighborhood area. n represents the number of micro-clusters at time t. Associated with each micro-cluster, we create a unique id whenever it is first created. The neighborhood is defined by the Euclidean distance between the centre of the freshly updated micro-cluster and the data points in that micro-cluster.

Definition 5.3.2 *Maturity Index*

A micro-cluster's maturity index μ_i represents the number of data points belonging to the micro-cluster η_i. For each micro-cluster at time t, we consider a maturity index that keeps increasing over time. The initial value for each micro-cluster is 1. This index portrays the well-known density metric used in density-based algorithms.

Definition 5.3.3 *Maturity Threshold*

The maturity threshold μ_{th} defines the upper bound of micro-cluster's maturity index to pass from an outlier to a potential micro-cluster that can be considered in the final phase of macro-clustering.

Definition 5.3.4 *Micro-cluster Centroid*

The most representative point within each micro-cluster η_i is the centroid $C_i(t)$ at time t which is used as a measure of cluster location. This is the arithmetic mean ("average") position of all the data points in the micro-cluster. The similarity of two micro-clusters is defined as the similarity of their centroids. The centroid keeps changing its position every time a new data point p joins the micro-cluster η_i.

$$C_i(t) = \frac{\mu_i \times C_i(t-1) + p}{\mu_i + 1} \tag{5.1}$$

Definition 5.3.5 *Magnetic Attraction Distance*

The degree of similarity between streaming data points is defined with the use of a distance measure that we denoted magnetic attraction distance ϕ. This parameter is used to identify dense neighborhoods and represents the maximal distance from any point in the micro-cluster to the centroid and decide whether the incoming data sample falls in the micro-cluster region or not.

Definition 5.3.6 *Outlier Micro-cluster*

An outlier micro-cluster at time t for a group of close points is a micro-cluster that hasn't reached yet its maturity threshold (a.k.a. sparse region or empty region). Outlier micro-clusters cannot be considered in the macro-clustering process.

$$\forall i \in [1..n],\ \textit{if } \mu_i < \mu_{th} \rightarrow \eta_i \textit{ is an outlier micro-cluster} \tag{5.2}$$

Definition 5.3.7 *Mature Micro-cluster*

A mature micro-cluster at time t for a group of close points is a micro-cluster with a number of data points (i.e. members) greater or equal to the maturity threshold (a.k.a. dense region). Only mature micro-clusters are considered in the macro-clustering process.

$$\forall i \in [1..n],\ \textit{if } \mu_i \geq \mu_{th} \rightarrow \eta_i \textit{ is a mature micro-cluster} \tag{5.3}$$

Definition 5.3.8 *Macro-cluster*

Each mature micro-cluster η_i joins a given macro-cluster ζ_j where $j \in [1..m]$ and m represents the number of the macro-clusters at time t. The macro-clustering phase discussed at a later stage will use this micro-clusters membership in order to create the final arbitrary shaped clusters.

$$\forall i \in [1..n]\ \textit{if } \mu_i \geq \mu_{th}\ \exists j \in [1..m]\ /\ \eta_i \subset \zeta_j \tag{5.4}$$

Definition 5.3.9 *Delaunay Triangulation*

A Delaunay triangulation for a set n of points in a plane $P \in \Re^2$ at time t is a triangulation $DT(n,t)$ such that no point is inside the circumcircle of any triangle in $DT(n,t)$. By considering circumscribed spheres, the notion of Delaunay triangulation extends to three and higher dimensions. The triangulation vertices of $DT(n,t)$ represent the micro-cluster centroids.

5.4 Proposed Algorithm : DeDaSC

While considering the aforementioned restrictions and in comparison with the methods discussed above, we propose DeDaSC, a novel data stream clustering algorithm. Our clustering algorithm addresses all these considerations and can be divided into three parts that are detailed in below : (1) neighborhood discovering using incremental Delaunay triangulation (2) online micro-clustering, and (3) macro-cluster construction.

5.4.1 Neighborhood Discovering Using Incremental Delaunay Triangulation

One of the main issues faced on data stream clustering is the ability to, rapidly and effectively, discover the neighboring micro-clusters of the freshly received data point. One trivial and widely adopted technique consists of computing the Euclidean distance to all the micro-cluster centroids to find the nearest neighbour (i.e. micro-cluster). The decision of joining this micro-cluster or not will be taken based on the difference between the aforementioned distance and the magnetic attraction distance. In this context, our aim is to propose a fully online spatial proximity method with reduced timing cost (i.e. time complexity). After this operation and based on its output, the algorithm groups data points with similar attributes together to be part of the same cluster. In this paper, we utilize the Delaunay triangulation for neighborhood discovering of a given data point p.

In trigonometry, a triangulation of a discrete set of points $P \in \Re^d$, subdivides data points into triangles, and by extension the subdivision of a higher-dimension geometric object into simplices (i.e. generalization of the notion of a triangle or tetrahedron to arbitrary dimensions) (Jick, 1979). This operation allows the determination of the location of a point by connecting it with triangles to known points. Frequently used and studied point set triangulation include the Delaunay triangulation.

A Delaunay triangulation is a particular way of joining a set of points to make a triangular mesh (De Loera et al., 2010; Guibas et al., 1992; De Berg et al., 2008). Delaunay triangulation tends to avoid skinny triangles (*i.e.* a triangle whose height is much greater than its base), as depicted in Figure 5.2. Delaunay triangulation is a fundamental geometric construction that has been applied in numerous applications in different domains. A Delaunay triangulation of a given dataset in a d-dimensional Euclidean space is a triangulation such that no point is inside the circum-hypersphere of any simplex in the triangulation. A circle circumscribing any Delaunay triangle does not contain any other input points in its interior. The problem of finding the Delaunay triangulation is equivalent to the problem of finding the convex hull of the same dataset in $(d+1)$-dimensional space. Given that the convex hull is unique, so is the triangulation, assuming all facets of the convex hull are simplices.

The pseudo-code of the incremental Delaunay triangulation is given in Algorithm 1. Figure 5.1 gives the steps performed by incremental Delaunay triangulation when a new data point p joins the dataset.

This algorithm was originally presented in (Green and Sibson, 1978), its pseudo-code in (Guibas and Stolfi, 1985) and the detailed implementation in (Lischinski, 1994). It consists of two parts :

Algorithm 1: Incremental Delaunay triangulation

Result: Updated Delaunay triangulation

```
Locate the triangle T containing the new data point p;
Create edges connecting p to the vertices of the containing triangle T;
Verify the empty circumcircle condition by inspecting the new edges;
while Candidate edges remain uninspected do
    if condition satisfied then
        The old edges remain unchanged;
    else
        Flip the offending edge and replace it by the other diagonal of the surrounding
         quadrilateral;
    end
end
```

1. Locate the triangle T containing the freshly received data point p.
2. Insert the new sample into the current triangulation and link it to the vertices of the containing triangle by creating new edges before making a list of modifications. In Figure 5.1, a summary of the scenario of updating the Delaunay triangulation is given. After locating the data sample, the edges of the containing triangle are inspected. If the circumcircle condition remains satisfied, the edges are unchanged otherwise the other diagonal of the quadrilateral takes place. The process is iteratively repeated until all the triangles satisfy the circumcircle condition and we obtain the updated Delaunay triangulation.

According to (Lischinski, 1994), if the inserted points are uniformly distributed, the expected number of operations to locate a point is $\mathcal{O}(n^{1/2})$. Therefore, finding the nearest neighbour will require only 3 comparisons to be performed with the vertices of the container triangle T which means a total computational complexity of $\mathcal{O}(n^{1/2})$. In practice, the average number of edges to be tested does not exceed 9 operations. Thus, the expected time complexity falls to a constant time per each insertion. Based on these timing costs the overall time complexity of the incremental Delaunay triangulation for neighborhood discovering is $\mathcal{O}(n^{1/2})$.

Without loss of generality and for the sake of clarity, we consider only bi-dimentional spaces for the remainder of the paper. An example of Delaunay triangulation is depicted in Figure 5.2. This figure clearly shows the capacity of this technique in terms of neighborhood discovery by comparing the dense regions to those in between.

The incremental Delaunay triangulation can be seen as a sub-step of the micro-clustering process which is detailed later. It aims on adding new points to the current triangulation and makes the necessary changes to maintain the invariant that the triangulation is Delaunay.

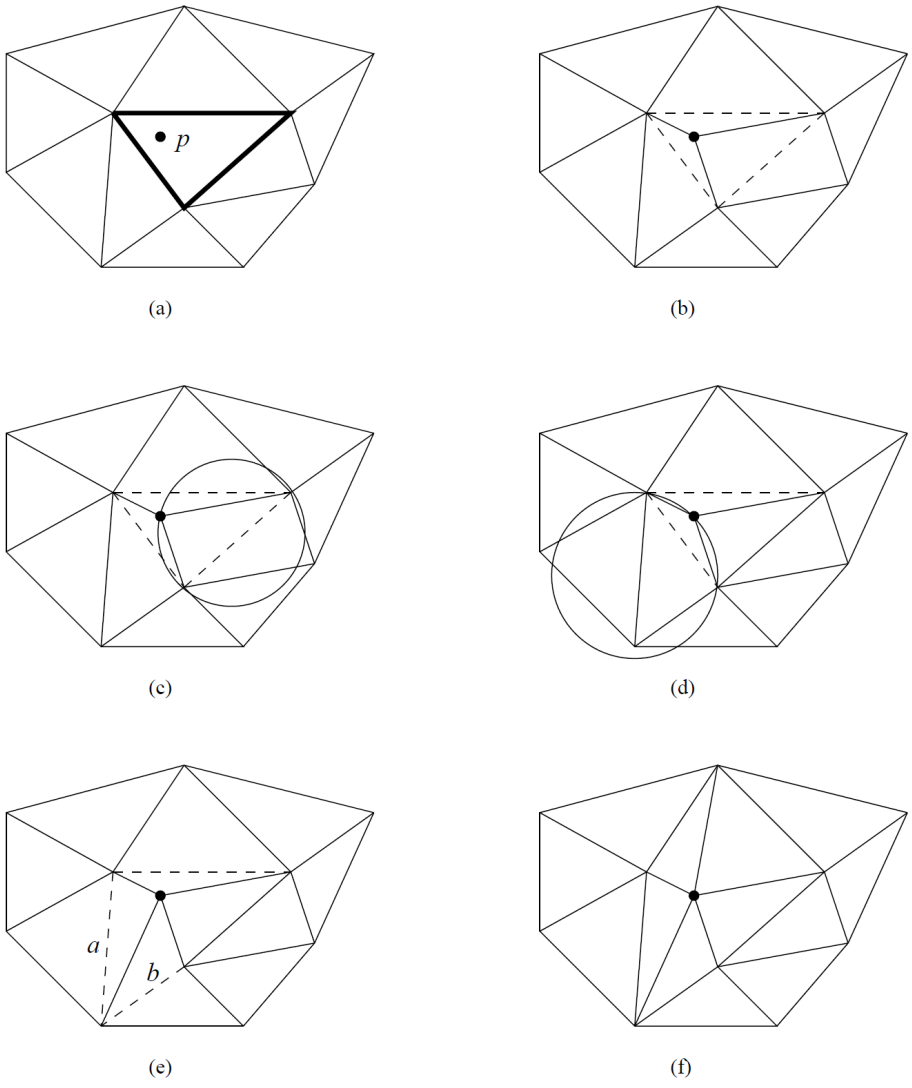


Figure 5.1 Inserting a new point in the triangulation. Dashed lines indicate edged that needs to be inspected by the algorithm (Lischinski, 1994).

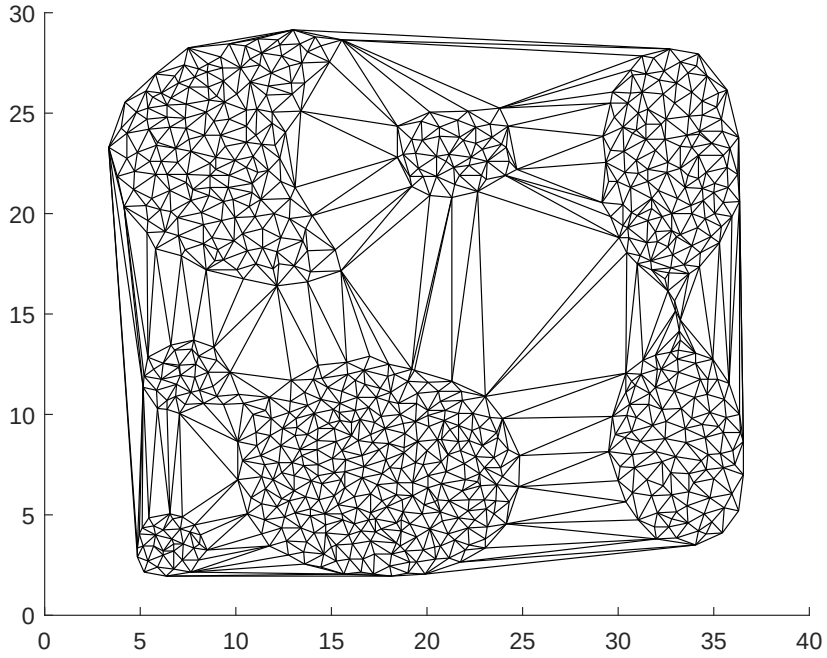


Figure 5.2 Delaunay triangulation of a dataset containing arbitrary shaped clusters. Dense regions or possible clusters involve smaller triangles compared to empty regions containing only outliers.

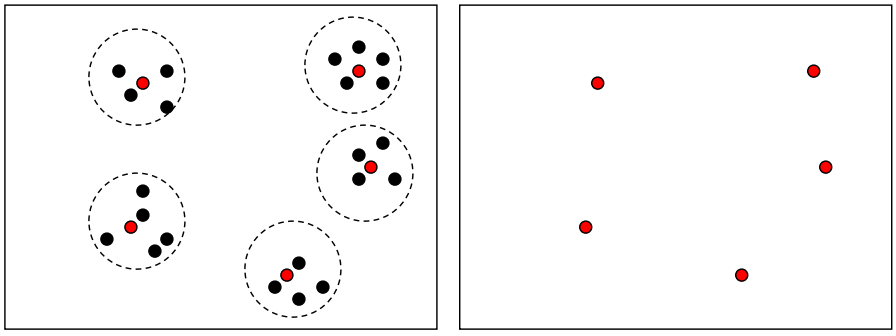
The Delaunay triangulation expresses the degree of proximity between micro-clusters. Each vertex in the Delaunay triangulation represents a micro-cluster centroid. It is performed after updating the micro-cluster data structure either with the new centroid or the recently adjusted one. For the rest of the paper, we consider χ_1 as the function modeling the event that a new micro-cluster has been created and χ_2 as the event that the data p joined an existing micro-cluster.

Keeping in mind, simultaneously, the single pass and timing constraints, an incremental implementation of Delaunay triangulation we propose. This incremental algorithm allows the incoming data point p at time t to update the Delaunay triangulation at time $t-1$ denoted $DT(n_{t-1}, t-1)$ one by one as they come. n_t represents the number of existing micro-clusters at time t. If the new data point p falls in an empty region, a new micro-cluster is created and the incremental Delaunay triangulation is performed by adding p to $DT(n_{t-1}, t-1)$. Otherwise, the new data point p joins an existing micro-cluster, the Delaunay triangulation $DT(n_{t-1}, t-1)$ will be updated after shifting the centroid of the freshly modified micro-cluster. The resulting Delaunay triangulation will be denoted by $DT(n_t, t)$ where :

$$n_t = n_{t-1} + 1 \text{ if } \chi_1 = 1 \text{ and } n_t = n_{t-1} \text{ if } \chi_2 = 1 \tag{5.5}$$

5.4.2 Online Micro-Clustering

Since data streams are continuously generated, DeDaSC efficiently compresses the incoming data in a set of micro-clusters. These micro-clusters are referred to as snapshots of data groups that keep changing and growing over time as new data points join the list. Therefore, DeDaSC provides a compact representation of the clusters compressing all the data points previously processed at time t. This crucial phase requires a very efficient process to save an appropriate summary of streaming data. The pseudo-code of the micro-clustering stage is given in Algorithm 2. The online micro-clustering summarizes the flow of incoming streams of data in reduced data structures containing its metadata (*i.e.* centroid location, maturity index, etc). In Figure 5.3 data points are coloured in black and the micro-cluster centroids in red. As depicted in Figure 5.3(b) only the red points representing the centroids reside in memory.



(a) Snapshot of data points

(b) Snapshot of micro-cluster centroids stored in memory

Figure 5.3 An image of streaming data and the created micro-clusters. Data points are colored in black and the micro-cluster centroids in red. The red points only reside in memory.

In order to achieve this micro-clustering goal, DeDaSC goes through a list of iterations.

5.4.2.1 Initialization

We first need to initially create three micro-clusters at the beginning of the data stream processing. This initialization process will allow the construction of the first triangle of the Delaunay triangulation. For this purpose, the first three points are stored on primary memory as three different micro-clusters regardless their spatial proximity as depicted in Figure 5.4.a. Once these initial three micro-clusters have been created, the online process of micro-clustering is initiated. In the same Figure we show a new data point depicted with a diamond marker falling in the space and launching the execution of the next phase of DeDaSC.

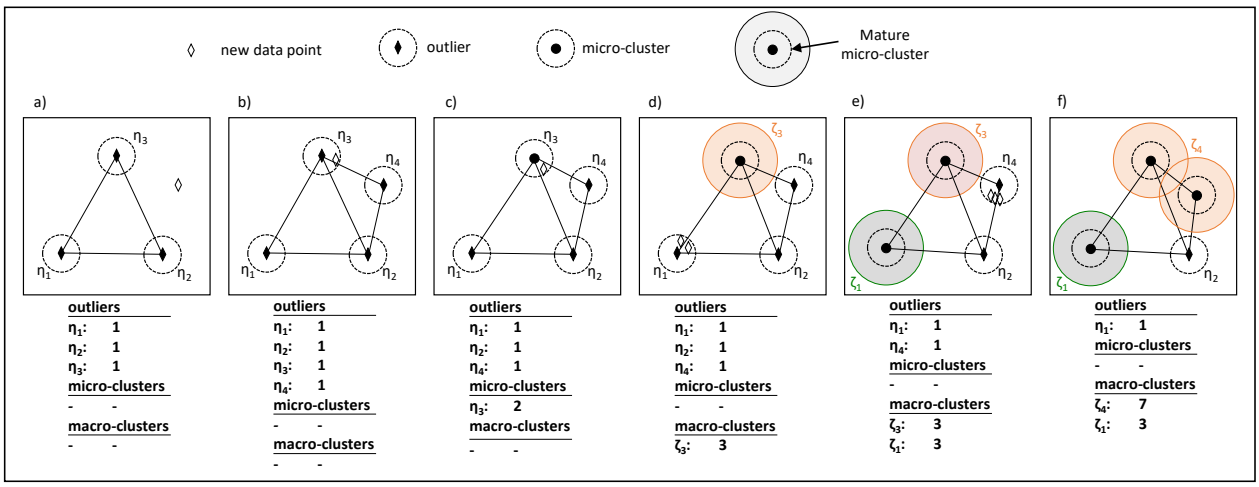


Figure 5.4 DeDaSC Algorithm : an illustrative example. In this example we assume that the maturity threshold is equal to 3.

5.4.2.2 Micro-cluster Assignment

Whenever a new data point is linearly scanned, the micro-clusters are updated in order to reflect the adjustments. Each data point either needs to be put in a new micro-cluster of its own (*i.e.* $\chi_1 = 1$), or needs to be absorbed by a pre-existing micro-cluster (*i.e.* $\chi_2 = 1$). To make such substantial decision with limited timing cost, we first refer to the most recent image of the Delaunay triangulation at time $DT(n_{t-1}, t-1)$. This original version of Delaunay triangulation allows us to easily find the nearest micro-cluster to the incoming data sample p at time t and the distance between them.

— Micro-cluster Birth

In many cases, the data point p may not be sufficiently close to this micro-cluster which means that the distance is greater than the predefined magnetic attraction distance. Figure 5.4.b illustrates an example of this situation where p should be placed in a new outlier micro-cluster of its own. This new micro-cluster contains only the new data point p_i and a new *id* is assigned to this micro-cluster. This *id* identifies the micro-cluster uniquely at any future stage of the data steam process. The micro-cluster centroid is the data sample location itself and the maturity index is initialized to 1. This may happen because of the arrival of an outlier point, or because of sudden changes in the data stream itself. A sudden change may lead to a new trend in the data stream which often exhibits itself in the form of a new micro-cluster. Once the new micro-cluster is created, the insertion process of the incremental Delaunay triangulation is launched with the new centroid.

— Micro-cluster Growth

If the data point p falls in a populated region (i.e. within the magnetic attraction field of the nearest micro-cluster) than the incoming data point p is absorbed by this micro-cluster. The centroid of the absorber (*i.e.* nearest micro-clusters) is updated, the maturity index is incremented. In Figure 5.4.b, we notice that the new data point p falls in the attraction field of the outlier micro-cluster η_3. Since, we are assuming that $\mu_{th} = 3$, η_3 becomes a mature micro-cluster when it absorbs a new data point as depicted in Figure 5.4.c. From that moment, we see in Figure 5.4.d that η_3 will be considered in any macro-clustering process whenever it overlaps with other mature micro-clusters. The same process happens to η_1 and η_4 as shown in Figures 5.4.d and 5.4.e consecutively. After each data point absorption, the Delaunay triangulation is updated by considering the new location of the corresponding micro-cluster centroid.

Algorithm 2: Online micro-clustering pseudo-code

Result: Updated micro-clusters and Delaunay triangulation

```
p = Load the new streaming data point;
t_i = Find triangle containing p_i from the Delaunay triangulation at DT(t − 1);
η_{p_i} = nearest neighbour (micro-cluster centroid);
dist(p_i, η_{p_i}) = Compute Euclidean distance between p_i and η_{p_i};
if dist(p_i, η_{p_i}) < φ then
    Add p_i to this micro-cluster η_{p_i};
    Update micro-cluster's centroid;
    Increment micro-cluster's maturity index;
    Update Delaunay triangulation after shifting the micro-cluster centre;
else
    Create new micro-cluster;
    Define micro-cluster's centroid;
    Initialize micro-cluster's maturity index to 1;
    Apply incremental Delaunay triangulation with the new micro-cluster;
end
```

5.4.3 Macro-Cluster Creation

The micro-clusters created, at this stage, serve as an intermediate statistical representation which can be maintained in an efficient way even for a data stream of large volume. The macro-clustering stage will not be performed for all the data points of the original stream of data. It will utilize the compactly stored micro-clusters (more accurately the mature micro-clusters). Therefore, it is not constrained by one-pass requirements. It is assumed that as input to the algorithm, the user supplies macro-cluster attraction distance to merge overlapped micro-clusters. If chosen appropriately, the magnetic attraction distance can make the tuning

of the macro-cluster attraction distance pseudo-automatic. Macro-clustering is a kind of a re-clustering process introducing a new scale at which any mature micro-cluster that will be within the macro-cluster attraction distance of the freshly updated micro-cluster will revise its membership to be the same.

The macro-clustering is not performed with every streaming data point. It will wait the updated micro-cluster to achieve the maturity threshold and then it can be launched. In Figure 5.4, we can easily notice that the re-clustering has been performed only once. It has been launched only when two mature micro-clusters overlapped as depicted in Figure 5.4.f. Hereafter we detail the process which leads to launching the macro-clustering process. As highlighted before, in Figure 5.4.e three data points join the outlier micro-cluster η_4 and its maturity index is incremented three times. Thus, η_4 becomes a mature micro-cluster ($\mu_4 > \mu_{th}$). This means that η_4 is mature enough to be considered for the re-clustering phase. To do this, we refer to the macro-cluster magnetic attraction distance. This second parameter will allow us to find the list of overlapped micro-clusters within the macro-cluster magnetic attraction field but currently belonging to another macro-cluster which is the case for η_3. Overlapped clusters gets re-clustered as part of an updated micro-cluster's macro-cluster that we denote ζ_4. Macro-cluster id always takes the index of the micro-cluster launching the macro-clustering process.

By applying the aforementioned process, every incoming data sample is instantly clustered and outliers are easily identified. Such approach maintains arbitrary shaped data space regions of global clusters online. If the micro-cluster is still young a.k.a. outlier micro-cluster where $\mu_i < \mu_{th}$, then the macro-clustering will not be performed and DeDaSC passes to standby state waiting for a new data point to be processed.

5.4.4 DeDaSC complexity

The key to the success of the data stream algorithm is high scalability of the micro-clustering algorithm. This is because this process is exposed to a potentially large volume of incoming data and needs to be implemented in an efficient and online fashion. In this section we examine the time complexity of DeDaSC according to the size of the data stream (*i.e.* number of data points). For this purpose, we use the parameter n to represent the number of micro-clusters seen so far.

To assess the time complexity of our algorithm, we focus on the redundant iterations that will be performed every time a new data point is received and we limit our implementation to bi-dimensional and tri-dimensional streaming data sets. Therefore, we studied the neighborhood discovering using incremental Delaunay triangulation and, implicitly, the on-

line micro-clustering, from one side, and the re-clustering from the other side. As mentioned before, incremental Delaunay triangulation is based on two main steps :

1. Finding the container triangle location and inserting the new data point to the triangulation. This process requires $\mathcal{O}(n^{1/2})$ time to be achieved.
2. After the assignment of the newly incoming data point to a given micro-cluster, we compute the distances between the new or updated micro-cluster centroid and all other micro-cluster centroid with a complexity of $\mathcal{O}(n)$.

The resulting complexity is therefore $\mathcal{O}(n) + \mathcal{O}(n^{1/2})$. This low complexity results in an algorithm that is not only fast, but has a low time penalty for increasing volumes of data.

In this context, it is worth mentioning that while this incremental Delaunay triangulation algorithm can be generalized to more than three dimensions, the execution time can be exponential and its convergence is not guaranteed in some cases. Thus, whenever the data set dimension is higher than three dimensions, DeDaSC switches to the euclidean distance mode. We use the euclidean distance to find the nearest micro-cluster to the freshly received data point. In such situation, DeDaSC reaches higher complexity equal to $\mathcal{O}(n) + \mathcal{O}(n)$. Therefore, even with higher dimensions DeDaSC still perform in a fast fashion.

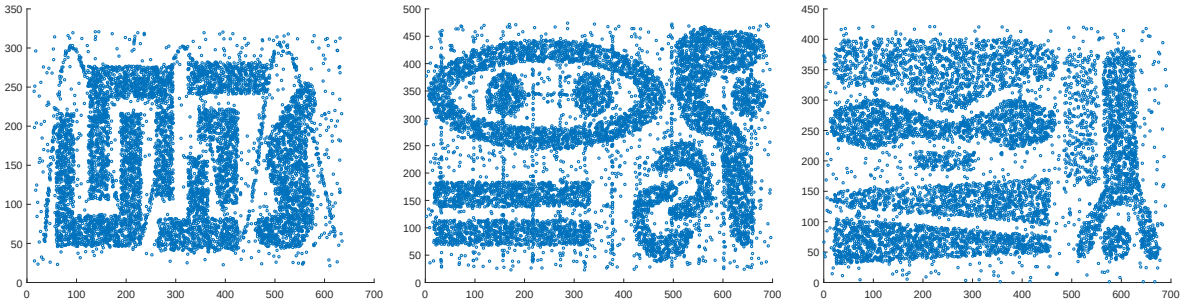
5.5 Experimental Evaluation

In this section, we provide a performance evaluation and a detailed discussion of our algorithm DeDaSC across a range of experiments. A thorough experimental study was conducted for the clustering quality, processing speed, memory efficiency, scalability, and sensitivity of the algorithm here proposed. We utilized synthetic datasets proposed in (Karypis et al., 1999) and commonly exploited by the data stream clustering research community. The three synthetic 2-dimensional datasets in a data streaming context termed as DS1, DS2 and DS3, are depicted in Figures 5.5(a), 5.5(b) and 5.5(c). Each of them contains thousands of data points.

— DS1 contains arbitrary shaped clusters, orientation and contains noise points as well as special artifacts such as streaks running across clusters.
— DS2 has the specificity of involving clusters inside the space enclosed by other ones.
— DS3 has clusters which are very close to each other with different densities. These clusters have non-convex shapes and some are interwoven.

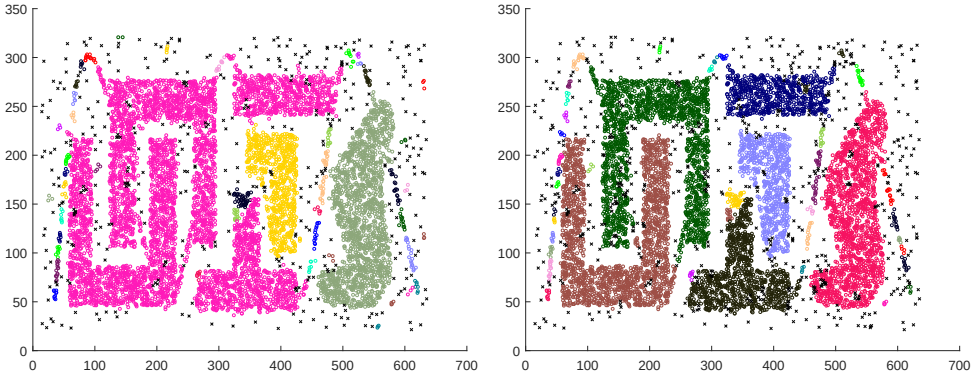
5.5.1 Sensitivity maturity threshold

As we explained in Section 5.4, clustering using DeDaSC is controlled by a small set of parameters. One of the most important parameters impacting the quality of the obtained

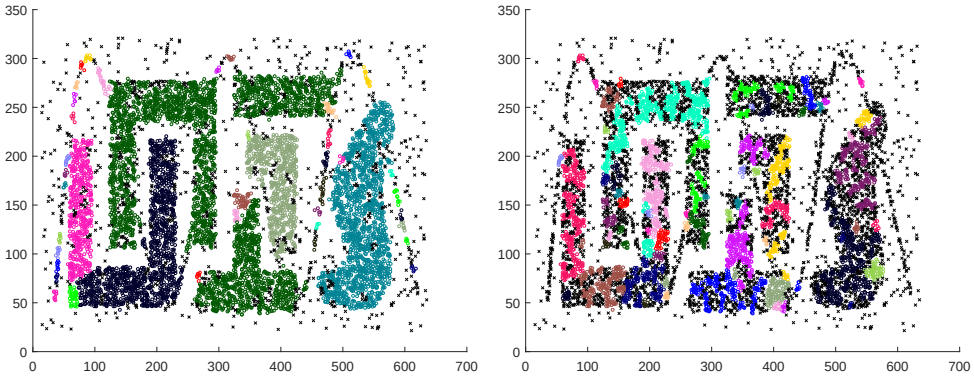


(a) DS1 (b) DS2 (c) DS3

Figure 5.5 The three data sets used in our experiments



(a) Clustered data using maturity index equal to 3 (b) Clustered data using maturity index equal to 4



(c) Clustered data using maturity index equal to 5 (d) Clustered data using maturity index equal to 10

Figure 5.6 Sensitivity of the clustering purity to the micro-clusters maturity threshold μ_{th} on the dataset DS1

clusters is the micro-cluster maturity threshold denoted μ_{th}. In this section, we evaluate how this parameter affects the quality of the clustering, considering different levels of upper

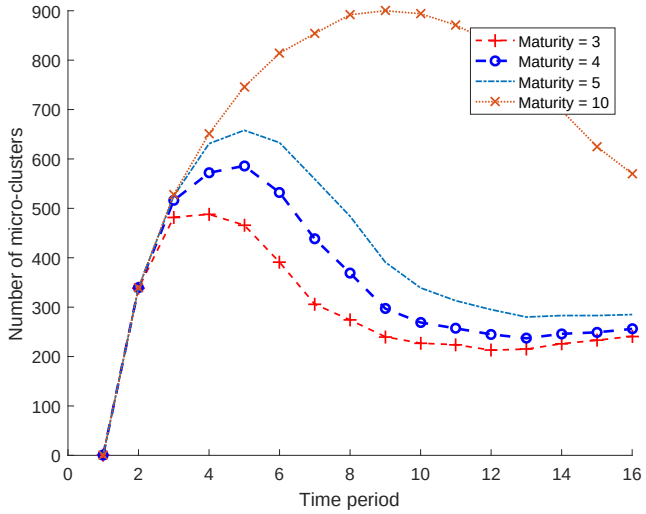
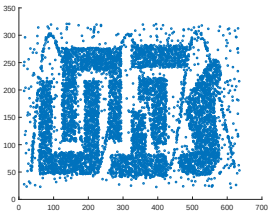


Figure 5.7 Impact of varying the maturity index on the memory usage of DeDaSC *i.e.* number of created micro-clusters

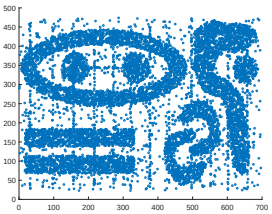
bound. In these experiments, we started by defining the parameter boundaries that we set to 2 and 10. For ease of comparison and reading, plots were limited to the values 3, 4, 5 and 10. Figure 5.6 shows a summary of the obtained results. When μ_{th} is set to a relatively small or high value, DeDaSC leads to very poor results. For example, when $\mu_{th} = 2$, only one macro-cluster is obtained and DeDaSC failed to discover the various clusters. When $\mu_{th} = 3$, new clusters start appearing, however, interwoven clusters fail to be separated. In figure 5.6(b), where μ_{th} is set to 4 the clustering quality is widely better than all the other possible values within the boundaries. Not only will the clustering quality be affected by the choice of the maturity index but also the memory usage and run-time. Figure 5.7 shows the impact of varying μ_{th} with regards to the number of generated micro-clusters. Figure reports the number of micro-clusters created by DeDaSC during its execution. The most important result showed by Figure 5.7, however, is that, the higher the level of maturity threshold, the worse DeDaSC performs with respect to the other values. We intentionally omitted the performance of DeDaSC with $\mu_{th} = 10$ because, being too big, did not offer any interesting insight. The form of the other plots shows that DeDaSC converges and covers the whole space in such a way that any falling new data point will have higher chances to join an existing micro-cluster than to create its own.

5.5.2 Clustering quality evaluation

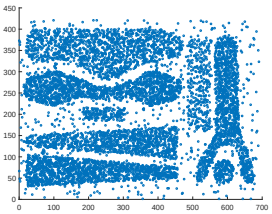
The sequence order of streaming data have a considerable effect on the clustering purity. To efficiently evaluate the quality of DeDaSC output, we generate the data sequentially at each time step. The data flow generation is performed in a random fashion. At each time, any



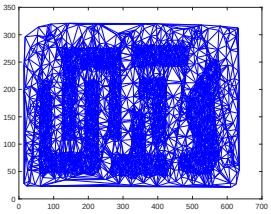
(a) Original dataset DS1 with noise



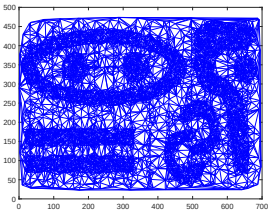
(b) Original dataset DS2 with noise



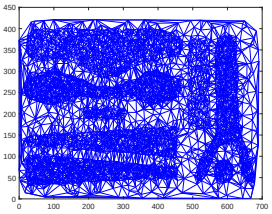
(c) Original dataset DS3 with noise



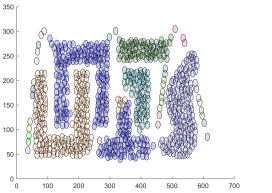
(d) Delaunay triangulation of DS1



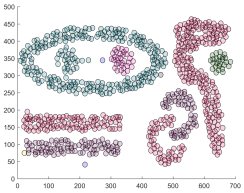
(e) Delaunay triangulation of DS2



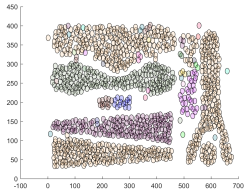
(f) Delaunay triangulation of DS3



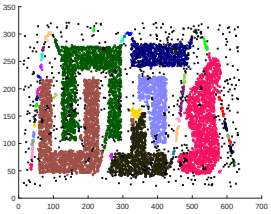
(g) Created micro-clusters of DS1



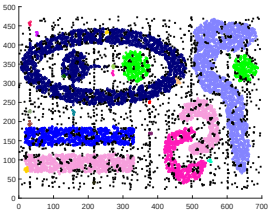
(h) Created micro-clusters of DS2



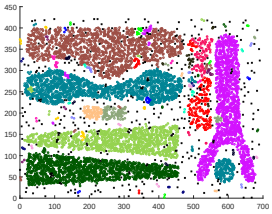
(i) Created micro-clusters of DS3



(j) DS1 data points placed into macro-clusters with $\mu = 4$



(k) DS2 data points placed into macro-clusters with $\mu = 4$



(l) DS3 data points placed into macro-clusters with $\mu = 2$

Figure 5.8 The three datasets used in our experiments (DS1, DS2, DS3) and their resulting Delaunay triangulation. In addition we give the generated micro-clusters coloured by their macro-cluster's membership. At the end we give the snapshot of data stream resulting clustering. Some data points inside the macro-clusters still unclustered since they belong to non-mature micro-clusters.

data point that has not been selected is equally likely to be picked as the new data record.

The final clustering results obtained by DeDaSC are shown in Figures 5.8(j), 5.8(k) and 5.8(l). The data points belonging to different clusters are marked by different colors. From these Figures, we can easily notice that DeDaSC can correctly discover the genuine clusters without users supply on the number of clusters in all three datasets DS1, DS2 and DS3 or a fixed pre-defined density index. Therefore, we can conclude that as this experiment illustrate DeDaSC is very effective in finding various clusters of arbitrary shape, density, and orientation, and is tolerant to outlier points, as well as artifacts such as streaks running across clusters. In Figures 5.8(g), 5.8(h) and 5.8(i), we show that the overlapped micro-clusters will be grouped together. In this context, it is worth to mention that only this information will be accessible during run-time. The original data once scanned it will be discarded and only the micro-clusters are updated.

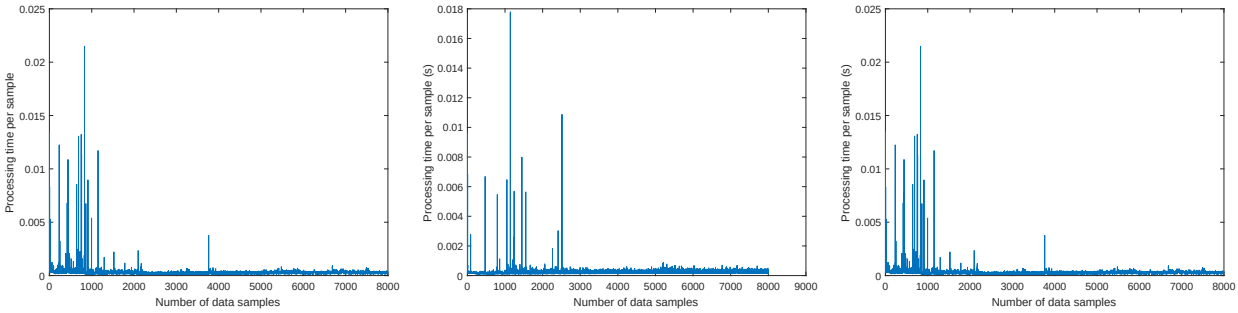
5.5.3 Processing Time

The key factor to the efficiency of DeDaSC is the rapid processing time of the micro-clustering creation process. The micro-clustering is exposed to a voluminous steaming data. The most time-consuming operation during this stage is that of finding the nearest micro-cluster for each freshly received data point. The use of the incremental Delaunay triangulation fixes the problem of spatial proximity among an irregularly distributed spatial datasets and rapidly finds the nearest micro-cluster. In this context, by rapidly, we mean in a square root time complexity time with the number of micro-clusters.

In Figure 5.9, we show the processing time of each data point of the streaming data. The vertical lines represent the relative start and stop times at which each data point was processed. Longer lines identify samples that took longer to process. At the beginning, DeDaSC requires relatively higher execution time to construct the initial set of micro-clusters. During this transient phase, incoming data points fall in empty regions and each operation leads to the creation of a new micro-cluster. However, once DeDaSC reaches convergence, the clustering process becomes faster. This convergence can be explained by the fact that new data points will have higher chances to be absorbed by an existing micro-cluster. At this advanced stage, DeDaSC covers appropriately the space.

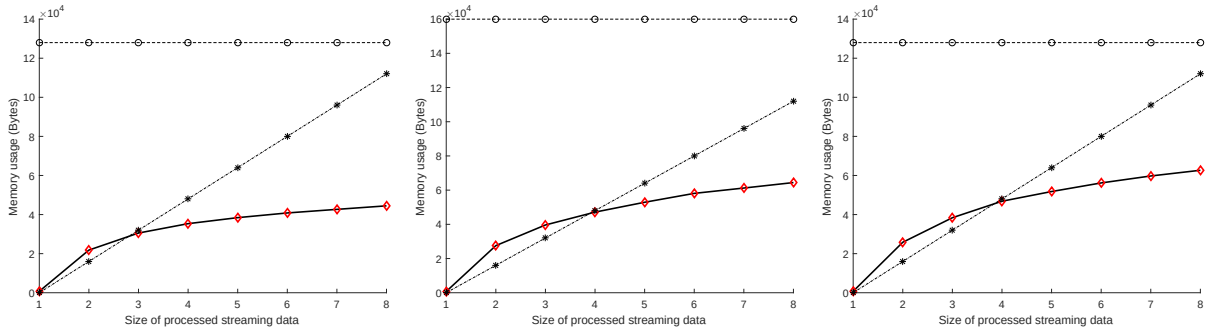
5.5.4 Memory Usage

Data stream clustering algorithms are required to have a relatively small memory usage. Even a linear growth of memory consumption is not accepted. In order to evaluate the memory usage of DeDaSC compared to the incremental building (used in the context of widowed or online/offline clustering algorithms) and the whole dataset (used in the context



(a) DS1 (b) DS2 (c) DS3

Figure 5.9 Evolution of the processing time of DeDaSC with every incoming data point over time



(a) DS1 (b) DS2 (c) DS3

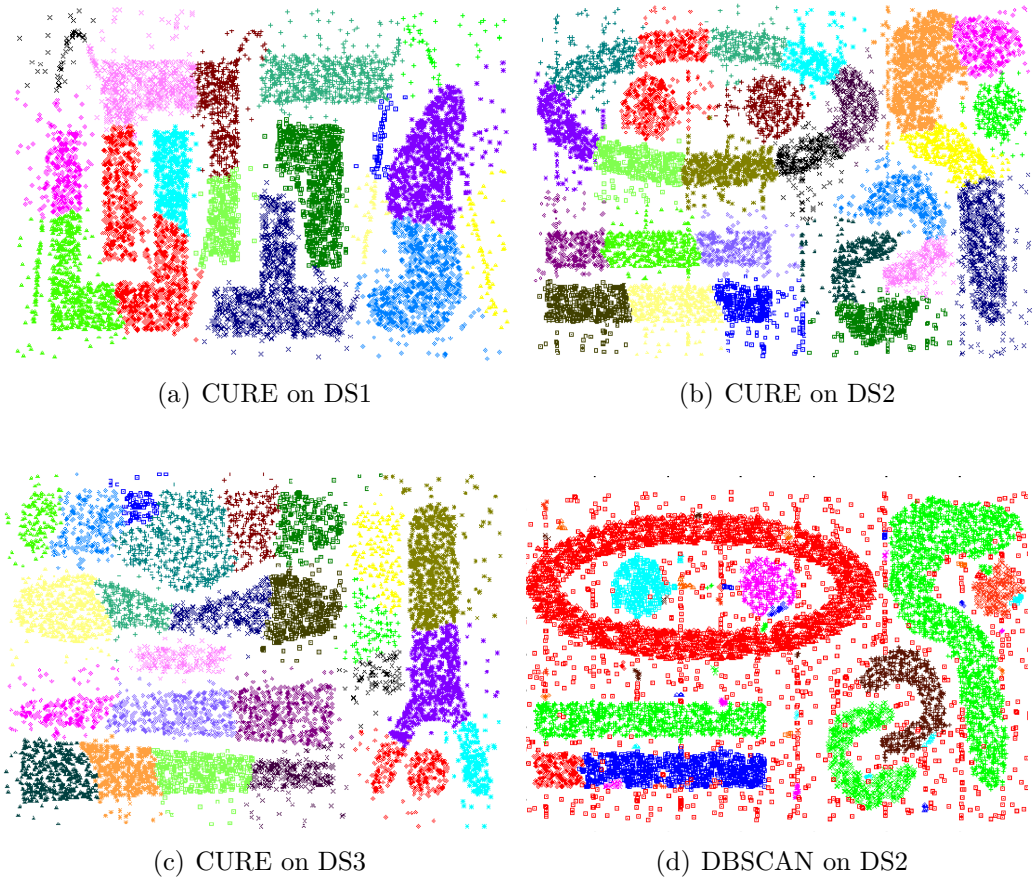
Figure 5.10 Evolution of the memory usage of DeDaSC with every incoming data sample over time compared to the whole and incremental building of the studied datasets.

of conventional or offline clustering algorithms). In our implementation, DeDaSC maintains an up-to-date micro-cluster model containing useful metadata (centroid location, maturity index, etc.). The memory usage is measured in terms of the number of saved micro-clusters. Figure 5.10 shows that the memory usage of DeDaSC is limited and bounded as the streams proceed. This upper bound is reached when DeDaSC achieves the steady state and the space is fully covered.

5.5.5 Clustering Results Comparison

We evaluated the performance of DeDaSC against CURE and DBSCAN as two samples of the current online clustering algorithms. The obtained clusters of DS1, DS2 and DS3 using these two algorithms were given in (Karypis et al., 1999). The experiments with CURE algorithms

were conducted with a shrinking factor [1] equal to 0.3 and the number of representative points is 10. In figures 5.11(a), 5.11(b) and 5.11(c), we can easily notice that CURE failed to discover the right clusters with arbitrary shapes. This erroneous result is due to the fact that CURE makes errors when merging sub-clusters by ignoring their density and focusing only on their proximity. On the other hand, DBSCAN which is considered as the best known and most used density-based clustering algorithms capable of discovering arbitrary shaped clusters. DBSCAN succeeded in finding the right clusters for datasets DS1 and DS3. However, DBSCAN failed with DS2 as this dataset contains clusters with variable internal density.



(a) CURE on DS1

(b) CURE on DS2

(c) CURE on DS3

(d) DBSCAN on DS2

Figure 5.11 CURE on the DS1, DS2, DS3 datasets with shrinking factor 0.3 and number of representative points 10. DBSCAN on the DS2 dataset with Eps parameter equal to 5.9 (Karypis et al., 1999).

1. A factor allowing the shrinking of the scattered points within the cluster toward the mean of the cluster.

5.6 Conclusion

The Internet of Things concept arises from the need to manage, automate, and explore all devices, instruments, and sensors in the world. In order to make wise decisions both for people and for the things in IoT, data mining technologies are integrated with IoT technologies for decision-making support and system optimization. Data stream clustering allows the discovering of the novel, interesting, and potentially useful patterns from data and applying algorithms to the extraction of hidden information. In this paper, we have developed an effective and efficient method that we called it Delaunay triangulation-based Data Stream Clustering algorithm (DeDaSC) for clustering voluminous evolving data streams. This is an incremental, one-pass density-based algorithm, which is able to construct arbitrary shaped in real time with considerable optimization of time and space. DeDaSC has intuitive advantages over traditional techniques which try to cluster the whole dataset at one time rather than processing the data stream evolving over time. We evaluate DeDaSC against state-of-the-art stream clustering algorithms on synthetic data streams. We show DeDaSC capabilities when it comes to fuse micro-cluster to create arbitrary shaped clusters as they evolve in a fully online manner. The study validates the following claims : DeDaSC derives higher quality clusters than traditional stream clustering algorithms, especially when the cluster distribution contains dramatic changes ; DeDaSC proves its capabilities in terms of timing and memory efficiency ; DeDaSC has very good scalability in terms of data stream size, and the number of clusters.

CHAPTER 6 ARTICLE 2 : MULTI-OBJECTIVE MAPPING OF FULL-MISSION SIMULATORS ON HETEROGENEOUS DISTRIBUTED MULTI-PROCESSOR SYSTEMS

Rabeh Ayari, Imane Hafnaoui, Alexandra Aguiar, Patricia Gilbert, Michel Galibois, Jean Pierre Rousseau, Giovanni Beltrame, Gabriela Nicolescu

Published to Journal of Defence Modeling and Simulation

Abstract : Full-Mission Simulators (FMS) are considered as the most critical simulation tool belonging to the flight simulators family. FMSs include a faithful reproduction of fighter aircraft. They are used by armed forces for design, training and investigation purposes. Due to the criticality of its timing constraints and the high computation cost of the whole simulation, FMSs need to run in a high-performance computing systems. Heterogeneous distributed systems are among the leading computing platforms and can guarantee a significant increase in performance by providing a large number of parallel powerful execution resources. One of the most persisting challenges raised by these platforms is the difficulty to find an optimal mapping of n tasks on m processing elements. The mapping problem is considered as a variant of the quadratic assignment problem, in which exhaustive search cannot be performed. The mapping problem is an NP-hard problem and solving it requires the use of meta-heuristics and it becomes more challenging when one has to optimize more than one objective with respect to the timing constraints. Multi-objective evolutionary algorithm have proven their efficiency when tackling this problem. Most of the existent works deal with the task mapping by considering either a single objective or homogeneous architectures. Therefore, the main contribution of this paper is a framework based on the model-driven design paradigm allowing us to map a set of intercommunicating real-time tasks making up the FMS model on the heterogeneous distributed multi-processor system model. We propose a multi-objective approach based on the well-known optimization algorithm Non-dominated Sorting Genetic Algorithm-II satisfying the tight timing constraints of the simulation and minimizing makespan, communication cost and memory consumption simultaneously.

6.1 Introduction

A Heterogeneous Distributed Multi-processor System (hereafter named "HDMS") is a complex yet powerful architecture that refers to a collection of autonomous interconnected multi-processor machines with various capabilities (Khokhar et al, 1993). This type of architecture

was developed to meet the high computation and timing requirements of real-time systems and have been widely used in industry, covering different domains such as aerospace, automotive and avionics. One of the most critical systems that needs to be executed on large distributed computing systems are Full-Mission Simulators. An FMS is a complex simulator allowing the crew to practice regular flight operations, fire weapons and familiarize themselves with the cockpit in normal and extreme situations by faithfully reproducing the aircraft and the mission environment in which it will operate. Moreover, many FMSs are able to communicate between each other in the distributed mission operation. The simulation provides an environment allowing military forces to train as they fight. For further details regarding the FMSs, please refer to section 2.

Exploiting the full potential of HDMS for improving FMSs performance is a current challenge in aerospace and defense fields. The FMS software is contributed to by different teams with different specific backgrounds (e.g. mechanic, hydraulic, electronic). Therefore, efficient FMS execution on a HDMS requires a global view of the FMS and deep knowledge of the execution platform. In this context, we propose a new approach for efficiently improving the FMS performance using the HDMS. This approach is based mainly on a model-based paradigm. Starting from FMS and HDMS models, we apply our optimization algorithm for mapping the different real-time tasks making up the FMS on the components of the HDMS. The FMS and the HDMS models represent the basis for collaboration between the different teams involved in the FMS design while the optimization algorithm using as inputs the FMS and HDMS models facilitate integration.

The problem of finding an optimal mapping of n tasks on m heterogeneous processing elements is NP-hard (Kang et al., 2011). In concrete words, if we assume that our aim is to map an application composed of 16 tasks on a quad-core architecture, the solution space that will be explored to find the optimal mapping schema contains $4^{16} = 4\,294\,967\,296$ possible solutions. Thereby, if evaluating one solution takes 0.1 second, evaluating the overall solution space takes more than 13 years. Therefore, the solution space defined as the set of possible mapping schemes is likely to be very large and checking whether each candidate satisfies the problem's constraints in addition to evaluating its fitness is unrealistic. While this exhaustive search technique is easy to implement and always leads to the optimal solution, its cost is prohibitive. Thus, the complexity involved in finding the optimal solution has been the main reason to accept good or near optimal solutions i.e. trading off accuracy for a faster exploration. These trade-offs have motivated the use of several meta-heuristics to tackle this issue. Based on the "no-free-lunch-theorem" (Wolpert and Macready, 1997), previous works and a set of experiments conducted internally, Evolutionary algorithms proved their efficiency in solving our problem. However, this problem has not been widely studied in the case of

HDMS compared to homogeneous systems. Taking the real-time constraints of FMSs into consideration makes it more complex.

In this work, by analyzing existing FMSs, we extract a series of key performance metrics. The solutions are rated according to three metrics : makespan, communication cost and memory consumption. As these metrics might be conflicting, suitable trade-offs are considered. To the best of our knowledge, there is no other work in the literature that addresses all the following research axes simultaneously : heterogeneity of the target architecture, dependency among the tasks modeling the system, the respect of tight timing constraints and multi-objective optimization. Accordingly, the main contribution of this paper is that we are providing an efficient framework for mapping real-time tasks implementing the FMS among the available components of the HDMS with respect to the hard real-time constraints and minimizing at the same time makespan, communication cost and memory consumption.

The remainder of this paper is organized as follows. In Section 6.2 we briefly review some basic concepts. Section 6.3 discusses related work. In Section 6.4, we formally model the FMS in the context of our problem. In Section 6.5 we detail our implementation and we illustrate it by a set of experiments described in Section 6.6. Finally, Section 6.7 presents concluding remarks and lists our future work.

6.2 Background information

This section provides a broad overview of the basic concepts applied in this work. Thus, the reader who is familiar with such concepts may skip to the next section.

6.2.1 Full-Mission Simulators

In the fast growing aviation industry, military aircraft contribute substantially. Military training rely heavily on hands-on experience with the training engines whether it be aircraft, tanks, ships, or submarines. One could easily imagine the high cost of training new recruits on these different devices. National Training Systems Association (NTSA) reports that for the fiscal year of 2000, flying an F-16 fighter costs an estimated $\$5,000$ per hour. That aside, the dangerous environment and difficulty of training might lead to human casualties as well as infrastructure damages that will cost billions of dollars. To mitigate these issues, simulators have long been employed in civil industry and especially military to reduce cost and casualties. Simulators are nowadays used for training on all various types of military training grounds such as Air Force pilots to fly fighter aircraft and Apaches, Navy officers to navigate ships and submarines, etc. Even better, the trainees have easily adapted to the use of

simulators as a training tool due to the fact that they come from a generation acquainted with simulations such as video games and virtual reality. These simulators will reproduce the sounds, motion, visual scenes, the representation of all the other instruments and systems to create a realistic training environment. This eased the integration of simulators which are extremely cost effective and provide a safe environment. In the NTSA report, training on an Apache helicopter, the cost is estimated at $\$3,101$ per hour ; whereas on a simulator, the cost plummets to \$70 per hour (Kennedy, 1999). FMSs are devices that artificially re-create aircraft flight and the external environment. These simulators are extremely complex, as they must replicate the equations that govern aerodynamics, flight controls, weapon load-out, and how the aircraft reacts to environmental factors such as air density, turbulence, wind shear, precipitation, etc.

6.2.2 Real-time systems

Systems are referred to as real-time when their correct behavior depends not only on the proper functioning of the operations they perform, but also on the time at which they are performed by respecting the system's deadline (Davis and Burns, 2011a). Therefore, in real-time applications, the timing requirements are the main constraints and controlling them is the predominant factor for assessing the quality of service (et al., 2002).

We distinguish two classes of real-time timing constraints ; hard and soft, depending on the criticality of timing constraints. We consider a real-time system as hard when a timing faults may cause some human or economic disaster. A real-time system is considered as soft when timing faults cause only some performance degradation. In terms of modeling, we use the term tasks to refer to the basic executable entities. These tasks could be periodic or aperiodic.

Regardless of the tasks' behavior, it is important to state that some of them, once elected for execution, may be interrupted to allocate the processor to another one. Due to such behavior, they are called preemptive. On the other hand, when an elected task should not be interrupted before the end of their execution, it is called non-preemptive.

6.2.3 Multi-objective optimization

Problems with more than one objective have the distinction of being much more difficult to treat than their mono-objective equivalent. The difficulty lies in the absence of a ranking criteria to compare solutions. A solution may be better than another on some objectives and worse on others (Yuanlong Chen and Ma, 2012). The solutions found by a multi-objective optimization approach have to be optimal with respect to distinct objectives, typically conflic-

ting. It is not possible to find an optimal solution that satisfies all objectives but rather a pool of efficient solutions characterized by the fact that their cost cannot be improved in one dimension without being worsened in another as depicted in Figure 6.1. That is why the concept of optimal solution becomes less relevant in multi-objective optimization. These solutions form the Pareto optimal front referring to the economist Vilfredo Pareto (Ehrgott, 2012).

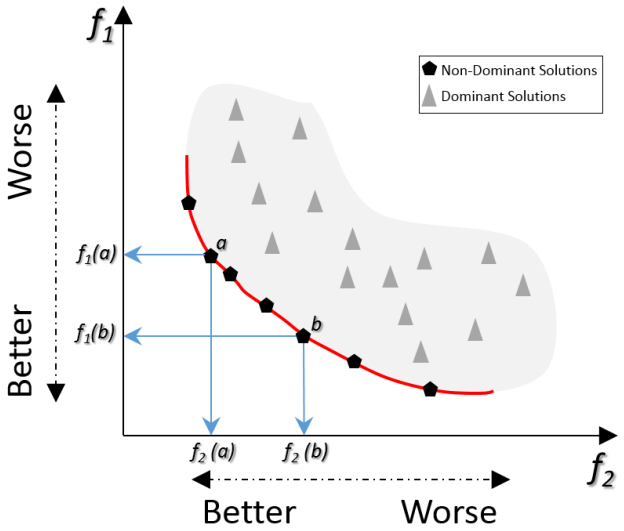


Figure 6.1 Example of a Pareto front. The gray points represent feasible choices, and pentagon values are preferred to the gray ones. Gray points do not belong to the Pareto frontier because it is dominated by both point A and point B. Points A and B are not strictly dominated by any other, and hence do lie on the frontier.

Mathematically, the multi-objective optimization problem is defined as follows :

Ω = Decision Space ; Ψ = Solution Space
$X = (x_1, x_2, ..., x_n) \in \Omega$; X is a vector of n decision variables
$F = (f_1, f_2, ..., f_m)$; m is number of objective functions

$$Y = F(X) = (f_1(X), f_2(X), ..., f_m(X)) = (y_1, y_2, ..., y_m) \in \Psi \tag{6.1}$$

In the literature, many approaches have been developed to address this problem and could be classified into two main categories :

Scalar or weight-based approach. Weight-Based approach consists of formulating a single-objective optimization problem such that its optimal solutions are optimal solutions to the multi-objective optimization problem. This technique is one of the oldest techniques in multi-objective optimization using heuristics such as Genetic Algorithms (GAs) (Syswerda and Palmucci, 1991), (Jakob et al., 1992), (Yang and Gen, 1994) and Simulated Annealing (Se-

rafini, 1992). Since setting a weight vector leads to a single point, to find different solutions with various trade-offs, the optimization process is performed with different weight vectors which produce an extensive computational cost and the decision maker have to set the most suitable weight combinations to reproduce a representative part of these Pareto solutions. Furthermore, a main technical shortcoming of this approach is that the non-convex points of the pareto front are unreachable.

Pareto approach. The Pareto approach directly uses the concepts of dominance in the solutions generation. Therefore, the Pareto optimum gives more than a single solution, but rather a set of solutions called non-dominant solutions. The main advantage of these approaches is the simultaneous optimization of conflicting objectives. NSGA (Srinivas and Deb, 1994), NSGA-II (Deb et al, 2002), SPEA (Zitzler and Thiele, 1999) and SPEA-II (Guliashki et al., 2009) are among the most known multi-objective algorithms based on this technique.

6.3 Related Work

It is now more than a quarter of a century since researchers started publishing papers on distributing computation across the execution resource of parallel architectures. This section is a sampling of related literature and is not meant to be exhaustive. It covers the different levels of parallelism, the system heterogeneity, the optimization objectives and whether the methodology considers real-time constraints.

In (Braun et al, 2001), Braun et al. presented a comparison among static heuristics for mapping applications onto heterogeneous distributed computing systems aiming to minimize the makespan. Through this comparison, which included GAs, tabu search, simulated annealing among others, the experimental results showed that GAs consistently gave the best results for the parameters and implementation used in this exhaustive study. Nevertheless, this work is limited to independent tasks and a single objective. Owing the relevance of biologically inspired methodologies for scheduling, Tumeo et al. (Tumeo et al., 2008) provided an ant colony optimization for mapping and scheduling in heterogeneous multiprocessor systems. Their solution showed 64% and 55% better results compared to Simulated Annealing and Tabu Search respectively. In (Lu and He, 2013), the authors proposed a PSO-based GA hybrid algorithm to schedule a set of tasks on a heterogeneous multi-processor system to minimize the makespan taking into account the precedence constraints. The schedule obtained outperforms the GA algorithm and is within 9% of the optimal schedule. In (Kang and Zhang, 2012), Kang and Zhang presented a hybrid GA for static task mapping and scheduling on heterogeneous systems with satisfaction of the precedence requirements of the application.

A study of the efficiency of Multi-Objective Evolutionary Algorithms in the task mapping and scheduling on heterogeneous systems is performed with two different objectives in (Chitra et al, 2010) ; minimizing the makespan and the average flow-time. The authors in (Devi and Anju, 2014) proposed a multi-objective scheduling algorithm using Multi-Objective Evolutionary Algorithm (MOEA) for scheduling a collection of dependent tasks on available resources in a multiprocessor environment. NSGA-II is used to get the Pareto optimal solutions to minimize at the same time the makespan and the reliability cost. Rajeswari et al. (Rajeswari et al., 2014) presented an efficient allocation and scheduling method using multi-objective GA for independent tasks. Such procedure minimizes the makespan and flow-time simultaneously in a distributed computing system.

In (Samur and Bulkan, 2010) the authors presented a GA approach to solve a bi-criteria problem (makespan and tardiness) in homogeneous parallel machines and tried to make their method fairly general to be applied to some other bi-criteria objective functions. In (Saranya et al, 2009), the authors present a parallel multi-objective GA for the task dispatching problem in heterogeneous distributed computing environment. It exploits the inherent parallelism of GA on a multi-core processor to optimize the result. In (Navaz and Ansari, 2012), Navaz and Ansari considered the two objectives ; execution time and total cost. They used the R-NSGA-II approach based on evolutionary computing paradigm and uses an epsilon dominance based MOEA. In (P. Serafini and Dehuri, 2011), a Combinatorial Multi-objective Particle Swarm Optimization Based Algorithm is proposed to map a set of dependent tasks on a heterogeneous systems with consideration of failure on processors and links. The obtained results outperform the NSGA-II. However, the authors don't consider the tuning of the NSGA-II parameters.

In (Bernabe Dorronsoro, 2010), Dorronsoro et al. investigated the problem of multi-objective mapping on Grids, optimizing simultaneously the makespan and its robustness. For this purpose, four different MOEAs were studied. These algorithms are NSGA-II, MOEA/D, IBEA and MOCell. From their experiments, the latter lead to the best results compared to the others. M. Miryani and al. (Miryani and Naghibzadeh, 2009b) scheduled hard real-time systems on heterogeneous multi-processor systems taking into account the precedence relationship between tasks. Likewise, the authors studied the mapping problem from a multi-objective perspective aiming to minimize the completion time and the number of processors.

After this brief description, Table 6.1 depicts a detailed comparison of these works. In terms of the comparison's characteristics, we identify the hardware architecture, real-time aspects, communication between tasks and optimization strategy.

Table 6.1 Comparison of Related Work on Multi-processor Mapping and Scheduling

Characteristics	Hardware Architecture			Real-time Aspects		Communication	Optimization		
Author	Heterogeneity	Distributed	Multi-Procesor	Real-Time	Preemption	Dependency	Multi-Objective	Algorithm	Objectives
Braun et al.	✓	✓	✓	✗	✗	✗	✗	11 heuristics	Makespan
Tumeo et al.	✓	✗	✓	✗	✗	✓	✗	Ant Colony	Makespan
Kang et al.	✓	✓	✓	✗	✗	✗	✗	PSO-based Genetic Algorithm	Makespan
Kang et al.	✓	✗	✓	✗	✗	✓	✗	Hybrid Genetic Algorithm	Makespan
Chitra et al.	✓	✗	✓	✗	✗	✓	✓	Weight-Based MOGA MOEA	Makespan flow-time Reliability Cost
Devi et al.	✓	✗	✓	✓	✗	✓	✓	NSGA-II	Makespan Reliability Cost
Rajeswari et al.	✓	✓	✗	✗	✗	✗	✓	Weight-Based Genetic Algorithm	Makespan Flow-time
Samur et al.	✗	✓	✗	✗	✗	✗	✓	Weight-Based Genetic Algorithm	Makespan Tardiness
Saranya et al.	✓	✓	✓	✗	✗	✗	✓	Parallel Multi-Objective Genetic Algorithm	Makespan Flow-Time
Navas et al.	✓	✓	✗	✗	✗	✗	✓	NSGA-II	Makespan Total Cost Reliability Cost
Roy et al.	✓	✓	✓	✗	✗	✓	✓	Combinatorial Multi-Objective Particle Swarm	System Cost System Reliability
Dorronsoro et al.	✓	✓	✗	✗	✗	✗	✓	NSGA-II, MOEA/D IBEA and MOCell	Makespan Robustness
Miriany et al.	✓	✗	✓	✓	✗	✗	✓	MOGA	Completion Time Number of Processors
Our Work	✓	✓	✓	✓	✓	✓	✓	NSGA-II	Makespan Communication Cost Memory Consumption

6.4 FMS Model

An FMS is a very complex piece of software and exhibits a high-level of system integration. Several system models need to communicate data to other systems to replicate functions of the aircraft. When developing military flight simulator software, the development team has to rely on the competence of experts in different areas and with expertise from a multitude of domains, such as mechanics, power electronics, avionics, and more. The use of different models and their combination for the production of software is referred to as Model-Based Design. This multi-domain modular approach to designing FMSs allows specialists to build, configure and test their modules concurrently, regardless of other modules. After initial testing, the modules are connected and characterized accordingly to form the complete simulation software for test and validation purposes. This latter process is called system integration. Such complex activity results in a network of sub-systems that need to be interconnected together in order to deliver the platform's intended functionality.

Aiming at automating and optimizing the integration process by efficiently mapping these real-time sub-systems (hereafter named tasks) implementing the FMS to the available resources of the HDMS. We formally define the graph modeling the FMS as $G =< T, E >$,

where $T = \{t_i | i \in \{1, ..., n\}\}$ is a set of n vertices, each one representing a real-time task and $E = \{e_{ij} | (i, j) \in \{1, ..., n\} \times \{1, ..., n\}\}$ is a set of edges that represents the dependencies among these tasks. The edges are weighted by the amount of data exchanged between each pair of connected tasks. Such graphs are commonly referred to as task graphs. We assume that there are n tasks and m processors belonging to different machines. In this paper, the precedence constraints are not considered. Therefore, from the scheduling point of view, the tasks are independent. Also, we assume that all the tasks are preemptive. Each processing unit can execute one task at a time, so that the system can process m tasks concurrently. Each task is defined as a tuple $t_i =< id_i, p_i, C_i, d_i >$, where : id_i is the ith task id, p_i is the task period i, C_i is the task execution time vector modeling the expected execution time c_{ij} to complete task t_i on processor p_j, and d_i is the deadline of the task t_i. In a mathematical formulation of the mapping problem, we define the function Φ that assigns each of n tasks to any of the m processors :

$$\Phi : t_1, ..., t_n \rightarrow p_1, ..., p_m$$

$$\exists j \in [1, m], \forall i \in [1, n] \ s.t. \Phi(t_i) = p_j \tag{6.2}$$

We start by defining the tasks expected execution time on the $HDMS$. This information is defined by specifying a $(n \times m)$ matrix named Execution Time Matrix (ETM), where c_{ij} is the expected execution time of the task i on processor j. This model expresses the execution heterogeneity among the $HDMS$ resources. The elements along a row indicate the execution times of a specific task on the different processing units, and those along a column indicate the expected execution time of all tasks on a single node. We also need to define the amount of data exchanged between each tasks couple. This information is defined in a square matrix $(n \times n)$ that we called Communication Cost Matrix (CCM), where $CCM(i, j)$ is the the weight of the edge connecting t_i to t_j and defined as e_{ij}. This matrix is known as the adjacency matrix. CCM is an upper triangular matrix since the data exchanged between tasks t_i and t_j are the same as the amount of data exchanged between tasks t_j and t_i. The main diagonal entries are equal to zero. Another important feature to consider in our model is the memory consumption of each task. This information is defined in a Memory Consumption Vector (MCV) with n elements where each term m_i describes the amount of memory needed to execute task t_i. For the scheduling strategy, we are using the Rate Monotonic algorithm (Liu and Layland, 1973). This is a fixed-priority algorithm that assigns priorities to tasks according to their periods. The static priorities are assigned on the basis

of the task's period : the shorter the cycle duration is, the higher is the task's priority.

$$p_i \leq p_j \Leftrightarrow \Pi_i \geq \Pi_j \tag{6.3}$$

Liu & Layland (Liu and Layland, 1973) proved that a feasible schedule, that will always meet deadlines, exists if the processor utilization U is below a specific bound. For each task set Θ_j composed of k tasks assigned to processor j this test has to be performed with success. This test has to succeed in all the processors of the target architecture to obtain a valid mapping solution. Thus, a necessary condition to verify the schedulability of Θ_j based on the processor utilization bound is defined by :

$$U = \sum_{i=1}^{k} \frac{c_{ij}}{p_i} \leq k.(2^{\frac{1}{k}} - 1) \tag{6.4}$$

6.5 Proposed solution

To achieve a valid mapping solution respecting all the timing constraints and minimizing the defined fitness functions, our solution uses an approach involving two phases. Firstly, we assign the tasks implementing the FMS onto the available resources of the target architecture depending on the objective functions. Then, we perform the schedulability test of the assigned tasks to each processor of the execution platform to ensure the correctness of the solution from a scheduling point of view. The system model and the mapping approach are depicted in Figure 6.2.

Our optimization approach is implemented on top of the NSGA-II taking into account schedulability. The studied fitness functions of our implementation are set out in more details below. We are using closed-form expression for the three objectives to avoid the timing cost raised by evaluation using simulation.

Makespan (Overall Execution Time) : is the total length of the schedule when all the tasks on each processor finish their processing. Make-span is the time interval between the start of the first task and the completion time of the last task of Θ_j. In this example, the makespan fitness function is defined as :

$$f_1 = min(\max_{1 \leq j \leq m} (\sum_{i \in \Theta_j} c_{ij})) \tag{6.5}$$

In Figure 6.3 an example of scheduling eight tasks on two processors is provided. The makespan is equal to 7.

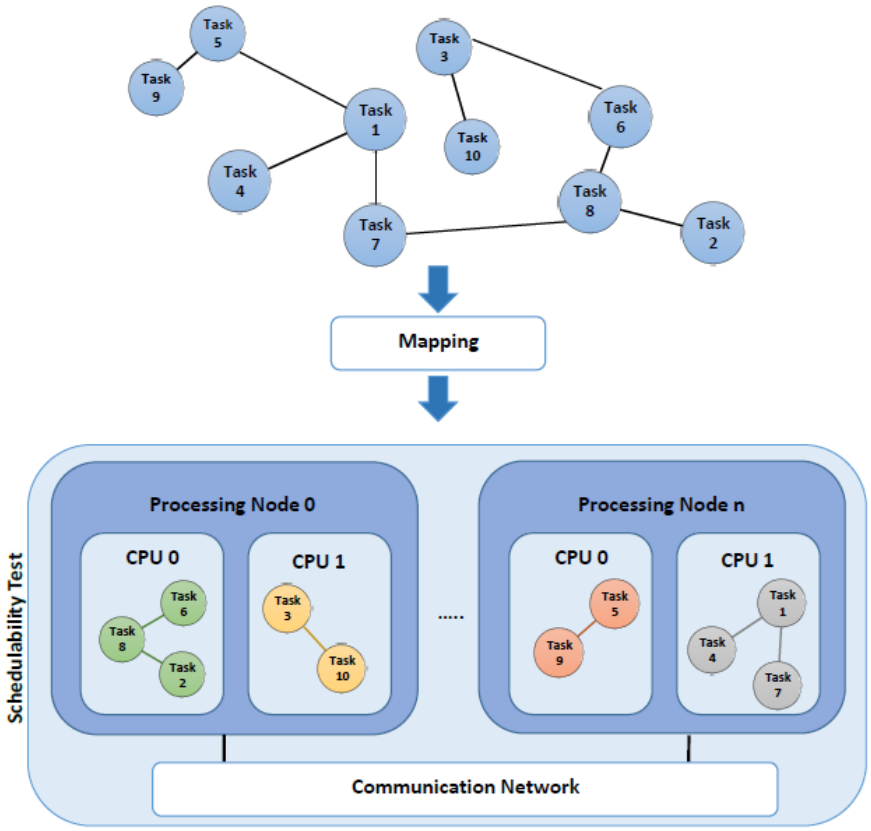


Figure 6.2 Mapping strategy of the FMS on the multi-processor target execution platform

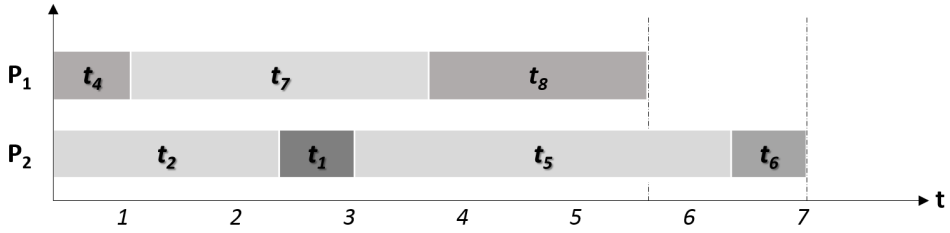


Figure 6.3 Example of scheduling eight tasks on two processors showing the obtained makespan

Communication Cost : one of the most important sources of overheads in the computation time in $HDMS$ occurs from data transmission between processing units in the network and reducing it represents the second fitness function of this study. A mathematical formulation of this fitness function is given below :

$$f_2 = min(\sum_{i=1}^{n}(\sum_{j=1}^{n}(e_{ij})))/i \in \Theta_\alpha; j \in \Theta_\beta; \alpha \neq \beta \tag{6.6}$$

Memory Consumption : the third objective is minimizing the total amount of memory consumption per processor. In other words, well distribute the memory consumption between processing elements. Seen from this angle, the problem is considered as a variant of the bin packing problem (Coffman et al., 1978) in which the bins refer to the amount of memory required to execute each task. The memory consumption fitness function is defined as :

$$f_3 = min(\max_{1 \leq j \leq m}(\sum_{i \in \Theta_j} m_i)) \tag{6.7}$$

In the bio-inspired GA, a given population represents a set of solutions to the problem and new generations are created through genetic operators. According to the evolution theory, only the strongest individuals of the population are likely to survive and generate offspring, transmitting their biological inheritance to the next generation. NSGA-II varies from GA not only in the fact that it addresses multi-objective optimization problems but also in the selection operation. It also gives the non-dominated solutions belonging or near the pareto front in one single run. Before selecting a number of individuals to apply the genetic operators on, the population is ranked on the basis of the non-dominance and crowding distance concepts. Below, we give more details about how we are implementing and adapting the NSGA-II to our problem to minimize the fitness functions and deliver valid solutions from a scheduling perspective.

The pseudo-code of the proposed solution is given in algorithm 3.

Individual Encoding : Each solution is represented as an array of size equal to the number of tasks and entry in position i indicates the processor allocated to task t_i. The genes in these chromosomes are in the range $[1..m]$. We are assuming an arbitrary topology for the physical connectivity of nodes in the distributed system. Figure 6.4 gives a chromosome example where a 10-task graph is mapped onto 4 processors.

Initial Population : A population is a collection valid individuals. The initial population P

Algorithm 3: Schedulability-Based NSGA-II

```
1  Initialize problem parameters;
2  Initialize genetic algorithm parameters;
3  Generate initial population with valid individuals;
4  Evaluate individuals of the initial population;
5  Non-dominated Sorting of the initial population;
6  while gen ≤ Number of Generations do
7  |  Apply Tournament Selection to create the Mating Pool;
8  |  while j ≤ Size of Intermediate Population do
9  |  |  Select two parents from the Mating Pool Perform a crossover;
10 |  |  Perform a mutation;
11 |  |  Move to Valid Solution;
12 |  |  Insert the Valid Solution to the Intermediate Population;
13 |  |  j = j + 1;
14 |  end
15 |  Recombination of the Old and Intermediate Populations;
16 |  Evaluate individuals of the New population;
17 |  Non-dominated Sorting of the New population;
18 |  gen = gen + 1;
19 end
```

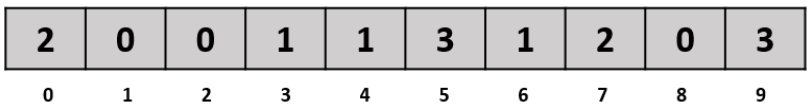


Figure 6.4 Example of chromosome Encoding (i.e. a possible mapping implementation)

consists of $|P|$ randomly generated valid solutions.

Non-dominated Sorting. NSGA-II uses the pareto-based approach in its selection process and the quality of a given solution is based on its dominance indicator. Non-dominated sorting is a process in which the solutions of a given generation are assigned to different fronts using the dominance indicator of this solution compared to all the solutions of the current population. If we consider that F_i is the front i where $rank(F_i) = i$, then all the non-dominated solutions of P are assigned to front F_1. Then, all the non-dominated solutions of $P - F_1$ will be assigned to F_2 and so on. We repeat this process until we assign all the solutions of P to the different fronts. Figure 6.5 gives an example of non-dominated sorting of a small population.

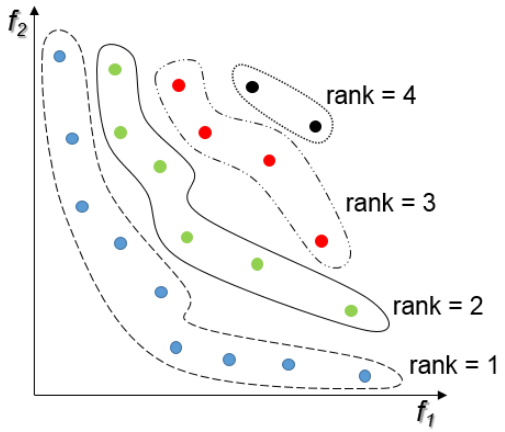


Figure 6.5 Non-dominated Ranking/Sorting of Feasible solution

Crowding Distance. Once we finish the non-dominated sorting of the current population, the crowding distance is calculated separately for the different fronts. It is defined as the distance between the two closest solutions on either side of a solution along each objective axis. This indicator provides a density estimation of solutions on each front and allows us to preserve diversity during the selection process. The crowding distance is calculated by :

$$crowdist_i = \sum_{m=1}^{M} \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}} \tag{6.8}$$

Tournament Selection. We are using a binary tournament selection operator based on the niched-comparison operator. Since this operator requires both the rank and crowding distance of each solution in the population, the comparison between two chromosomes randomly picked from the population to be part of the mating pool is carried out as follows :

1. The individual with lesser rank is selected if $rank_i \neq rank_j$
2. If the two individuals belong to the same front ($rank_i = rank_j$), the individual with

the greater crowding distance is selected.

Crossover : After selecting individuals, crossover involves crossing random couples of the mating pool to produce the individuals of the intermediate population. The child chromosome inherits the genetic material of its parents. Our operator is a single point crossover in which a point is randomly selected. The genes of the offspring are copied from one section of the first parent and the other section from the second parent, as depicted by Figure 6.6.

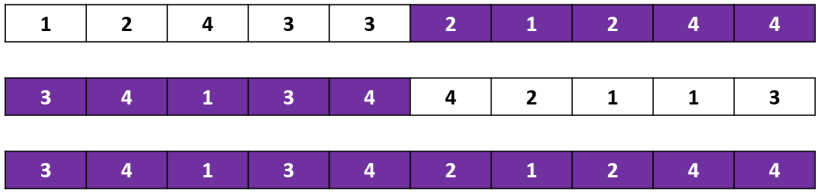


Figure 6.6 Example of single point crossover operator

Mutation. The mutation operator is applied with a much lower probability than the crossover operator. A new chromosome is created by copying a randomly picked chromosome and changing one or more of its genes, as depicted in Figure 6.7. For the Mutation, we are using a uniform operator which adds a unit rectangular distributed random value to the chosen gene.

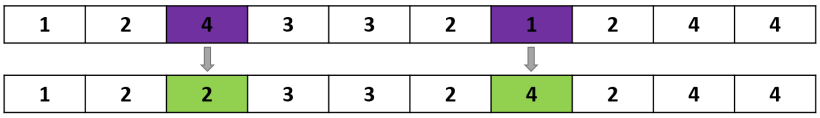


Figure 6.7 Example of uniform mutation operator

Movement to valid solutions. Since our work aims not only to optimize the predefined fitness functions but also to respect the timing constraints of the FMS using the schedulability test, the validity of the obtained solution is a key issue. Since our crossover and mutation operate randomly on the input solutions, we cannot guarantee that all the processors of the new offspring will succeed their schedulability tests. Therefore, we propose a local search algorithm allowing the movement from the obtained invalid solution to a valid one in its neighborhood by applying local changes. We are assuming that the target architecture is able able to run the overall simulation. Therefore, we are insuring the existence of valid solutions in our solution space. However, to ensure that the algorithm does not fall in an infinite loop without finding any valid solution in the neighborhood, we defined a maximum number of attempts. Thus, if after this predefined number of attempts the search algorithm

fails to find a valid configuration, the initial invalid solution is rejected. The pseudo code of the movement towards valid mapping configuration is given in algorithm 4 :

Algorithm 4: Movement to Valid Solution

```
   Input  : Invalid Mapping Configuration
   Output: Valid Mapping Configuration
 1 iter = 0;
 2 do
 3 |  while i ≤ Number of processors do
 4 |  |  Find the tasks mapped in the processor i;
 5 |  |  Test = Apply schedulability test in the processor i;
 6 |  |  if Test=fail then
 7 |  |  |  while Test=fail do
 8 |  |  |  |  Put the last task mapped in processor i in the Queue;
 9 |  |  |  |  Remove the last task from processor i;
10 |  |  |  |  Test = Apply schedulability test in the processor i;
11 |  |  |  end
12 |  |  else
13 |  |  |  while test=success do
14 |  |  |  |  Add the last element of the Queue to the processor i;
15 |  |  |  |  Test = Apply schedulability test in the processor i;
16 |  |  |  |  if test=success then
17 |  |  |  |  |  Remove the last task from the Queue;
18 |  |  |  |  else
19 |  |  |  |  |  Remove the added task from processor i;
20 |  |  |  |  end
21 |  |  |  end
22 |  |  end
23 |  |  i = i + 1;
24 |  end
25 |  i = 0;
26 |  iter = iter + 1;
27 while Queue.size ≠ 0 or iter ≤ iter_max;
28 if iter = iter_max then
29 |  Reject solution;
30 end
```

Recombination. After applying the genetic operators to the individuals of the mating pool obtained by tournament selection, the individuals of the intermediate population are combined with the current population and selection is performed to set the individuals of the next generation. If by adding all the individuals in front F_j all the individuals of the intermediate population are placed, then individuals in front F_j are selected based on their crowding

distance in the descending order until the last solution is added in the population of the next generation. At this stage, the individuals of the new population are sorted based on non-dominance and crowding distances are updated. This process is repeated until reaching the stopping criterion.

6.6 Experimental study

This section is dedicated to the computational experiments in which the performance of our framework is evaluated. For this purpose, a protocol test is developed in which the data are randomly generated. During this phase, we opted for a procedure to generate values close to that of a real FMS. First, the execution time of tasks implementing the FMS are randomly generated in the range $[0, 3]$ ms. Second, the amount of exchanged data between each pair of tasks are generated with a uniform distribution on $[0, 50]$ kB/s. Then, for the memory consumption, the values are in the interval $[0, 10]$ MB. Afterward, the periods are generated in such a way that the fastest task will run with a period equal to 16 ms. This choice is based on visualization purposes since the FMS runs 30 fps on a 60 Hz screens.

Since it is a variant of GAs, one of the main issues of NSGA-II is the parameter tuning process to improve the quality of the obtained solutions and minimize the convergence time. To calibrate these genetic discrete parameters, we used a star plan in which we start from an initial configuration of these values. Then, each parameter covers a sample of its possible values while the other others are held fixed at their latest values. When the performance metrics reach their best rates, the GA parameter under tuning is fixed to its current value and we pass to the next parameter, and so on. This operation has been processed based on a FMS modeled by 100 tasks and 10 heterogeneous machines with 4 processors each. Thence, a summary of the genetic parameters values yielded the best results on average which will be used to guide all our experiments.

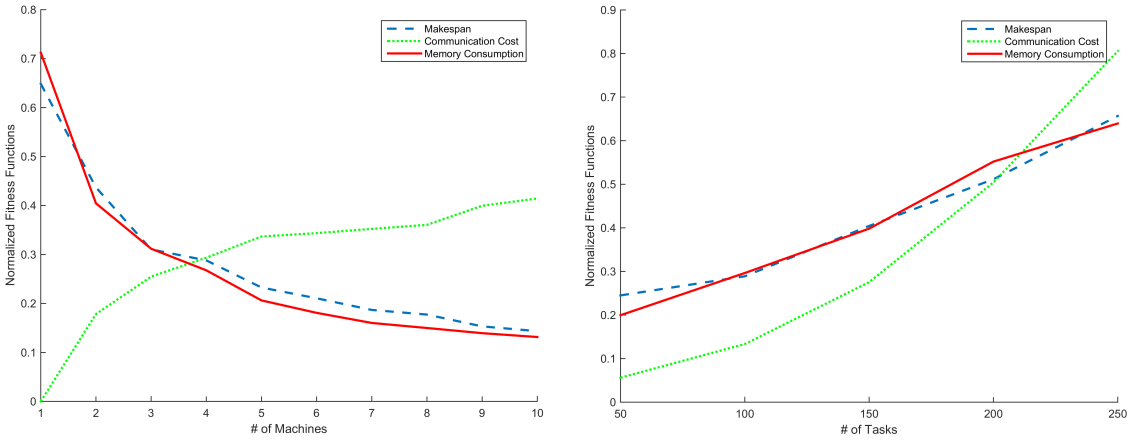
In the whole experimental process, the focus was given to measure the impact of our mapping methodology in improving the FMS performance metrics. To quantify the algorithm sensitivity to the problem parameters (Number of machines, number of tasks, connectivity degree, etc), thirty independent runs are performed for each of the test sets in order to guarantee a Gaussian approximation for the distribution of the reported averages. Then, we compute the average accuracy values. Since, the studied fitness functions have different scale, we apply feature scaling to create shifted and scaled versions of these values to facilitate the comparison and analysis tasks.

In Figure 6.8, we present the sensitivity of the different fitness functions to the number of

Table 6.2 Final Genetic Algorithm Parameters Configuration

GA Parameters	Value	GA Parameters	Value
Population Size	100	Mutation Probability	0.2
Number of Generations	500	Selection Operator	Tournament Selection
Crossover Operator	Single Point Crossover	Execution Time ranges	$[0, 3]$ ms
Crossover Probability	1.0	Communication Cost ranges	$[0, 50]$ kB/s
Mutation Operator	Uniform Mutation	Memory Consumption ranges	$[0, 10]$ MB

machines and the number of tasks implementing the FMS. The study shows that increasing the number of machines of the HDMS helps in reducing the makespan, and the memory consumption which is expected. For the communication cost, the algorithm will try to place tasks that communicate frequently on the same processor or machine to increase the data locality. By doing so, it creates a conflict with the previously mentioned fitness functions. Nonetheless, increasing the number of tasks implementing the FMS has an opposite impact on the mapping metrics by increasing them.



(a) Sensitivty to the size of the execution platform

(b) Sensitivty to the size of the real-time application

Figure 6.8 Impact of varying the size of processing elements and the size of Full-Mission Simulator

Another crucial parameter that differs from one FMS to another is the connectivity degree of the tasks implementing a given simulator. In the graph modeling the FMS, two tasks are connected if there is an edges between them. We define the connectivity of a task as the average number of tasks directly connected to it.

For this parameter, we studied different connectivity ratios to cover a wide range of simulator categories going from 5 to 50 for an FMS modeled by 100 tasks. From figure 6.9, we note that the connectivity degree has a perfect linear correlation relationship with the communication

cost fitness function with a correlation coefficient ≈ 1. On the other hand, we notice a minor impact on the other objectives. This slight impact is explained by the effect of this parameter on the made trade-offs.

In figure 6.10, we show the progression of the three fitness functions during the evolution process. We notice that as the number of generations increases, the fitness functions improve gradually while keeping a trade-off between the different metrics. Since we are using the optimal number of generations found by the tuning process, the figure does not show the steady state of the fitness functions after $Gen = 500$.

Finally, figure 6.11 shows the non-dominated solutions obtained in a single run by NSGA-II. Each point in the three-dimensional space is a valid mapping configuration near or belonging to the pareto front and respecting all the predefined timing constraints of the FMS. It's up to the decision makers to choose the most suitable one for their needs.

6.7 Conclusion and Future Work

In this paper, we propose a new framework enabling a multi-objective mapping of real-time tasks implementing the full-mission simulator on heterogeneous distributed multi-processor systems. The aim of our approach is to simultaneously minimize the makespan, the commu-

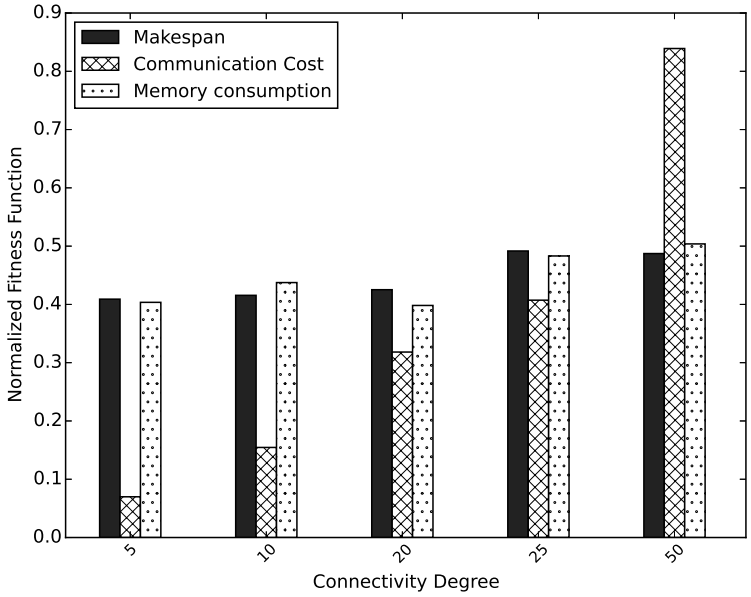


Figure 6.9 Impact of varying the Connectivity Degree on the quality of the obtained Pareto front

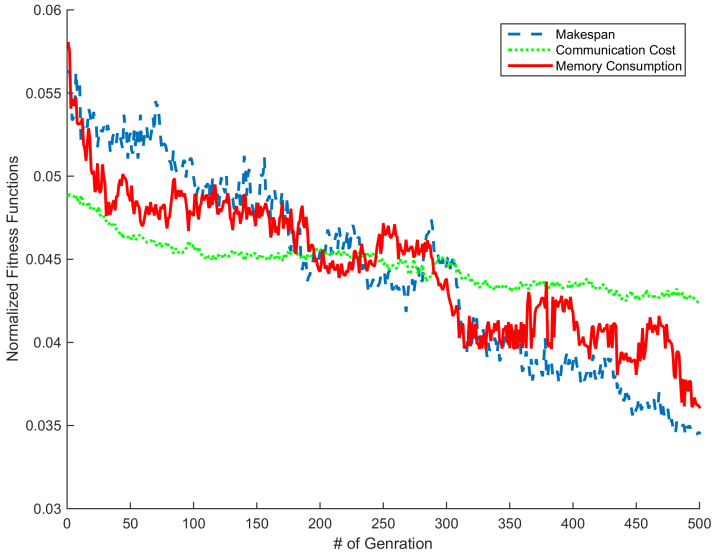


Figure 6.10 Evolution of the fitness values of the different performance metrics

nication cost and the memory consumption ensuring that the hard timing constraints are met. We showcased that exact algorithms cannot solve our problem, thus we studied the use of Multi-Objective Evolutionary Algorithms and we opted for the Non-Dominated Sorting Genetic Algorithm II in our study. This paper also paves the way for future research building upon its contributions. Our attempt in future work will be, from one side, to migrate to NSGA-III for more flexibility and scalability when integrating new objectives. From the other side, to implement a tool to select the most representative sample of non-dominated solutions to facilitate the task of the decision maker. A deep study of the quality indicators is needed for this purpose.

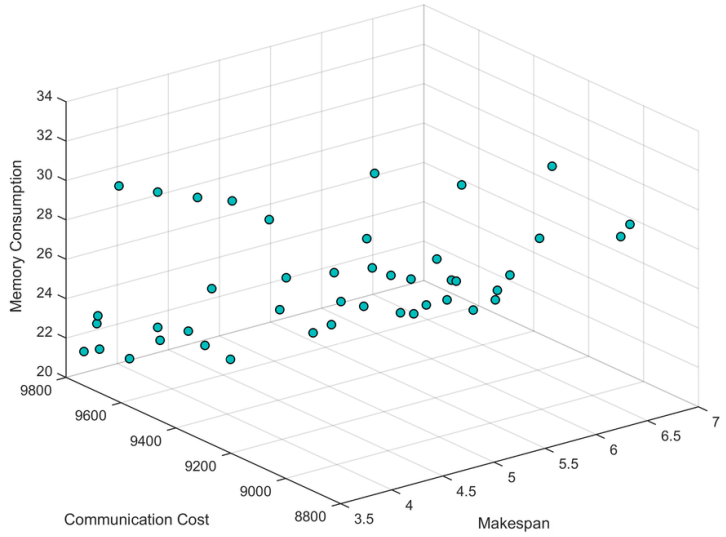


Figure 6.11 The list of the obtained non-dominated solutions (Pareto Front) at the end of the exploration process

CHAPTER 7 ARTICLE 3 : HYPAP : A HYPERVOLUME-BASED APPROACH FOR REFINING THE DESIGN OF EMBEDDED SYSTEMS

Rabeh Ayari, Mahdi Nikdast, Imane Hafnaoui, Giovanni Beltrame, Gabriela Nicolescu
Published to Embedded Systems Letters

Abstract : Designing complex embedded systems requires simultaneous optimization of multiple system performance metrics that can be addressed by applying Pareto-based multi-objective optimization techniques. At the end of this type of optimization process, designers always face Pareto Fronts including a large number of near-optimal solutions from which selecting the most proper system implementation is potentially infeasible. In this paper, for the first time, we present HypAp, a hypervolume-based automated approach to systematically help designers efficiently choose their preferred solutions after the optimization process. HypAp is a two-stage approach relying on clustering Pareto optimal solutions and then finding a subset of solutions that maximizes the hypervolume by using a genetic algorithm. The performance of HypAp is evaluated through applying HypAp to the Pareto front by the case study of mapping applications on NoC based heterogeneous MPSoC.

7.1 Introduction

New applications, such as internet-of-things (IoT), have driven the integration of a large number of functionalities into modern embedded systems. As a result, more software and hardware components need to be used, considerably enlarging the design space of modern embedded systems (Henzinger, 2006). Furthermore, the design of such systems involves the optimization of multiple competing objectives (i.e, multi-objective optimization) in which the preferred system configuration can be realized by finding the best trade-offs between these objectives (e.g., energy, throughput, etc.) usually expressed using fitness functions (Hill et al., 2008). This optimization process is referred to as the Design Space Exploration (DSE), allowing designers to find a near-optimal solution. It is worth mentioning that there is no unique optimal solution but rather a set of efficient solutions, also known as Pareto solutions. The set of all the Pareto solutions constitutes the Pareto Front (PF) (Legriel et al., 2010).

From the designers' perspective, evaluating all the near-optimal solutions (i.e., non-dominated solutions) is unrealistic. On the other hand, selecting a preferred solution from a large PF is potentially infeasible. Therefore, a possible solution is to provide a refined representation of the Pareto Front optimal solutions (i.e., a subset of solutions belonging to the PF) to

which we refer as the Reduced Pareto Front (RPF). The novel contribution of this paper is in developing HypAp, a hypervolume-based automated approach generating a RPF that is much smaller in size compared to the PF, and yet maintains the main characteristics of the PF. HypAp helps the system designers effectively to choose their preferred solutions, now from a RPF includes fewer solutions.

HypAp is a two-stage approach consisting of clustering Pareto Front solutions, and then finding a subset of solutions from each cluster that maximizes the hypervolume by using a genetic algorithm. We define several quality indicators, including hypervolume, non-uniformity, and outer-diameter, to evaluate the similarity and effectiveness of the RPF compared to the PF. As a case study, we apply HypAp to the Pareto front of a network-on-chip (NoC) mapping optimization problem.

The rest of this paper is organized as follows. Section 7.2 briefly reviews related work. In Section 7.3, we detail the proposed two-stage approach, called HypAp. Section 7.4 evaluates the effectiveness of the proposed approach and includes the case study of an NoC mapping problem and the results. Finally, Section 7.5 concludes our work.

7.2 Related Work

High-level design of embedded systems can be performed by employing different automated approaches relying on Pareto-based algorithms. Nevertheless, the major drawback of such works is that designers cannot easily choose their preferred solution(s) from the resulting large-size PF.

Designers' preferences can be ignored (Ayari et al., 2016d) or considered in different multi-objective optimization approaches before the optimization (i.e., a-priori methods), after the optimization (i.e., a-posteriori methods), or interactively during the optimization process. An a-priori approach was presented in (Ayari et al., 2016e) to guide the search in the design space towards a preferred region. In (Miettinen and Mäkelä, 2002), an a-priori method was proposed based on assigning a relative preference factor (i.e., weight) to each design objective considering the designer's preferences. An a-priori technique to optimize one objective and assign other objectives with an upper constraint was proposed in (Haimes et al., 1971). This method can alleviate the difficulties faced by the scalarization approach when solving problems with concave Pareto Fronts. Such techniques, however, require extensive knowledge of the problem in advance and cannot guarantee the Pareto-optimality of the obtained solutions, while missing some regions including promising solutions.

A-posteriori preference consideration methods are employed when the relative importance of

the objectives is unknown. Such techniques guarantee that no superior solution will be missed and the feasible implementations are available for evaluation. In (Chaudhari et al., 2010), a clustering approach was proposed to find a RPF by applying K-means method and then picking the closest solution to the centroid of each cluster as the candidate of that cluster. A graphical representation, called Level Diagrams, for n-dimensional Pareto Front analysis was proposed in (Blasco et al., 2008). By clustering the PF and then employing level diagrams to represent and analyze the RPF, (Zio et al., 2011) proposed a two-stage approach aimed at identifying a limited number of representative solutions to be presented to the designer.

While there is no unique definition of an optimal selection, all the aforementioned a-posteriori methods failed to comprehensively evaluate the quality of their RPFs in terms of different Pareto quality indicators. In this work, an effort is made to find a RPF that highly represents the PF. Particularly, we evaluate the effectiveness of the RPF in terms of several quality indicators, such as the Pareto-coverage, distribution, uniformity, and extent, and hence quantify the characteristics of the PF that have been maintained in the RPF.

7.3 HypAp : Our Proposed Approach

This section details our proposed hypervolume-based automated approach, called HypAp, developed to systematically help with the design choices obtained after the optimization process (i.e., a-posteriori approach). HypAp is a two-stage approach : Pareto optimal solutions will be first clustered and then a subset of solutions that maximizes the hypervolume will be selected. Please note that different performance metrics that need to be optimized during the DSE have a large size or great variability due to their different distributions and orders of magnitudes. Consequently, this kind of variability can lead to a knock-on effect on the clustering result. Therefore, prior to clustering, we normalize the fitness values by adjusting them to notionally averages (a.k.a. feature scaling) which leads to a better symmetry, and hence more reliable learning.

7.3.1 Clustering Pareto Front optimal solutions

The first stage of HypAp aims at clustering Pareto Front optimal solutions with respect to their similarity, facilitating the application of the genetic algorithm in the next stage. We consider using K-means, which is an unsupervised learning algorithm (Hartigan et al., 1979), to perform the clustering. K-means groups a given set of solutions into k different clusters through calculating the centroid for each cluster, and then assigning each solution to the cluster with the nearest centroid. Finding the solutions that belong to the same cluster, K-

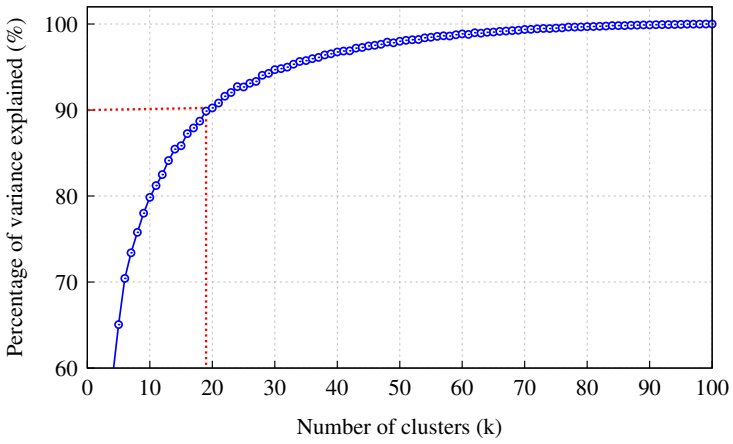


Figure 7.1 Using the Elbow method to determine the number of clusters : the percentage of variance explained versus the number of clusters.

means considers employing the euclidean distance. As a result, the sum of squared distances, d, between the centroid of each cluster, x_j, and the solutions on the PF, x_i, should be minimized :

$$d = \sum_{j=1}^{k} \sum_{i=1}^{n} (x_i - \bar{x}_j)^2, \tag{7.1}$$

in which k and n are the the number of clusters and solutions, respectively.

In any clustering technique, choosing the right number of clusters (i.e., k) is challenging. The best choice for k is often ambiguous as it highly depends on the shape and scale of the PF. In our work, we consider using the Elbow method to determine k (Tibshirani et al., 2001). Using this method, we consider the number of clusters based on the percentage of variance explained [1] (see Fig. 7.1), which is the ratio of the between-cluster variance to the total variance. As a result, based on Fig. 7.1, we choose the number of clusters k corresponding to the percentage of variance explained of 90%, after which the variance explained gain marginally drops, and hence increasing the number of clusters will no longer add significant information.

7.3.2 Hypervolume maximization

Although the resulted clusters from the first stage are informative, the number of solutions existing in each cluster can still be very large for the designer to make right choices. Furthermore, selecting solutions maintaining the information of the PF from each cluster is

1. In statistics, variance explained measures the proportion to which a mathematical model accounts for the variation of a given data set.

challenging. We address this problem by employing an optimization approach seeking ideal subsets of the PF that maximizes the hypervolume, as we discuss in the following. As a result, the problem can be seen as a variant of the maximum coverage problem where the RPF Λ', in which $|\Lambda'| = k$ and k is the number of clusters, is obtained in advance. Our aim is to maximize the diversity of the solutions belonging to Λ' and the coverage of the PF Λ.

The hypervolume indicator (a.k.a. Lebesgue measure (Coello et al., 2002)) is one of the most popular quality indicators for multi-objective optimization. It can solely capture the coverage of the solutions and their distances from the true Pareto front. Therefore, a subset with a larger hypervolume is likely to present a better set of trade-offs. Considering solutions as points in an objective space, the hypervolume is the n-dimensional space that is contained by a solution relative to a reference point defined as the nadir point (depicted in Fig. 7.2) in our approach. Nadir point is the worst-known value in each dimension.

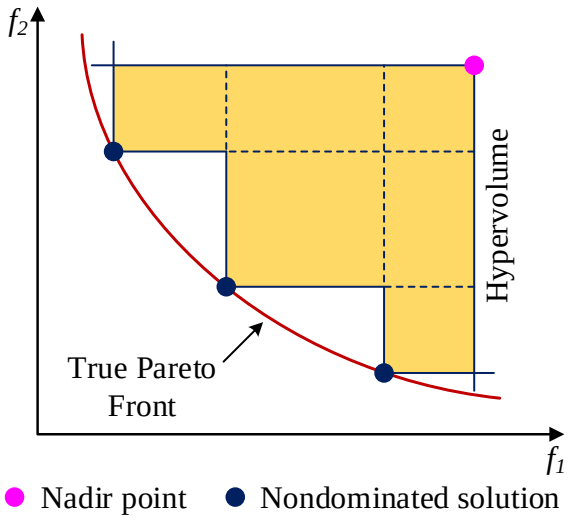


Figure 7.2 Hypervolume indicator for the non-dominated solutions with respect to the nadir point in a bi-objective optimization problem.

The hypervolume indicator for a normalized Pareto Front $\bar{\Lambda}$ is defined as (Zitzler et al., 2007) :

$$H(\bar{\Lambda}) = \int_{\mathcal{Y}} 1\left[\exists x \in X : f(x) \prec y \prec z\right] dy, \tag{7.2}$$

in which $z \in \Re^N$ is the reference point (i.e., the nadir point in our method), f is the vector obtained by stacking the objectives, and $\prec$ is the dominance operator defined as $x \prec y \Leftrightarrow (x_1 < y_1) \wedge \cdots \wedge (x_m < y_m)$.

The application of the hypervolume indicator has been greatly limited by the high computation cost of existing algorithms for the exact hypervolume computation as it is a #P-Hard

problem. We consider a methodology based on Monte-Carlo sampling (Hastings, 1970) to estimate the hypervolume, and hence improve its computation time, while sacrificing the accuracy within a tolerable limit. Since using the exact hypervolume values are not crucial in our optimization process, a Monte-Carlo estimation is performed in order to compute the percentage of random points in the performance space to be dominated by the Pareto Front. The estimation error related to the employed Monte-Carlo approach after considering 10^6 random points is within 0.02% which is highly reasonable.

Finally, due to the increasing number of possible configurations, selecting the RPF is non-trivial and considered as an NP-hard problem. Therefore, one needs to use meta-heuristics to find a good approximation of the optimal configuration. In order to find this configuration, we implement a genetic algorithm aiming on the maximization of the hypervolume. After that, HypAp successfully identifies a RPF that highly represents the PF, as we indicate in the next section.

7.4 Evaluations and Results

7.4.1 Quality indicators

Several unary quality indicators have been proposed to evaluate the quality of solution sets in terms of convergence, and diversity. In our work, convergence is ignored since we are assuming that the PF is the input of HypAp. In order to study the quality of the RPF, we opted for :

Non-Uniformity : A RPF with a lower non-uniformity is more evenly-distributed and better estimates the PF. Given a normalized Pareto Front $\bar{\Lambda}$, non-uniformity is given by(Erbas, 2007) :

$$NU(\bar{\Lambda}) = \sum_{i=1}^{|\bar{\Lambda}|-1} \frac{|d_i - \bar{d}|}{\sqrt{m}(|\bar{\Lambda}| - 1)}, \tag{7.3}$$

where d_i is the euclidean distance between two consecutive solutions, and $\bar{d}$ defines the average distance. Also, $|\bar{\Lambda}| = k$ is the number of clusters, and m is the dimension of the design space. Please note that $0 \leq NU(\bar{\Lambda}) \leq 1$, where $NU(\bar{\Lambda}) = 0$ means that the set is uniformly distributed.

Outer Diameter : The outer diameter indicator aims at computing the distance between the ideal objective vector and the nadir objective vector of a given Pareto Front using a distance metric. This unary indicator is given by (Zitzler et al., 2008) :

$$OD(\bar{\Lambda}) = \max_{1 \leq i \leq n} w_i \left((\max_{x \in \bar{\Lambda}} f_i(x)) - (\min_{x \in \bar{\Lambda}} f_i(x)) \right), \tag{7.4}$$

with weights $0 < w_i \leq 1$. If $\forall i \in [2, m], w_i = 1$, then the outer diameter becomes the maximum extent over all the dimensions of the design space. The outer diameter has a low computation cost and it is agnostic to the problem features.

7.4.2 Case study

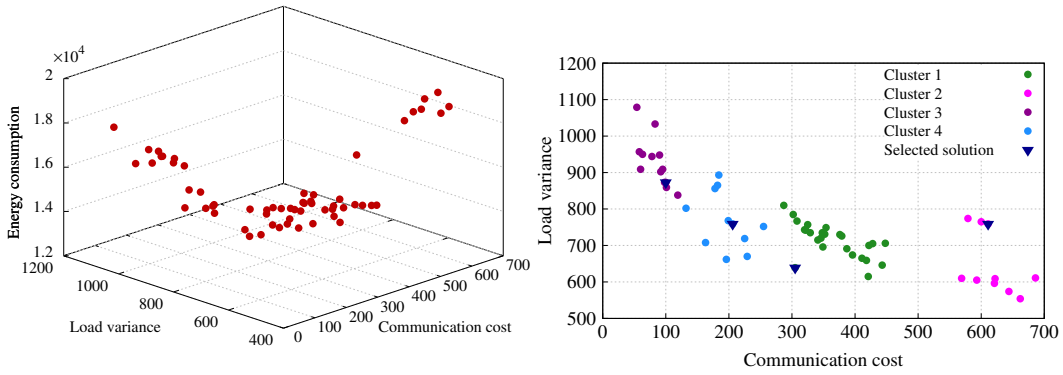
In this subsection, we present the case study of a NoC mapping problem with three objectives ; load variance, communication cost, and energy consumption (Belkacemi et al., 2016). In this problem, TGFF (Dick et al., 1998) is used to generate a set of purely-synthetic task graphs with varying number of tasks to model the application. The architecture model consists of p types of processing elements (PEs) interconnected by a Spidergon NoC topology.

We apply HypAp to the PF, depicted in Fig. 7.3(a), obtained by mapping the generated task graphs on the NoC architecture. HypAp first classifies the optimal solutions on the PF into four clusters. Solutions in each cluster are indicated using a unique color in Fig. 7.3(b). As can be seen, clusters represent different aspects of the PF with respect to the cost functions.

Table 7.1 Comparing the RPF Obtained using HypAp and K-means with the PF of the NoC case study

Quality indicator	PF	HypAp RPF	K-means RPF
Hypervolume	0.089	0.078	0.051
Non-uniformity	0.031	0.027	0.032
Outer diameter	0.364	0.243	0.261

After clustering, HypAp selects the representative solutions (i.e., RPF), indicated in Fig. 7.3(b), based on maximizing the hypervolume. Employing the quality indicators defined before, Table 7.1 compares the RPF suggested by HypAP with the PF of the problem. For better comparison, Table 7.1 also considers the RPF obtained by applying the K-means method proposed in (Chaudhari et al., 2010). As can be seen, HypAp achieves a high hypervolume that is very close to the one for the PF, while it is also higher than the hypervolume obtained using K-means. In other words, HypAp proposes a RPF including only four solutions that roughly represents the same space coverage as the PF. Moreover, higher hypervolume also means that the selected solutions in HypAP are among the closest ones to the true Pareto front. As a result, designers can easily choose from these four solutions based on their preferences. Moreover, HypAp enables the designers to refine their choices through iteratively discovering the neighborhood of the preferred solutions : each iteration can consider a new subset using a predefined radius r, representing the smallest disk containing $|\Lambda'| = k$ solutions.



(a) PF of the NoC case study

(b) 2D illustration of the PF

Figure 7.3 (a) PF obtained by mapping the generated task graphs on the NoC architecture along with its 2D illustration, (b), including the clusters and RPF.

Considering the non-uniformity results in Table 7.1, the RPF obtained using HypAp has a better uniformity compared with that from K-means and even the one from the PF itself. In other words, knowing that the PF is not uniformly distributed, HypAP suggests a RPF that indicates a higher uniformity : HypAp not only maintains the features of the PF in the RPF, but also improves them when it is possible. Moreover, considering the outer-diameter results, the first observation is that the extent (i.e., spread of the data) is better using K-means, while HypAp still provides a good extent. Nevertheless, a deeper interpretation of the results obtained using this indicator shows that HypAp avoids selecting the solutions that are in the borders of the PF (i.e., extremity of the PF). It is worth mentioning that such selections are quite poor in terms of representativity : even if they result in a good extent on the PF, they cannot provide a good-enough reference point for the refinement process explained above. Consequently, designers will be guided towards fewer directions as no more improvement can be achieved through the selections made on the extremity of the PF.

7.5 Conclusion

This paper presents HypAp, a hypervolume-based automated approach to systematically help embedded systems designers choose their preferred solutions after the optimization process. HypAp first clusters the Pareto Front solutions, and then seeks a Reduced Pareto Front that maximizes the hypervolume. The result of this preliminary work is critical for the design of complex embedded systems, in which designers require to choose from Pareto Fronts including a large number of near-optimal solutions.

CHAPTER 8 ARTICLE 4 : IMGA : AN IMPROVED GENETIC ALGORITHM FOR PARTITIONED SCHEDULING ON HETEROGENEOUS MULTI-CORE SYSTEMS

Rabeh Ayari, Imane Hafnaoui, Giovanni Beltrame, Gabriela Nicolescu
Submitted to Design Automation for Embedded Systems

Abstract : Efficient mapping of tasks onto heterogeneous multi-core systems is very challenging especially in the context of real-time applications. Assigning tasks to cores is an NP-hard problem and solving it requires the use of meta-heuristics. Relevantly, genetic algorithms have already proven to be one of the most powerful and widely used stochastic tools to solve this problem. Conventional genetic algorithms were initially defined as a general evolutionary algorithm based on blind operators with pseudo-random operations. It is commonly admitted that the use of these operators is quite poor for an efficient exploration of big problems. Likewise, since exhaustive exploration of the solution space is unrealistic, a potent option is often to guide the exploration process by hints, derived by problem structure. This guided exploration prioritizes fitter solutions to be part of next generations and avoids exploring unpromising configurations by transmitting a set of predefined criteria from parents to children. Consequently, genetic operators, such as initial population, crossover, mutation must incorporate specific domain knowledge to intelligently guide the exploration of the design space. In this paper, an improved genetic algorithm (ImGA) is proposed to enhance the conventional implementation of this evolutionary algorithm. In our experiments, we proved that ImGA leads to perceptible increase in the performance of the genetic algorithm and its convergence capabilities.

8.1 Introduction

The trend in the design of modern hardware platforms has shifted towards increasing the number of processing elements. The rise of multi-core architectures reshaped computer engineering, and long-lasting certainties had to be updated. Moore's law shifted its focus from the number of transistors to the number of cores that could be integrated on a chip and Amdahl's law (Davis and Burns, 2011a) was no longer sufficient to describe speed-ups provided by parallelization. Multi-core systems are being accepted in a wide spectrum of disciplines in industry. The use of multi-cores with real-time constraints is still a largely unexplored problem, since, in such systems, ease of integration and predictability are highly sought out.

However, due the increase in system complexity, real-time systems producers are begrudgingly accepting the use of multi-core systems due to their ever rising performance. Therefore, better real-time approaches need to be adopted to satisfy their elevated requirements. In real-time systems, the correctness of the system behavior depends not only on the results of its computations, but also on the time at which these results are produced (Burns, 1991). The schedulability analysis of a system verifies their temporal correctness under a specific scheduling policy.

In the past, research efforts has been concentrated on scheduling and schedulability analysis of single processor systems. In more recent years, several research works dealt with the problem of scheduling in multi-core systems (Tafesse et al., 2011). In uni-processor systems, the processor switches between multiple jobs (Mohammadi and Akl, 2005) whereas in multiprocessor systems, tasks run concurrently in the different processing units of the parallel execution platform. In the context of real-time scheduling on multi-core systems, one widely used approach is the partitioned scheduling. This approach is based on a two-stage approach going from a global mapping onto the architecture cores to a local scheduling in each core. For such scheduling technique, one should be able to determine : where the task should be executed (mapping) and when the task should be executed with respect to the other tasks running on the same core (uni-processor scheduling).

The complexity involved in finding efficient mapping solutions in multi-core systems has motivated the use of several meta-heuristics (Singh et al., 2013). Genetic algorithms (GAs), introduced by Goldberg and Holland (Goldberg and Holland, 1988), have already proven to be an effective tool for the implementation of partitioned scheduling (Braun et al., 2001). Conventional genetic algorithms that mimic the process of natural selection were initially defined as a general evolutionary algorithm based on blind genetic operators (selection, crossover, mutation). While they are known by their ability to explore the solution space of small problems, this benighted exploration reveals a considerable weakness with bigger problems presenting sizable solution spaces. Hence, conventional GAs need to be enhanced to solve such problems. In order to find better solutions rapidly, exploration is often aided by a guided strategy.

The endeavor of this paper is to propose an Improved Genetic Algorithm (ImGA) to guide the exploration process and extensively ameliorate the conventional GA findings. To do this, we are proposing a climbing hill repairing strategy for the population initialization. This strategy aims on adjusting the randomly generated solutions by incorporating a local search algorithm. This climbing hill technique iteratively apply local changes on the generated chromosomes to find a fitter one in its neighborhood. Later, we integrated the schedulability-guided cros-

sover operator proposed in (Ayari et al., 2016f) to be part of our improved algorithm. Then, maintaining a vast population diversity is crucial to ensure that the design space was appropriately explored and avoid premature convergence by stagnating on sub-optimal solutions. To face this problem, we propose two main contributions ; (1) a circular mutation operator switching the workload of the heaviest and lightest processors. (2), a restart technique based on injecting randomly generated solutions at advanced stages of the optimization engine giving a new impetus to the remaining generations. To demonstrate the capabilities of our proposed ImGA, we compared it against a traditional genetic algorithm. The results show that ImGA surpasses the traditional genetic algorithm by a large margin and that it was able to evolve up new fitness peaks and move the population to new and better parts of the design space.

The rest of the paper is structured as follows. We review some related work in Section 8.2. Section 8.3 describes the system model used to implement and validate our proposal. Section 8.4 presents ImGA with all the details of the enhanced genetic operators. Finally, the experimental set-up, our results and their discussion, and the conclusions are presented in Section 8.5, and Section 8.6, respectively.

8.2 Related Work

8.2.1 Scheduling in Multi-core Systems

The trend in multi-core systems is the use of different dissimilar processing units with various capabilities to build a heterogeneous parallel computing system : Hill et al. (Hill et al., 2008) claimed that asymmetric multi-core architectures offer potential speedups that are much greater than symmetric multi-core chips and never worse. Van Craeynest et al. (Van Craeynest et al., 2012b) stated that a heterogeneous architecture composed by a single big core and few smaller cores can attain higher performance and reduce energy consumption. In more recent years, several new research works dealt with the problem of scheduling in multi-core systems (Zhuravlev et al., 2012b) (Kwok and Ahmad, 1999), however the problem of scheduling in heterogeneous multi-cores carries many open questions, and the use of multi-cores with real-time constraints is still a largely unexplored problem. Davis and Burns (Davis and Burns, 2011a) provide an excellent survey of real-time scheduling algorithms for multi-core systems. They review both partitioned and global approaches, noting that the use of partitioned scheduling allows to exploit a wealth of results on schedulability and optimality for multi-processor systems. However, heterogeneous systems are explicitly excluded in this survey. Tumeo et al. (Tumeo et al., 2008), proposed an exploration approach based on ant colony

algorithm for mapping and scheduling in heterogeneous multiprocessor systems. Moreira et and al. (Moreira et al., 2007b) present a real-time scheduling algorithm for heterogeneous multi-cores mixing time-division multiplexing and static-order scheduling. This is a global static-priority algorithm that requires the solution of a linear programming optimization problem. Their solution showed 64% and 55% better results compared to Simulated Annealing and Tabu Search respectively. In (Ayari et al., 2016b), an improved implementation of NSGA-II has been proposed to schedule the execution of full-mission simulators on heterogeneous distributed multi-processor systems using a partitioned approach aiming on optimizing the makespan, memory consumption and communication cost. Qin and Jiang (Qin and Jiang, 2005) propose a heuristic-based, reliability-driven real-time scheduling (global and dynamic-priority) algorithm for heterogeneous multi-core that "reduces reliability cost by up to 71.4%. An excellent comparison among eleven of static heuristics for mapping applications onto heterogeneous distributed computing systems aiming to minimize the makespan has been conducted by Braun et al. in (Braun et al., 2001). Through this comparison, which included GAs, tabu search, simulated annealing among others, the experimental results showed that GAs consistently gave the best results for the parameters and implementation used in this exhaustive study. Several other works in literature (Erbas et al., 2010; Choi et al., 2012) have also proven that GAs can lead to better findings in terms of mapping schemes compared to other heuristics. However, none of the aforementioned works introduces an efficient guidance approach for the solution space exploration, in order to enhance the search not only of GA but also of other heuristics. Heuristics were proposed by (Mehran et al., 2008)(ter Braak et al., 2010)(Hong et al., 2009)(Brião et al., 2008)(Carvalho, 2010) to map tasks at run-time when new applications arrive and no prior knowledge of the behaviour of these applications. (Mehran et al., 2008)(ter Braak et al., 2010) propose a dynamic spiral mapping that searches the allocation of tasks in a spiral way. (Mehran et al., 2008) aimed to optimize energy by considering a homogeneous 2D-mesh topology and placing the highest communicating task at the center of the mesh. An improvement of 29% was observed in (Hong et al., 2009) when migration of tasks to other processors in a homogeneous architecture to balance workload variation was performed. (Brião et al., 2008) proposed a method that relies on Dynamic Voltage Scaling and turned off idle processors to reduce energy as well as reduce execution time. A heuristic was proposed in (Carvalho, 2010) to reduce the communication overhead on the NoC and hence decrease execution time over a heterogeneous multicore architecture.

8.2.2 Improved implementations

In evolutionary algorithms, guiding the search in the solution space is a widely studied problem tackled by introducing sophisticated operators and hybrid techniques. In our previous

work (Ayari et al., 2016f), a simple genetic algorithm based on a novel schedulability-guided operator (SGX) has proven its efficiency. Our genetic algorithm using SGX easily outperforms the classical operators by offering at least 21% improvement in terms of ratio of certainly schedulable tasks. In the current paper, SGX takes its place in ImGA to be part of the process with the aim to improve the quality of findings. The crossover operator is considered as the fundamental search operator in GAs (Zhang et al., 2007). In (Zhang et al., 2007), Zhang and al. underlined the efficiency of sophisticated operators to improve the quality of the solutions obtained by standard crossovers. In this work, authors introduced a more intelligent crossover operator using a local hill-climbing search to construct good building blocks for object classification. Ahuja et al. (Ahuja et al., 2000) proposed a greedy genetic algorithm combining the genetic algorithm with greedy approaches to solve large scale quadratic assignment problems. Also, an improved genetic implementation has been proposed in (Drezner, 2008) by including a local search algorithm to tackle the same problem. In (Shrestha and Mahmood, 2016) presented two main ideas of Mitochondrial DNA and Continent Model to improve the quality of GA. In (Toğan and Daloğlu, 2008), authors proposed a novel strategy for initial population generation. In addition, two new self-adaptive member grouping strategies were discussed. In (Ayari et al., 2016g), a simulation-based approach to assess solutions discarded by schedulability pessimism, and include them in the optimization process is proposed. Authors included a simulation stage to consider discarded solutions by schedulabity test pessimism in the exploration process. All these promising results motivated us to design an improved genetic algorithm to tackle the problem of embedded real-time application scheduling on heterogeneous multi-core systems under timing constraints.

8.3 System model

Our system model includes a taskset composed of a pool of n tasks to be executed on a heterogeneous multi-core architecture composed of m processing elements (PEs). We assume that all the tasks are preemptive and their jobs strictly periodic. Each PE can execute one task at a time, which allows the system to process m tasks concurrently. Each task is defined as a tuple $t_i =< T_i, C_i, D_i, \Pi_i >$, where : T_i is the period of the task t_i, C_i is the worst case time execution time vector of i^{th} task on all the PEs to point out system heterogeneity, D_i is the deadline of the task t_i and $\Pi_i >$ its priority. We represent the tasks worst case execution time on the different cores by establishing a $(n \times m)$ matrix, where c_{ij} corresponds to the worst case execution time of the task t_i on processor p_j. This model expresses the heterogeneity of the platform. The elements along a row indicate the execution times of a specific task on the different processing units, and those along a column indicate the expected

execution time of all tasks on a single core.

In the partitioned scheduling adopted in our work, whenever the mapping phase is processed, the taskset assigned to each PE is scheduled accordingly using a uni-processor Rate-monotonic scheduling policy. Rate-monotonic scheduling algorithm (RM) is by far the most used real-time algorithm and it is one of the easiest policies to implement. RM is a static-priority scheduling algorithm for real-time systems (Mohammadi and Akl, 2005). It is a preemptive algorithm that assigns higher priorities to the tasks with shorter periods T_i.

The Liu & Layland bound (Liu and Layland, 1973) is a schedulability test that provides a sufficient condition for a task-set to be schedulable when using an RM scheduling policy. Liu & Layland proved that a feasible schedule (i.e. that will always meet deadlines) exists if the total processor utilization U is below a specific bound. For each task-set τ_j composed of k tasks assigned to processor j, the task-set is guaranteed to be schedulable if this test yields. The necessary condition to verify the schedulability of τ_j based on the processor utilization bound, is defined as :

$$U_j = \sum_{i=1}^{k} \frac{c_{ij}}{T_i} \leq k.(2^{\frac{1}{k}} - 1) \tag{8.1}$$

8.4 ImGA for Partitioned Scheduling

In the biology-inspired GA, a population of problem solutions, termed chromosomes, is evolved over successive generations using a set of genetic operators (selection, crossover and mutation). In each generation, the fitness of every individual in the population is evaluated. The fittest individuals are stochastically selected from the current population, and each individual's genes are recombined and mutated to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been achieved, or a satisfactory fitness level has been reached for the population. However conventional genetic algorithm suffers from premature convergence and pseudo-random search. These problem occurs when the genetic operators, are no more able to generate children that are better than their parents. This is resulting from the lose of diversity in the optimization engine or the blind search in the design space. For more details and information concerning the conventional genetic algorithm and how they can be used to solve real-world problems, we refer the reader to (Davis, 1991), (Goldberg, 1989), (Banzhaf et al., 1998).

In this paper, a novel strategies for the initial population, crossover, mutation and injection are adopted in the GA process for the aforementioned intentions. Figure 8.1 presents the

ImGA flow graph as a series of steps to be performed sequentially. A detailed explanation of each ImGA's operation is presented, after, in this section.

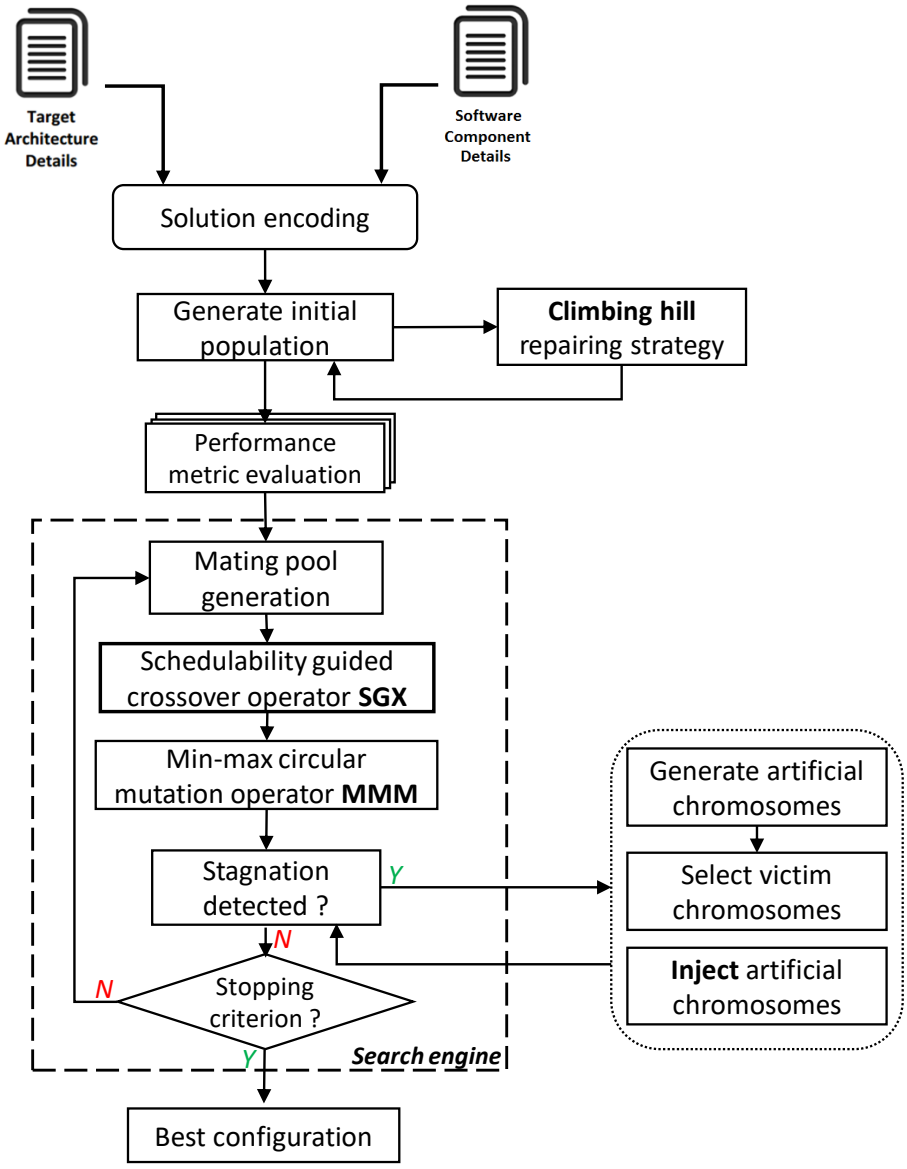


Figure 8.1 Improved Genetic Algorithm work flow

Fitness function

We know that schedulability tests, when successful, provide the guarantee that a task-set will be schedulable on a single processor. In the context of heterogeneous multiprocessors, the sub-taskset assigned to each core is the result of the mapping process. Our objective is to guide the genetic partitioning such that the schedulability of the entire taskset across the entire multi-core architecture is improved. A sufficient condition for the i^{th} task, assigned

to the j^{th} core, to be schedulable is whether or not the schedulability test of all the tasks assigned to core p_j is successful. Each solution needs to be awarded a figure of merit, to indicate how close is to this goal. Since our aim is to maximize the ratio of schedulable tasks on a given multi-core system based on the Liu & Layland schedulability test, we minimize the ratio of tasks that fail the test. The fitness function f is defined as the sum of the number of tasks assigned to those cores the sub-taskset of which violate the schedulability condition, over the size of the taskset :

$$f = \frac{1}{n} \sum_{j=1}^{m} \left(\sum_{i \ s.t. \ S_i=j} 1 \times \Gamma \left(\sum_{i \ s.t. \ S_i=j} \frac{c_{ij}}{p_i} \leq k_j.(2^{\frac{1}{k_j}} - 1) \right) \right) \tag{8.2}$$

With n, m being respectively the number of tasks and PEs and function Γ returns `0` when the condition it takes as an input is met, and `1` otherwise. A similar equation can be written for other scheduling policy , such as EDF scheduling algorithm, by replacing the schedulability test passed to Γ.

Solution encoding

In our mapping problem, we used an integer coding : each mapping solution is represented as an array of size n (the number of tasks) where each entry in position i indicates the processor on which will be mapped the task t_i. The genes in these chromosomes are in the range $[0..m-1]$.

Climbing hill repairing strategy

The conventional approach of genetic algorithm is randomly initialized population of $|P|$ design solutions. This initial population significantly influence the speed and the convergence of the optimization engine. A new initial population methodology has been proposed aiming on reducing the pseudo-random walking steps to reach the optimum design in the design space. This methodology is based on a climbing hill repairing strategy. It consists on repairing the $|P|$ arbitrary generated solutions by looking in the neighborhood while incrementally changing a single element of these solutions with the aim to find fitter design implementations. Therefore, we propose a climbing hill repairing strategy allowing the movement from the generated implementation to a better one in its neighborhood by applying local changes. This repairing strategy is inspired by the local search technique proposed in (Ayari et al., 2016h).

Mating pool generation

Tournament selection involves running several tournaments among a set of randomly chosen competitors from the current population. The winner of each tournament, which has the best

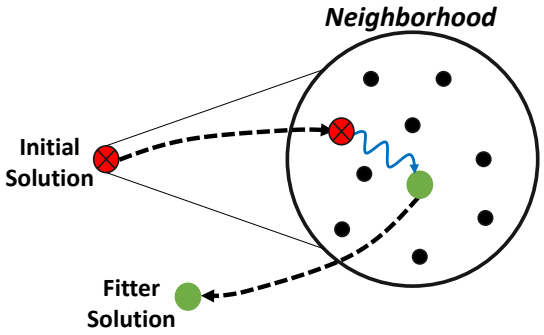


Figure 8.2 Climbing hill repairing strategy

fitness value, is selected to be part from the mating pool. Solutions belonging to the mating pool will pass iteratively through the crossover and/or mutation operators. Only, offsprings with higher quality will be part of the new population. It is worth to mention here that we adopted an elitism strategy in our recombination process. This strategy aims on copying the fittest candidates unchanged to the next generation.

Schedulability-Guided Crossover Operator (SGX)

The proposed Schedulability-Guided Crossover Operator, hereafter named SGX, consists of four steps discussed below using an illustrative example. Lower utilization rates are more likely to be transferred from parents to children. A privilege will be granted to the fitter parent.

— **Step 1 :** Two parents, $parent_1$ and $parent_2$, are randomly selected from the mating pool to be considered for the crossover process. We assume, for the rest of the procedure, that $parent_1$ has better fitness value (see section 8.4 for fitness function definition). In this paper, the choice of fitter parent was random.

Parent 1	2	0	0	1	1	3	1	2	0	0
	0	1	2	3	4	5	6	7	8	9
Parent 2	0	3	0	3	3	1	1	2	2	1
	0	1	2	3	4	5	6	7	8	9

Figure 8.3 Selected solutions $parent_1$ and $parent_2$

— **Step 2 :** In this step, each parent is represented with an Equivalent Model representation. Originally, a chromosome is encoded using an array of tasks with the allocated processor at each position ($task_{id}$). In the new model, processors are considered as containers of the tasks assigned to it. Even though they may look different, both representations illustrate the same information. By itself, the new model allows us to

easily keep track of the utilization rate of each processor after all the modifications that will be performed. In Figure 8.4, we give an example of this representation.

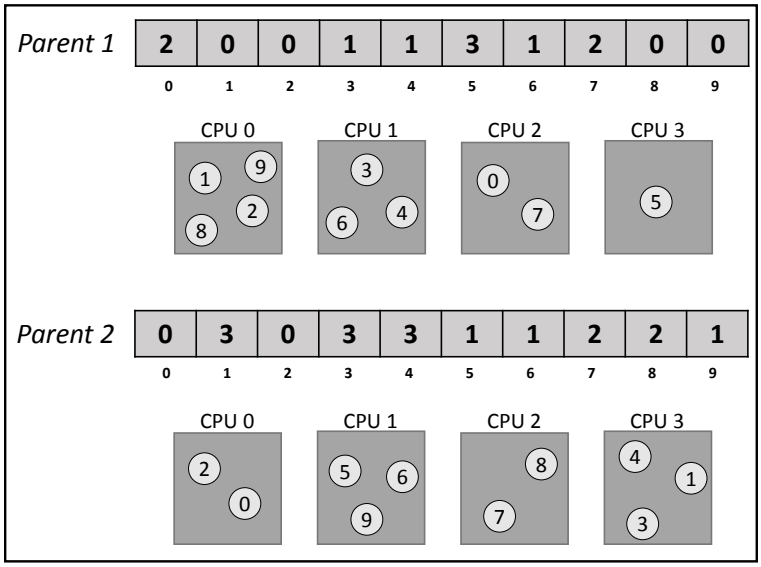


Figure 8.4 Equivalent Models of $parent_1$ and $parent_2$

— **Step 3 :** This step is at the core of our operator. It goes through a sequence of iterations to select the most fitting genes of each parent. Our process can be highlighted as follows. Since our aim is to guide the solution space exploration by schedulability, the two parents go through an evaluation stage in which a goodness value based on the aforementioned fitness function defined in equation 8.2 is assigned to each one of them. The fittest parent, $parent_1$, is scrutinized at first to select the least busy processing element $cpu_{elected}$ (i.e. that which was assigned a task-set that results in the minimum total utilization rate among the cores of the multi-core system). The tasks mapped to the elected processor are transferred immediately to $cpu_{elected}$ of the new child. Then, these tasks are removed from $parent_2$ regardless of their host processors. Subsequently, the processor $cpu_{elected}$ of $parent_2$ is excluded from the next election iteration and all its tasks are withdrawn from $parent_1$. Finally, after all these modifications, the processors utilization rates are updated. This procedure is iterated for the requested number of processors by alternating the parents using a round-robin principle. This round-robin approach between parents is conducted to avoid the algorithm from premature convergence. The workings of this step are detailed as follows :

- As mentioned before, we assumed that $parent_1$ has a better fitness value. Hence, the first election iteration will be carried out on the $parent_1$. Among the four processing

units, the fourth processor cpu_3 is elected since it has the lowest utilization rate. All the tasks in this processor (t_5) are copied in the fourth processor cpu_3 of the offspring.
- Before starting the second election round which will be performed on $parent_2$ and based on the round-robin strategy, we exclude the previously selected processor cpu_3 from the running tournament. Then, we erase all the tasks that have been copied in the latest sub-step (t_5) and those belonging to the excluded processor (t_1, t_3, t_4) from the two parents. Afterwards, cpu_1 is elected among the remaining processors and the sub-taskset containing t_6 and t_9 is copied in the same core cpu_1 of the new child.
- Now, cpu_3 and cpu_1 are forbidden from the election tour performed on $parent_1$ and all the tasks mapped on them in the two parents are withdrawn (t_9). Then, cpu_2 is elected and tasks t_0 and t_7 are transmitted to the offspring.
- During this last stage, cpu_0 is added to the list of prohibited processors (cpu_3 and cpu_1). Then, we remove the tasks belonging to cpu_2 of the first parent (t_0) from the second parent after discarding cpu_0 containing t_7 and t_8. Afterwards, the remaining resident tasks in cpu_0 of $parent_2$ are copied in its counterpart of the offspring.

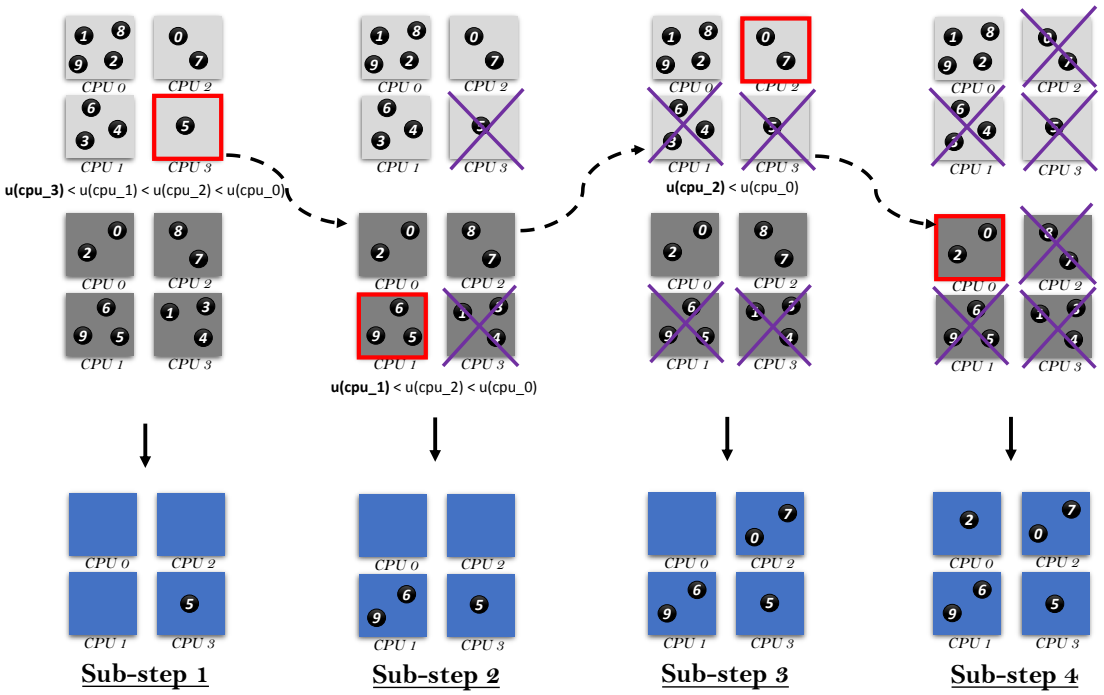


Figure 8.5 Work flow of third step of the proposed Schedulability-Guided Crossover Operator using a 10-task application and a quad-core architecture

— **Step 4 :** At this stage, some tasks are left non-mapped onto the new offspring. These remaining genes are obtained by applying a biased recombination between the two

parents. For each individual, we associate a dominance ratio δ_i defined as follows.

$$\delta_1 = \frac{f_1}{f_1 + f_2} \quad and \quad \delta_2 = \frac{f_2}{f_1 + f_2} \tag{8.3}$$

The parent's gene that has a higher dominance value contributes more to the new offspring. If both dominance values are equal then crossover becomes uniform. As depicted in red in Figure 8.6, the remaining non-mapped genes are obtained by applying a biased crossover by assuming that both of parents have the same weight.

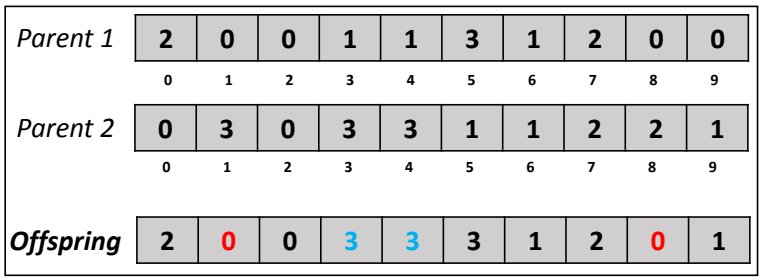


Figure 8.6 Biased crossover applied for the remaining non-mapped tasks

Min-Max Circular Mutation Operator

Mutation mechanism is a substantial operator aiming on maintaining the vast diversity of the population and assist in overcoming local minima/maxima in the design space exploration. In ImGA, we propose a novel min-max circular mutation operator. The operation corresponds to a rearrangement of the solution encoding entries by moving the final entry to the first position, while shifting all other entries to the next position. After applying this circular permutation, we look for the heaviest and lightest processors in terms of number of assigned tasks. Then we swap these tasks by mapping the tasks assigned to the heaviest processor to the lighter one and vice versa.

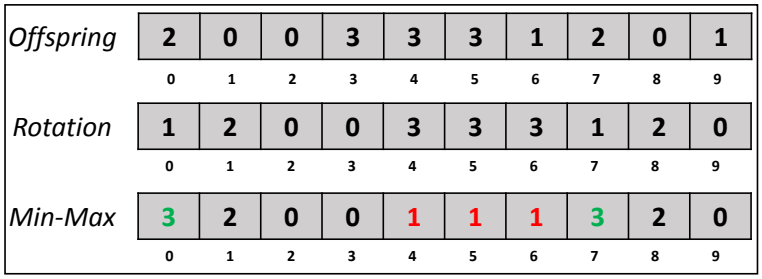


Figure 8.7 Min-Max Circular Mutation Operator

Restart using injection strategy

One of the key improvements enhancing the quality of the traditional GA in our ImGA, is that it has a supplementary powerful diversity mechanism in addition to the mutation operator. This mechanism is based on an injection strategy. It consists on injecting a user-defined rando number of artificial chromosomes (*i.e.* mapping solutions. Such technique will smoothly help the optimization engine to escape sub-optimal (local) solutions. This injection strategy is provoked every time a potential case of premature convergence is detected. This detection is based on the computation of the hamming distance between successive best solutions belonging to the different generations. If the best solution remain the same after a set of generations, than the injection is launched. The search engine will gain the capacity to surpass its stagnation and hopefully will lead the process to jump out the local traps.

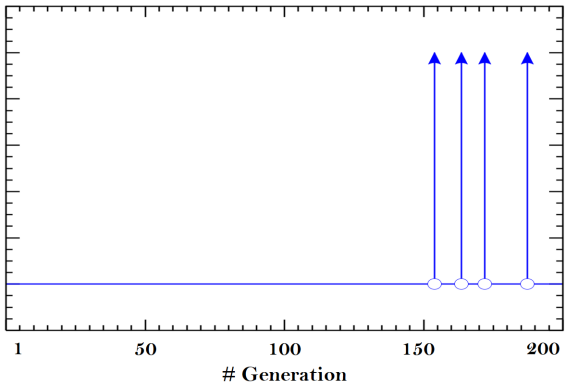


Figure 8.8 An exploration process involving four injections of new solutions after different stagnation phases

8.5 Results and Discussion

We carried out a performance-based comparison of our operator with fundamentals crossover operators used in literature. In GAs, one of the persisting big challenges is the parameter tuning process to improve the quality of the obtained solutions and minimize the convergence time (Angelova and Pencheva, 2011). In general, researchers acknowledge that a good parameter configuration is substantial for significant performance improvements. Due to the large number of options in the tuning of the genetic parameters, a great deal of time and effort is required to calibrate them. Such tuning allows studying the effect of each parameter on the obtained results, as well as the effects of interactions between these parameters. The tuning process will not be discussed in depth in this paper, only the results of this procedure are given. The values of the genetic parameters in Table 8.1 yielded the best results on average and are used for all the conducted experiments.

Table 8.1 Final Conventional Genetic Algorithm Parameters Configuration

Genetic Parameters	Value
Population Size	100
Mating Pool Size	50
Number of Generations	100
Crossover Probability	1.0
Mutation Probability	0.2
Crossover Operator	Single Point Operator
Mutation Operator	Gaussian Operator
Survival Selection	Tournament Selection of size 2

All the experiments discussed here were carried out by randomly generating real-time application made by 100 tasks for a quad-core fully heterogeneous architecture. The experiments were performed thirty times in order to be able to carry out the assumption of a Gaussian approximation for the distribution of the reported averages listed in this section.

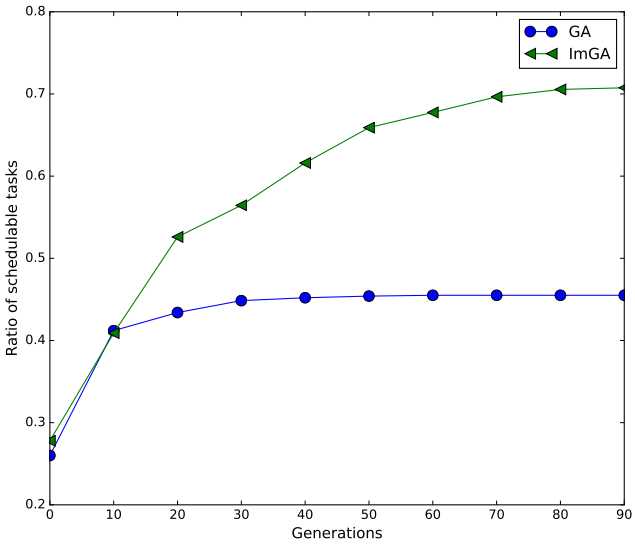


Figure 8.9 Evolution of the ImGA compared to the conventional GA

Figure 8.9 reports the ratio of schedulable tasks, found by ImGA during its evolution process for Liu & Layland schedulability test against the traditional GA described above. It can be noticed that ImGA clearly and greatly surpasses the conventional GA. Our ImGA was able to reach fitter solutions after only 10 generations before converging to higher quality of near-

optimal solution. Therefore, ImGA to find out the way to better solutions with successive generations starting from the first generations. This figure reports also that ImGA maintains a diverse population during its evolution improving GA's exploratory power and avoiding premature convergence. Thus, ImGA is able to refine better solutions from good ones.

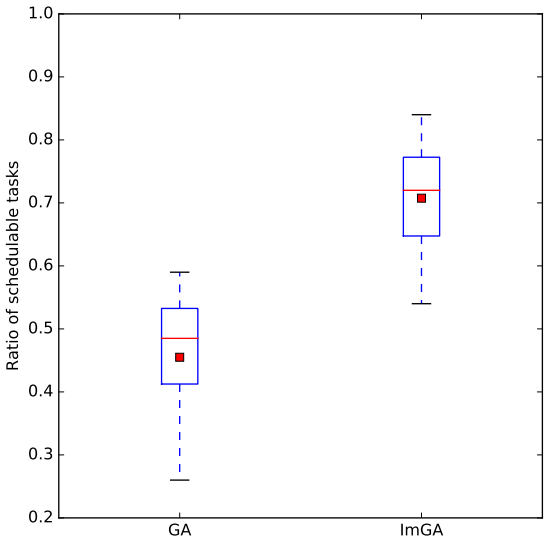


Figure 8.10 Comparison of the partitioning quality of the ImGA and the conventional GA.

Figure 8.10 provides a revealing summary of the conducted experiments. In this figure, we present a box plot graph allowing us to portray outliers and study the spread of the obtained results. In this figure, we show that ImGA achieves better findings compared to the conventional genetic algorithm. This is further supported by observing that the upper bound of the classical GA barely reaches the lower quantile of the ImGA.

After ensuring the increase in performance gained by the use of ImGA, we evaluate how each of these operations (population initialization, crossover, mutation and injection) affects the quality of the partitioning. Results are presented in Figures 8.11. We note that, as we expected, the ratio of schedulable tasks is always higher under ImGA than classical GA. Moreover, we can observe from these results that the repairing strategy added to the initial population and the crossover operator are the parameters having the strongest positive impact on performance. On the other hand, injection strategy and small mutation operator have a lower effect on it. This impact can be explained by the fact that the aim of these operations is mainly diversity maintaining than convergence.

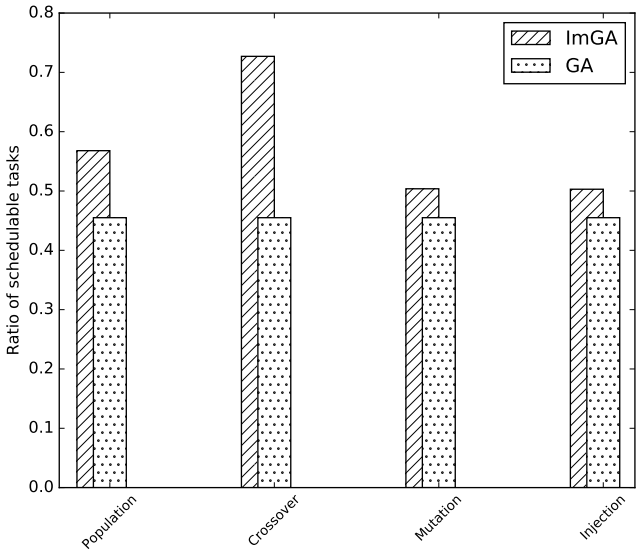


Figure 8.11 Comparison of the partitioning quality of the genetic algorithm using the different crossover operators.

8.6 Conclusions

Methods for increasing the genetic algorithms accuracy are of great importance. In this paper, we proposed an improved genetic algorithm that we call it ImGA. ImGA incorporates a guided search strategy based on a novel genetic operators involving the problem structure. We start by a local search repairing strategy to enhance the quality of the solutions belonging to the initial population. Than, we used our schedulability-driven crossover operator to intelligently explore the design technique. Later, two main techniques were proposed to avoid premature convergence and increase the population diversity ; mutation operator and random injection strategy. With the help of the proposed improvements, ImGA is enhanced and becomes capable of finding near-optimal solutions for the mapping problem on heterogeneous multi-core systems. We defined the quality of a partitioning for real-time tasksets as the fraction of tasks with undecided schedulability and we used this metric to compare the partitioning produced by ImGA against the classical GA. The experiments and the results presented in the paper strongly reveal the potential capability of the proposed method act as a powerful tool for the solution space exploration in the context of partitioned scheduling to implementation particularities of the algorithm.

CHAPTER 9 GENERAL DISCUSSION

The new trend in the processing of the produced data of IoT real-time embedded devices has shifted towards doing more computation near the source of data, in the edge of the network. This need to perform on-device computation helps to reduce latency for critical and confidenetial data of real-time applications and better processing of the huge amounts of data being generated by these devices.

Succeeding this transition towards the new paradigm called *edge computing* requires at the same time ; designing powerful real-time embedded devices, optimizing the mapping and scheduling topology real-time embedded systems on multi-processor target platforms and light mining techniques enabling smarter functioning of these devices.

The data produced by IoT devices contain valuable information about user preferences and needs. However, this information is hidden behind billions of events. Therefore, efforts are needed to dynamically extract the requested information. Moreover, the huge size and dynamic nature of the streaming data make the task of learning harder.

In the previous chapters, we presented a set of optimization and data mining techniques and tools to enhance the design and usage of these real-time devices. These methods are structured in three parts to align with the three major objectives presented in the Introduction :

— Part 1 : Perform online mining of the produced data in order to extract the valuable hidden information.
— Part 2 : Conceive better real-time embedded devices.
— Part 3 : Help designers in the task of decision making while choosing the most prferred implementation.

The proposed methods can be bivided in two main categories :

1. Online data mining for better functioning of real-time embedded systems
   — The definition of an effective fully online data stream clustering algorithm called DeDaSC. The proposed algorithm is based on incremental Delaunay triangulation DeDaSC. This is an incremental, one-pass density-based algorithm, which is able to construct arbitrary shaped in real time with considerable optimization of time and space. DeDaSC has intuitive advantages over traditional techniques which try to cluster the whole dataset at one time rather than processing the data stream evolving over time. In this context, DeDaSC has been applied to enhance IoT embedded devices functioning. However, the proposed algorithm, due to its capa-

bilities, may be used at various stages of embedded systems life cycle from design to aging detection.

2. Optimization to design more powerful real-time embedded systems
   — The proposition of a fully automated approach enabling a multi-objective mapping of real-time tasks implementing the full-mission simulator on heterogeneous distributed multi-processor systems using Non-Dominated Sorting Genetic Algorithm II. The aim of the proposed approach is to simultaneously minimize the makespan, the communication cost and the memory consumption ensuring that the hard timing constraints are met. Once the list of near-otimal solutions (*Pareto front*) is obtained, designers face the problem of choosing the appropriate solution to prototyping from this large non-dominated solutions.
   — At this point, the challenge that has been raised at this stage is how to efficiently find a representative set of Pareto optimal solutions, quantify the trade-offs in satisfying the different objectives, and finding a single solution that satisfies the subjective preferences of the designer. In this context, in Chapter 6, we presented a hypervolume-based approach to choose a representative subset of the obtained Pareto front (*HypAp*). This reduced subset facilitates designer's task by easily picking the best solution based on designer preferences.
   — The definition and implementation of an improved genetic algorithm (*ImGA*) to guide the search engine during the design space exploration. ImGA incorporates a guided search strategy based on a novel genetic operators involving the problem structure (in this context system's schedulability). We defined the quality of a partitioning for real-time tasksets as the fraction of tasks with decided schedulability. We used this metric to strongly reveal the potential capability of the partitioning produced by ImGA against the classical GA.

CHAPTER 10 CONCLUSION AND PERSPECTIVES

This thesis is motivated by the the raising challenges concerned with the design and functioning of real-time embedded systems that will be placed on the edge of the network. In this last part of this document, we present the summary of the thesis and directions for future research.

10.1 Final remarks

Nowadays, it may be evidence that more and more researchers believe that edge computing will be the next big trend after cloud computing in the context of IoT networks. In this dissertation, we zoomed in on the problem of design and usage of real-time applications on multi-processor based devices. Accurately, our aim was to leverage the time-to-market of complex real-time systems, in relation with Internet of Things, and more specifically to serve the edge computing paradigm. The panoply of tools developed within this research will help IoT community to succeed the processing transition from the heart to the edge of the network.

In this context, the scientific contributions behind this thesis can be summarized as follows :

1. The definition and implementation of a new fully online data stream clustering algorithm.
2. The implementation of a fully automated design space exploration process based on multi-objective evolutionary algorithm to dispatch the execution of real-time applications on heterogeneous multi-processor systems.
3. The proposition of a hypervolume-based approach to help designers choose the preferred implementation from the obtained Pareto front.
4. The definition of a schedulability guided exploration engine based on an improved evolutionary algorithm.

10.2 Thesis perspectives

The results presented in the different papers of this dissertation point out to several interesting directions for future work. This section discusses potential ways forward providing possible improvements. Several research lines are open by the mining and optimization methods proposed in the previous chapters.

1. The analysis of the efficiency of the proposed online clustering algorithm needs to be addressed with other real-life datasets. These real-data are not only limited to the IoT but also to other domains.
2. Expanding the multi-objective optimization algorithm in order to incorporate new metrics. Expanded implementation may assess other performance metrics, such as power consumption.
3. The automation of the guided design space exploration process using agnostic operators would be a further improvement of our ImGA.
4. The hypervolume based approach can be improved by applying a dimensionality reduction process such us principal component analysis prior to clustering the Pareto front. Introducing this reduction allow the generalization of HypAp to Pareto fronts with higher dimensions.
5. While we claim that the online data stream clustering algorithm can hugely impact the search engine if adequately included in the exploration of the design space, more experiments are needed to solidify our hypotheses.

LIST OF REFERENCES

M. Aazam, I. Khan, A. A. Alsaffar, et E.-N. Huh, "Cloud of things : Integrating internet of things and cloud computing and the issues involved", dans *Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on*. IEEE, 2014, pp. 414–419.

C. C. Aggarwal, *Data streams : models and algorithms*. Springer Science & Business Media, 2007, vol. 31.

C. C. Aggarwal et C. K. Reddy, *Data clustering : algorithms and applications*. CRC press, 2013.

R. K. Ahuja, J. B. Orlin, et A. Tiwari, "A greedy genetic algorithm for the quadratic assignment problem", *Computers & Operations Research*, vol. 27, no. 10, pp. 917–934, 2000.

A. Amini, T. Y. Wah, et H. Saboohi, "On density-based data streams clustering algorithms : a survey", *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 116–141, 2014.

M. Angelova et T. Pencheva, "Tuning genetic algorithm parameters to improve convergence time", *International Journal of Chemical Engineering*, vol. 2011, 2011. DOI : 10.1155/2011/646917

M. Ankerst, M. M. Breunig, H.-P. Kriegel, et J. Sander, "Optics : ordering points to identify the clustering structure", vol. 28, no. 2. ACM, 1999, pp. 49–60.

G. Ascia, V. Catania, et M. Palesi, "Multi-objective mapping for mesh-based NoC architectures", *Proceedings of the 2nd IEEE/ACM/IFIP ...*, 2004.

R. Ayari, I. Hafnaoui, A. Aguiar, P. Gilbert, M. Galibois, J.-P. Rousseau, G. Beltrame, et G. Nicolescu, "Multi-objective mapping of full-mission simulators on heterogeneous distributed multi-processor systems", *The Journal of Defense Modeling and Simulation : Applications, Methodology, Technology*, pp. 1–12, July 2016.

——, "Multi-objective mapping of full-mission simulators on heterogeneous distributed multi-processor systems", *The Journal of Defense Modeling and Simulation : Applications, Methodology, Technology*, p. 1548512916657907, 2016.

——, "Multi-objective mapping of full-mission simulators on heterogeneous distributed multi-processor systems", *The Journal of Defense Modeling and Simulation*, p.

1548512916657907, 2016.

R. Ayari, I. Hafnaoui, G. Beltrame, et G. Nicolescu, “Schedulability-guided crossover operator for real-time scheduling on heterogeneous multi-core systems”, dans *Proceedings - IEEE International Symposium on Rapid System Prototyping*, 2016.

——, “Schedulability-guided crossover operator for real-time scheduling on heterogeneous multi-core systems”, dans *Proceedings of the IEEE International Symposium on Rapid System Prototyping, RSP*, 2016.

——, “Simulation-based schedulability assessment for real-time systems”, dans *Proceedings of the Conference on Summer Computer Simulation*. Society for Computer Simulation International, 2016.

——, “Simulation-based schedulability assessment for real-time systems”, dans *Proceedings of the Summer Computer Simulation Conference*. Society for Computer Simulation International, 2016, p. 30.

——, “Simulation-based schedulability assessment for real-time systems”, dans *Proceedings of the Summer Computer Simulation Conference*. Society for Computer Simulation International, 2016, p. 30.

——, “Dedasc : Delaunay triangulation-based data stream clustering algorithm for internet of things”, *IEEE IoT Journal*, 2017, under review.

——, “Imga : An improved genetic algorithm for partitioned scheduling on heterogeneous multi-core systems.” *Design Automation for Embedded Systems*, 2017.

R. Ayari, M. Nikdast, I. Hafnaoui, G. Beltrame, et G. Nicolescu, “Hypap : a hypervolume-based approach for refining the design of embedded systems”, *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 57–60, 2017.

B. Babcock, S. Babu, M. Datar, R. Motwani, et J. Widom, “Models and issues in data stream systems”, dans *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2002, pp. 1–16.

T. P. Baker, “A comparison of global and partitioned edf schedulability tests for multiprocessors”, dans *In International Conf. on Real-Time and Network Systems*. Citeseer, 2005.

W. Banzhaf, P. Nordin, R. E. Keller, et F. D. Francone, *Genetic programming : an introduction*. Morgan Kaufmann San Francisco, 1998, vol. 1.

A. Bastoni, B. B. Brandenburg, et J. H. Anderson, "An empirical comparison of global, partitioned, and clustered multiprocessor edf schedulers", dans *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*. IEEE, 2010, pp. 14–24.

D. Belkacemi, Y. Bouchebaba, M. Daoui, et M. Lalam, "Network on chip and parallel computing in embedded systems", dans *Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2016 IEEE 10th International Symposium on*. IEEE, 2016, pp. 146–152.

C. Bergamini, L. Oliveira, a.L. Koerich, et R. Sabourin, "Combining different biometric traits with one-class classification", *Signal Processing*, pp. 2117–2127, 2009. DOI : 10.1016/j.sigpro.2009.04.043

J. A. C. Bernabe Dorronsoro, Pascal Bouvry, "Multi-objective robust static mapping of independent tasks on grids", dans *IEEE World Congress on Computational Intelligence*, 2010.

E. Bini, G. Buttazzo, et G. Buttazzo, "Rate monotonic analysis : the hyperbolic bound", *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, Jul 2003. DOI : 10.1109/TC.2003.1214341

X. Blasco *et al.*, "A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization", *Information Sciences*, vol. 178, no. 20, pp. 3908–3924, 2008.

F. Bonomi, R. Milito, J. Zhu, et S. Addepalli, "Fog computing and its role in the internet of things", dans *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", *Journal of Parallel and Distributed computing*, vol. 61, no. 6, pp. 810–837, 2001.

——, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", *Journal of Parallel and Distributed computing*, vol. 61, no. 6, pp. 810–837, 2001.

T. Braun et al, “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems”, *Journal on Parallel and Distributed Computing*, vol. 61, June 2001.

E. Brião, D. Barcelos, et F. Wagner, “Dynamic task allocation strategies in MPSoC for soft real-time applications”, *DATE*, 2008.

A. Burns, “Scheduling hard real-time systems : a review”, *Software Engineering Journal*, May 1991.

E. d. S. Carvalho, “Dynamic task mapping for MPSoCs”, *... of Computers, IEEE*, 2010.

P. Chaudhari *et al.*, “Computing the most significant solution from Pareto front obtained in multi-objective evolutionary”, *International Journal of Advanced Computer Science and Applications*, 2010.

M. Cheramy, “Etude et evaluation des politiques d’ordonnancement temps reel multiprocesseur”, Thèse de doctorat, University of Toulouse, 2014.

P. Chitra et al, “Evolutionary algorithmic approaches for solving three objectives task scheduling problem on heterogeneous systems”, dans *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, February 2010, pp. 38–43.

J. Choi, H. Oh, S. Kim, et S. Ha, “Executing synchronous dataflow graphs on a SPM-based multicore architecture”, *Proceedings of the 49th Annual Design ...*, 2012. En ligne : `http://dl.acm.org/citation.cfm?id=2228480`

C. A. C. Coello *et al.*, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2002, vol. 242.

E. G. Coffman, Jr., M. R. Garey, et D. S. Johnson, “Approximation algorithms for np-hard problems”, D. S. Hochbaum, éd. Boston, MA, USA : PWS Publishing Co., 1997, ch. Approximation Algorithms for Bin Packing : A Survey, pp. 46–93.

E. G. Coffman, Jr, M. R. Garey, et D. S. Johnson, “An application of bin-packing to multiprocessor scheduling”, *SIAM Journal on Computing*, vol. 7, no. 1, pp. 1–17, 1978.

L. Davis, “Handbook of genetic algorithms”, 1991.

R. I. Davis et A. Burns, “A survey of hard real-time scheduling for multiprocessor systems”, *ACM computing surveys (CSUR)*, vol. 43, no. 4, p. 35, 2011.

——, “A survey of hard real-time scheduling for multiprocessor systems”, *ACM Comput. Surv.*, vol. 43, no. 4, pp. 35 :1–35 :44, Oct. 2011. DOI : 10.1145/1978802.1978814

M. De Berg, O. Cheong, M. Van Kreveld, et M. Overmars, *Computational Geometry : Introduction*. Springer, 2008.

J. A. De Loera, J. Rambau, et F. Santos, *Triangulations Structures for algorithms and applications*. Springer, 2010.

K. Deb et S. Agrawal, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : NSGA-II”, *Lecture notes in . . .*, 2000.

K. Deb, D. Joshi, et A. Anand, “Real-coded evolutionary algorithms with parent-centric recombination”, *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 61–66, May 2002. DOI : 10.1109/CEC.2002.1006210

K. Deb et al, “A fast and elitist multiobjective genetic algorithm : NSGA-II”, *IEEE Transactions on Evolutionary Computation,*, vol. 6, no. 2, pp. 182–197, April 2002.

M. Devi et A. Anju, “Multiprocessor scheduling of dependent tasks to minimize makespan and reliability cost using nsga-ii”, *International Journal in Foundations of Computer Science & Technology*, 2014.

R. Dick *et al.*, “Tgff : task graphs for free”, dans *Workshop on Hardware/Software Codesign*, 1998.

Z. Drezner, “Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem”, *Computers & Operations Research*, vol. 35, no. 3, pp. 717–736, 2008.

M. Ehrgott, “Vilfredo pareto and multi-objective optimization”, dans *Optimization stories : 21st International Symposium on Mathematical Programming, Berlin*, August 2012, pp. 447–453. DOI : 10.1109/REAL.1989.63567

C. Erbas, S. Cerav-Erbas, et A. D. Pimentel, “Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system”, *Journal of Parallel and Distributed Computing*, pp. 758–766, July 2010. DOI : 10.1016/j.jpdc.2010.03.011

C. Erbas, *System-level Modelling and Design Space Exploration for Multiprocessor Embedded System-on-chip Architectures*. Amsterdam University Press, 2007, vol. 132.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” dans *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, et X. Xu, “Incremental clustering for mining in a data warehousing environment”, dans *VLDB*, vol. 98. Citeseer, 1998, pp. 323–333.

C. F. et al., *Scheduling in Real-Time Systems*. Wiley, 2002.

U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et R. Uthurusamy, *Advances in knowledge discovery and data mining*. AAAI press Menlo Park, 1996, vol. 21.

J. Fenn et M. Raskino, “Gartner’s hype cycle special report for 2011”, *Stamford, CT : Gartner*, 2011.

S. Funk et S. Baruah, “Task assignment on uniform heterogeneous multiprocessors”, dans *Real-Time Systems, 2005. (ECRTS 2005). Proceedings. 17th Euromicro Conference on*, 2005, pp. 219–226. DOI : 10.1109/ECRTS.2005.31

L. Golab et M. T. Özsu, “Issues in data stream management”, *ACM Sigmod Record*, vol. 32, no. 2, pp. 5–14, 2003.

D. E. Goldberg, “Genetic algorithms in search, optimization, and machine learning”, *Reading : Addison-Wesley*, 1989.

D. E. Goldberg et J. H. Holland, “Genetic algorithms and machine learning”, *Machine Learning*, vol. 3, pp. 95–99, Oct 1988.

P. J. Green et R. Sibson, “Computing dirichlet tessellations in the plane”, *The Computer Journal*, vol. 21, no. 2, pp. 168–173, 1978.

S. Guha, R. Rastogi, et K. Shim, “Cure : an efficient clustering algorithm for large databases”, dans *ACM Sigmod Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

S. Guha, A. Meyerson, N. Mishra, R. Motwani, et L. OCallaghan, “Clustering data streams : Theory and practice”, *IEEE transactions on knowledge and data engineering*, vol. 15, no. 3, pp. 515–528, 2003.

L. Guibas et J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of voronoi”, *ACM transactions on graphics (TOG)*, vol. 4, no. 2, pp. 74–123, 1985.

L. J. Guibas, D. E. Knuth, et M. Sharir, “Randomized incremental construction of delaunay and voronoi diagrams”, *Algorithmica*, vol. 7, no. 1, pp. 381–413, 1992.

V. Guliashki, H. Toshev, et C. Korsemov, “Survey of evolutionary algorithms used in multiobjective optimization”, *Problems of Engineering Cybernetics and Robotics*, vol. 60, pp. 42–54, 2009.

X. Guo, E. Ipek, et T. Soyata, “Resistive computation : avoiding the power wall with low-leakage, stt-mram based computing”, dans *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 371–382.

I. Hafnaoui, R. Ayari, G. Nicolescu, et G. Beltrame, “Simulation-based model generator for software performance estimation”, dans *Proceedings of the Conference on Summer Computer Simulation*. Society for Computer Simulation International, 2016.

——, “An analysis of random cache effects on real-time multi-core scheduling algorithms”, dans *Proceedings of the IEEE International Symposium on Rapid System Prototyping, RSP*, 2017.

——, “Scheduling real-time systems with cyclic dependence using data criticality”, *Design Automation for Embedded Systems*, vol. 21, no. 2, pp. 117–136, June 2017.

Y. Y. Haimes *et al.*, “On a bicriterion formulation of problems of integrated system identification and system optimization”, *IEEE Transactions on Systems Man and Cybernetics*, 1971.

W. A. Halang et A. D. Stoyenko, *Real Time Computing*. Springer-Verlag New York, Inc., 1994.

J. A. Hartigan *et al.*, “Algorithm AS 136 : A k-means clustering algorithm”, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

W. K. Hastings, “Monte carlo sampling methods using Markov chains and their applications”, *Biometrika*, vol. 57, pp. 97–109, 1970.

T. Henzinger, “The embedded systems design challenge”, dans *International Symposium on Formal Methods*, 2006, pp. 1–15.

M. Hill *et al.*, “Amdahl’s law in the multicore era”, *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

A. Hinneburg, D. A. Keim *et al.*, “An efficient approach to clustering in large multimedia databases with noise”, vol. 98, 1998, pp. 58–65.

S. Hong, S. H. K. Narayanan, M. Kandemir, et O. Ozturk, “Process variation aware thread mapping for chip multiprocessors”, dans *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 821–826.

E. S. H. Hou, N. Ansari, et H. Ren, “A genetic algorithm for multiprocessor scheduling”, pp. 113–120, 1994.

J. Hu et R. Marculescu, “Energy-and performance-aware mapping for regular NoC architectures”, *Aided Design of Integrated Circuits*, 2005.

C. V. N. Index, “Forecast and methodology, 2014-2019 white paper”, *Technical Report, Cisco, Tech. Rep.*, 2015.

W. Jakob, M. Gorges-Schleuter, et C. Blume, “Application of genetic algorithms to task planning and learning”, dans *Parallel Problem Solving from Nature 2, PPSN-II,*, Brussels, Belgium, 1992, pp. 291–300.

Z. Jia et A. Pimentel, “Nasa : A generic infrastructure for system-level mp-soc design space exploration”, *Systems for Real*, 2010.

T. D. Jick, “Mixing qualitative and quantitative methods : Triangulation in action”, *Administrative science quarterly*, vol. 24, no. 4, pp. 602–611, 1979.

D. Johnson, A. Demers, J. Ullman, M. Garey, et R. Graham, “Worst-case performance bounds for simple one-dimensional packing algorithms”, *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974. DOI : 10.1137/0203025

D. S. Johnson, “Near-optimal bin packing algorithms”, Thèse de doctorat, Massachusetts Institute of Technology, 1973.

Q. Kang, H. He, et H. Song, “Task assignment in heterogeneous computing systems using an effective iterated greedy algorithm”, *Journal on System Software*, vol. 84, pp. 985–992, June 2011.

Y. Kang et D. Zhang, “A hybrid genetic scheduling algorithm to heterogeneous distributed system”, *Journal on Applied Mathematics*, vol. 3, pp. 750–754, 2012.

G. Karypis, E.-H. Han, et V. Kumar, “Chameleon : Hierarchical clustering using dynamic modeling”, *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

H. Kennedy, “Simulation reshaping military training”, Septembre 1999.

A. Khokhar et al, “Heterogeneous computing : challenges and opportunities”, *Computer*, vol. 26, pp. 18–27, June 1993.

V. G. Khoroshevsky et M. G. Kurnosov, “Mapping Parallel Programs Into Hierarchical Distributed Computer Systems”, pp. 123–128, 2009.

V. Kianzad et S. Bhattacharyya, “Efficient techniques for clustering and scheduling onto embedded multiprocessors”, *IEEE Trans. Parallel Distrib. Syst.*, vol. Volume 17, no. 7, pp. 667–680, 2006.

M. Klein, T. Ralya, B. Pollak, R. Obenza, et M. G. Harbour, *A practitioner’s handbook for real-time analysis : guide to rate monotonic analysis for real-time systems*. Springer Science & Business Media, 2012.

D. Koufaty, D. Reddy, et S. Hahn, “Bias scheduling in heterogeneous multi-core architectures”, dans *Proceedings of the 5th European conference on Computer systems*, série EuroSys ’10, 2010, pp. 125–138.

C. M. Krishna, *Real-Time Systems*. Wiley Online Library, 1999.

Y.-K. Kwok et I. Ahmad, “Static scheduling algorithms for allocating directed task graphs to multiprocessors”, *ACM Computing Surveys (CSUR)*, vol. 31, no. 4, pp. 406–471, 1999.

J. Legriel *et al.*, “Approximating the Pareto front of multi-criteria optimization problems”, dans *TACAS*, 2010, pp. 69–83.

J. Lehoczky, L. Sha, et Y. Ding, “The rate monotonic scheduling algorithm : exact characterization and average case behavior”, dans *Real Time Systems Symposium, 1989., Proceedings.*, 1989, pp. 166–171. DOI : 10.1109/REAL.1989.63567

L. Lin et C. Wang, “Communication-driven task binding for multiprocessor with latency insensitive network-on-chip”, *. . . , 2005. Proceedings of . . .*, 2005.

D. Lischinski, “Incremental delaunay triangulation”, *Graphics gems*, pp. 47–59, 1994.

C. Liu et J. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment”, *Journal of the ACM*, vol. 20, pp. 46–61, Jan. 1973.

Y. K. H. Lu et J. He, “A pso-based genetic algorithm for scheduling of tasks in a heterogeneous distributed system”, *JOURNAL OF SOFTWARE*, vol. 8, no. 6, pp. 1443–1444, 2013.

J. Macqueen, “Some methods for classification and analysis of multivariate observations”, dans *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

D. K. Manhart, “Will edge computing devour the cloud ?” `https://www.wespeakiot.com/will-edge-computing-devour-cloud/`, 2017, [Online ; accessed 30-Janvier-2018].

C. Marcon, “Comparison of network-on-chip mapping algorithms targeting low energy consumption”, *Computers & Digital*, 2008.

J. Mattai et M. Joseph, *Real-Time Systems : specification, verification, and analysis*. Prentice Hall PTR, 1995.

A. Mehran, A. Khademzadeh, et S. Saeidi, “DSM : A Heuristic Dynamic Spiral Mapping algorithm for network on chip”, *IEICE Electronics Express*, 2008.

K. Miettinen et M. M. Mäkelä, “On scalarizing functions in multiobjective optimization”, *OR spectrum*, vol. 24, no. 2, pp. 193–213, 2002.

M. R. Miryani et M. Naghibzadeh, “Hard real-time multiobjective scheduling in heterogeneous systems using genetic algorithms”, dans *Proceedings of the 14th International CSI Computer Conference (CSICC’09)*, 2009.

M. Miryani et M. Naghibzadeh, “Hard real-time multiobjective scheduling in heterogeneous systems using genetic algorithms”, *Computer Conference, 2009. ...*, 2009.

A. Mohammadi et S. G. Akl, “Scheduling algorithms for real-time systems”, *School of Computing Queens University, Tech. Rep*, 2005.

O. Moreira, F. Valente, et M. Bekooij, “Scheduling multiple independent hard-real-time jobs on a heterogeneous multiprocessor”, *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, pp. 57–66, Sep 2007. DOI : `10.1145/1978802.1978814`

——, “Scheduling multiple independent hard-real-time jobs on a heterogeneous multiprocessor”, dans *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, série EMSOFT ’07, 2007, pp. 57–66.

F. Murtagh, “A survey of recent advances in hierarchical clustering algorithms”, *The Computer Journal*, vol. 26, no. 4, pp. 354–359, 1983.

S. Muthukrishnan *et al.*, “Data streams : Algorithms and applications”, *Foundations and Trends® in Theoretical Computer Science*, vol. 1, no. 2, pp. 117–236, 2005.

S. Navaz et U. Ansari, “An evolutionary algorithm in grid scheduling by multi-objective optimization using variants of nsga”, *International Journal of Scientific and Research Publications*, vol. 2, September 2012.

R. T. Ng et J. Han, “Efficient and effective clustering methods for spatial data mining”, dans *Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994, pp. 144–155.

M. Niemeier, A. Wiese, et S. Baruah, “Partitioned real-time scheduling on heterogeneous shared-memory multiprocessors”, dans *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*, 2011, pp. 115–124. DOI : 10.1109/ECRTS.2011.19

H. Orsila, T. Kangas, et E. Salminen, “Automated memory-aware application distribution for multi-processor system-on-chips”, *Journal of Systems*, 2007.

M. D. P. Serafini et S. Dehuri, “Simulated annealing for multiple objective optimization optimization problem”, dans *Advances in Computing and Communications*, vol. 193, 2011, pp. 113–125.

R. Piscitelli et A. D. Pimentel, “Interleaving Methods for Hybrid System-level MPSoC Design Space Exploration”, *Embedded Computer Systems (SAMOS), 2012 International Conference on*, pp. 7–14, 2012. DOI : 10.1109/SAMOS.2012.6404152

X. Qin et H. Jiang, “A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters”, *Journal of Parallel and Distributed Computing*, vol. 65, no. 8, pp. 885–900, 2005.

D. Rajeswari, V. Kumar, et R. Kanaga, “Efficient scheduling using multi-objective genetic algorithm for independent task”, *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, pp. 272–277, March 2014.

G. Raravi et V. Nelis, “A ptas for assigning sporadic tasks on two-type heterogeneous multiprocessors”, dans *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, 2012, pp. 117–126. DOI : 10.1109/RTSS.2012.64

G. Raravi, B. Andersson, K. Bletsas, et V. Nelis, “Outstanding paper award : Task assignment algorithms for two-type heterogeneous multiprocessors”, dans *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, 2012, pp. 34–43. DOI : 10.1109/ECRTS.2012.21

S. Samur et S. Bulkan, “An evolutionary solution to a multi-objective scheduling problem”, dans *Proceedings of the World Congress on Engineering*, 2010.

S. Saranya et al, “Scheduling independent tasks on heterogeneous distributed computing systems using multiobjective optimization approach on multicore processors”, dans *Advances in Computing, Control, Telecommunication Technologies, 2009. ACT ’09. International Conference on*, 2009, pp. 481–483.

P. Serafini, “Simulated annealing for multiple objective optimization optimization problem”, dans *Tenth International Conference on Multiple Criteria Decision Making*, Taipei, 1992, pp. 87–96.

L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, et A. K. Mok, “Real time scheduling theory : A historical perspective”, *Real-time systems*, vol. 28, no. 2-3, pp. 101–155, 2004.

G. Sheikholeslami, S. Chatterjee, et A. Zhang, “Wavecluster : A multi-resolution clustering approach for very large spatial databases”, dans *VLDB*, vol. 98, 1998, pp. 428–439.

W. Shi, J. Cao, Q. Zhang, Y. Li, et L. Xu, “Edge computing : Vision and challenges”, *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

A. Shrestha et A. Mahmood, “Improving genetic algorithm with fine-tuned crossover and scaled architecture”, *Journal of Mathematics*, vol. 2016, 2016.

H. Siegel, I. Ahmad, et Y. Kwok, “A semi-static approach to mapping dynamic iterative task onto heterogeneous computing systems”, 2006.

G. Sih et E. Lee, “A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures”, *Parallel and Distributed Systems, IEEE Transactions on*, vol. 4, no. 2, pp. 175–187, 1993. DOI : 10.1109/71.207593

J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. de Carvalho, et J. Gama, “Data stream clustering : A survey”, *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 13, 2013.

A. Singh, A. Kumar, et T. Srikanthan, “A hybrid strategy for mapping multiple throughput-constrained applications on MPSoCs”, *Proceedings of the 14th international . . .*, 2011.

A. K. Singh, M. Shafique, A. Kumar, et J. Henkel, “Mapping on multi/many-core systems : survey of current and emerging trends”, dans *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 1.

N. Srinivas et K. Deb, “Muiltiobjective optimization using nondominated sorting in genetic algorithms”, *Evolutionary Computation, IEEE Transactions on*, vol. 2, pp. 221–248, 1994.

J. A. Stankovic et K. Ramamritham, *Hard real-time systems*. [sn], 1988.

J. A. Stankovic, M. Spuri, K. Ramamritham, et G. C. Buttazzo, *Deadline scheduling for real-time systems : EDF and related algorithms*. Springer Science & Business Media, 2012, vol. 460.

L. P. Suresh, S. S. Dash, et B. K. Panigrahi, “Artificial intelligence and evolutionary algorithms in engineering systems”, *Proceedings of ICAEES*, vol. 1, 2014.

G. Syswerda et J. Palmucci, “The application of genetic algorithms to resource scheduling”, dans *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp. 502–508.

B. Tafesse, A. Raina, J. Suseela, et V. Muthukumar, “Efficient scheduling algorithms for mpsoc systems”, dans *Information Technology : New Generations (ITNG), 2011 Eighth International Conference on*. IEEE, 2011, pp. 683–688.

T. D. ter Braak, P. K. Hölzenspies, J. Kuper, J. L. Hurink, et G. J. Smit, “Run-time spatial resource management for real-time applications on heterogeneous mpsocs”, dans *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2010, pp. 357–362.

R. Tibshirani *et al.*, “Estimating the number of clusters in a data set via the gap statistic”, *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

V. Toğan et A. T. Daloğlu, “An improved genetic algorithm with initial population strategy and self-adaptive member grouping”, *Computers & Structures*, vol. 86, no. 11, pp. 1204–1218, 2008.

A. Tumeo, C. Pilato, F. Ferrandi, D. Sciuto, et P. Lanzi, “Ant colony optimization for mapping and scheduling in heterogeneous multiprocessor systems”, dans *Embedded Computer Systems : Architectures, Modeling, and Simulation, 2008. SAMOS 2008. International Conference on*, 2008, pp. 142–149. DOI : 10.1109/ICSAMOS.2008.4664857

K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, et J. Emer, “Scheduling heterogeneous multi-cores through performance impact estimation (pie)”, dans *ACM Computer Architecture News*, vol. 40, no. 3. IEEE Computer Society, 2012, pp. 213–224.

——, “Scheduling heterogeneous multi-cores through performance impact estimation (pie)”, *SIGARCH Comput. Archit. News*, vol. 40, no. 3, 2012. DOI : 10.1145/2366231.2337184

W. Wang, J. Yang, R. Muntz *et al.*, “Sting : A statistical information grid approach to spatial data mining”, dans *VLDB*, vol. 97, 1997, pp. 186–195.

D. H. Wolpert et W. G. Macready, “No free lunch theorems for optimization”, *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 1, p. 67–82, 1997.

N. Xu, B. Cui, L. Chen, Z. Huang, et Y. Shao, “Streaming Graph Partitioning”, vol. 27, no. 6, pp. 1560–1572, 2015.

X. Yang et M. Gen, “Evolution program for bicriteria transportation problem”, dans *16th International Conference on computers and industrial Engeeniring*, Ashikaga, Japan, 1994, pp. 451–454.

C. Ykman-Couvreur, “Linking run-time resource management of embedded multi-core platforms with automated design-time exploration”, *Computers & Digital ...*, 2011.

D. L. Yuanlong Chen et P. Ma, “Implementation of multi-objective evolutionary algorithm for task scheduling in heterogeneous distributed systems”, *Journal of Software*, vol. 7, pp. 1367–1374, June 2012.

M. Zhang, X. Gao, et W. Lou, “A new crossover operator in genetic programming for object classification”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 5, pp. 1332–1343, 2007.

T. Zhang, R. Ramakrishnan, et M. Livny, “Birch : an efficient data clustering method for very large databases”, dans *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

T. ZHOU et H. XIONG, “Design of Energy-efficient Hierarchical Scheduling for Integrated Modular Avionics Systems”, *Chinese Journal of Aeronautics*, vol. 25, no. 1, pp. 109–114,

2012.

S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, et M. Prieto, “Survey of scheduling techniques for addressing shared resources in multicore processors”, *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 4, 2012.

——, “Survey of scheduling techniques for addressing shared resources in multicore processors”, *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 4, 2012.

E. Zio *et al.*, “A clustering procedure for reducing the number of representative solutions in the Pareto front of multiobjective optimization problems”, *European Journal of Operational Research*, 2011.

E. Zitzler et L. Thiele, “Multiobjective evolutionary algorithms : a comparative case study and the strength Pareto approach”, *Evolutionary Computation, IEEE Transactions on*, vol. 3, pp. 257–271, November 1999.

E. Zitzler *et al.*, “The hypervolume indicator revisited : On the design of pareto-compliant indicators via weighted integration”, dans *International Conference on Evolutionary Multi-Criterion Optimization*, 2007.

——, “Quality assessment of Pareto set approximations”, dans *Multiobjective Optimization*. Springer, 2008, pp. 373–404.