Visualization of heuristic-based multi-objective design space exploration of embedded systems

Taghavi Razavi Zadeh, T.

Publication date
2012

Document Version
Final published version

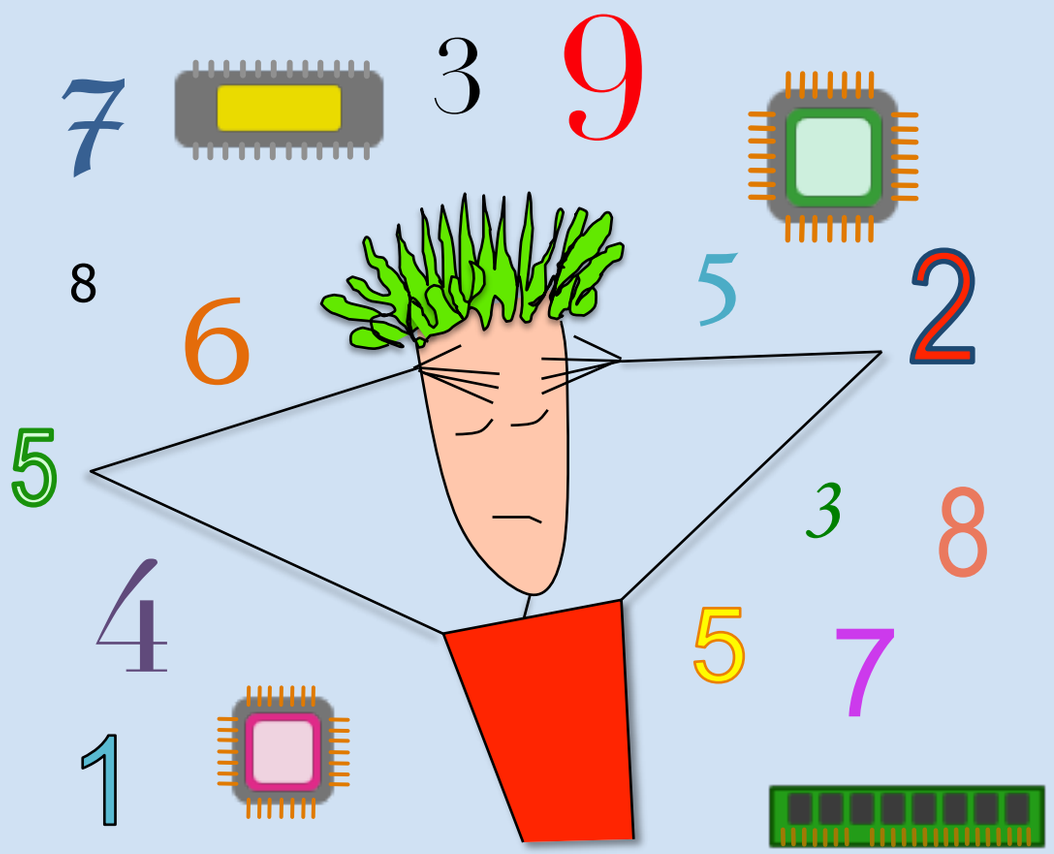
UNIVERSITY OF AMSTERDAM

ISBN 978-90-90265-51-3

Visualization of Heuristic-Based Multi-Objective Design Space Exploration of Embedded Systems

Toktam Taghavi Razavi Zadeh

Visualization of Heuristic-Based Multi-Objective Design Space Exploration of Embedded Systems



Toktam Taghavi Razavi Zadeh

Toktam Taghavi Razavi Zadeh

Visualization of Heuristic-Based Multi-Objective Design Space Exploration of Embedded Systems

Visualization of Heuristic-Based Multi-Objective Design Space Exploration of Embedded Systems

---

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus,
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor
promoties ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op woensdag 18 januari 2012 om 10:00 uur

door

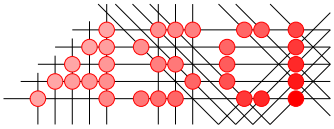
Toktam Taghavi Razavi Zadeh

geboren te Mashhad, Iran

Promotiecommissie:

Promotor:	
Prof. dr. C. R. Jesshope	Universiteit van Amsterdam, NL
Co-promotor:	
Dr. A. D. Pimentel	Universiteit van Amsterdam, NL
Overige Leden:	
Prof. dr. ir. Cees T. A. M. de Laat	Universiteit van Amsterdam, NL
Prof. dr. ir. Jarke J. van Wijk	Technische Universiteit Eindhoven, NL
Prof. Smail Niar	University of Valenciennes and Hainaut-Cambresis, FR
Dr. Todor P. Stefanov	Universiteit Leiden, NL

Faculteit der Natuurwetenschappen, Wiskunde en Informatica



Netherlands Organisation for Scientific Research

The work described in this thesis has been carried out in the ASCI graduate school and was financially supported by the Netherlands Organization for Scientific Research NWO.

ISBN 978-90-90265-51-3
ASCI dissertation series number 248

Printed in The Netherlands

To my parents

Visualization of Heuristic-Based Multi-Objective Design Space Exploration of Embedded Systems

Toktam Taghavi Razavi Zadeh

Abstract

The design of modern embedded systems is quite complex. Typically, multiple and conflicting criteria should be optimized simultaneously such as performance, power, cost, weight, etc. To support such a wide spectrum of non-functional demands, modern embedded systems often have a heterogeneous system architecture. They consist of components that range from fully programmable processor cores for providing flexibility to fully dedicated hardware components for time-critical application tasks. The complexity of embedded systems forces designers to model and simulate systems and their components to explore the wide range of design choices. Such Design Space Exploration (DSE) is especially needed during the early design stages, where the design space is at its largest and where a wrong design decision can make the difference between the success or failure of the final product. DSE methodologies targeting early design stages are applied at a high level of abstraction, thereby minimizing the modeling effort and optimizing the simulation speed.

Due to the exponential design space in real problems, evaluating and comparing every single point in the design space is infeasible. Therefore, heuristic search techniques are often used to search the design space for optimum design points using only a finite number of design-point evaluations. As the searched design space still is vast, interpreting and drawing the right conclusions from such copious simulation results and understanding how the design space was searched by heuristic searching algorithms can be extremely cumbersome. Such analysis is, however, essential to the designer as it provides insight into the landscape of the design space.

In this thesis, we address the problem of interpreting and analyzing the large amount of data generated by system-level computer architecture simulators during the exploring alternative architectural solutions. To do that, we embrace data visualization as an aid for analysis and interpretation. We introduce the structural use of visualization techniques in the design of embedded systems, and specifically for supporting the design space exploration of embedded systems. We have incorporated all the developed visualization techniques into a tool, called VMODEX. Actually, we build a bridge between two previously separate research fields: data visualization and embedded system design.

In this thesis, for performing a comprehensive study of the DSE process, we define three separate stages, which are discussed in detail throughout the three chapters 3,

4 and 5. For each stage, we have developed several methods and visualization techniques to enable users a rapid and more accurate analysis. The first stage is developing an optimization method that efficiently and effectively explores the design space. The algorithm developers can use VMODEX to perform a comprehensive study on the performance of different multi-objective optimization algorithms, from different points of view, in order to find the best one for their specific problem. At the second stage, the most efficient optimization algorithm is used for exploring the design space. In VMODEX, we have developed a variety of capabilities that help designers to analyze and interpret the DSE results from different perspectives and at multiple levels of abstraction, in order to gain insight into the landscape of the design space. At the last stage, the most preferred design point should be chosen for the final implementation. Various decision making approaches are provided in VMODEX that help decision makers to better understand the trade-offs between different criteria and guide them to make better decisions.

Acknowledgements

Over the last four years that I have been working towards my PhD degree, I have benefited from the help, sympathy, and encouragement offered by many people. Without their support and guidance, I would not be able to make my accomplishments come true. I would like to take the opportunity here to express my appreciation.

First and foremost, I would like to thank my co-promotor, Dr. Andy Pimentel for giving me the opportunity to do my PhD studies in the CSA group at the University of Amsterdam. He gave me the full freedom to pursue my own direction of research and ideas, provided me an excellent working environment and helped me a lot to improve my writing skills. I would also like to thank my promotor Prof. Chris Jesshope for his interest and support to our research. I would like to extend my gratitude to my PhD examination committee, especially to Prof. Jarke J. van Wijk, whose detailed comments helped me to improve my thesis.

I am thankful to my former advisors in Ferdowsi University of Mashhad, Dr. Mohsen Kahani, Prof. Mahmoud Naghibzadeh, Dr. Hossein Deldari, and Dr. Mohammad Hossein Yaghmaee. I learned a lot from them during my master and bachelor studies.

I am very grateful to all the members of computer systems architecture group. A very special thanks goes to Roberta Piscitelli, Peter van Stralen, Mark Thompson and Simon Polstra for their help and support. I would also like to express my sincere gratitude to Erik Hitipeuw for his assistance in resolving various bureaucratic issues. I am also thankful to Narges Zarrabi and Narges Javaheri for being my best friends and supportive whenever I needed them. I also greatly appreciate the help, co-operation and friendship of all my Iranian friends in the Netherlands.

I am highly indebted to my parents for their unconditional love and support. They provided me with the best possible education that I could think of. This would not have been possible without their prayers, blessings and sacrifices. I am also thankful to my sisters Maryam and Marjaneh for their kindness and support.

Last but not least, I am extremely grateful to my dear husband Mojtaba. He is always my main source of inspiration and motivation. He has definitely been my best counselor and a wonderful caring person throughout the years.

Toktam Taghavi Razavi Zadeh

Amsterdam, The Netherlands, January 2012

Table of contents

List of Tables

List of Figures

1 Introduction

1.1 Design Space Exploration in Embedded Systems Design

The design of modern embedded systems is quite complex. On one hand, these systems often target mass production and battery-based devices, and thus they should be low cost, small in size, light weight and be power efficient. On the other hand, they need to execute a wide range of functionalities, and therefore need to achieve high (real-time) performance and flexibility. To support such a wide spectrum of non-functional demands, modern embedded systems often have a heterogeneous system architecture. They consist of components that range from fully programmable processor cores to fully dedicated hardware components for time-critical application tasks. Increasingly, such heterogeneous system components reside together on a single chip, yielding heterogeneous Multi-Processor System-on-Chip (MPSoC) architectures that exploit task-level parallelism in applications. These MPSoC systems have now become the keystones in the development of today's modern embedded systems, such as smart phones, digital televisions, game consoles and navigation systems.

The complexity of these systems forces designers to simulate systems and their components in the very early design stages. This is often referred to as "system-level design". The system-level models are applied at a high level of abstraction, thereby minimizing the modeling effort and optimizing the simulation speed that is needed for targeting the early design stages. Design Space Exploration (DSE) assesses alternative architectural solutions and plays a crucial role in system-level architecture design. It can have a large impact on the success or failure of the final product. Especially, exploration of different design choices during the early design stages, where the design space is still at its largest, is of eminent importance. Wrong decisions early in the design can be extremely costly in terms of re-design effort, or even deadly to the product's success.

The heterogeneity of these embedded systems and at the same time their conflicting design criteria greatly complicate system design. A large number of alternative

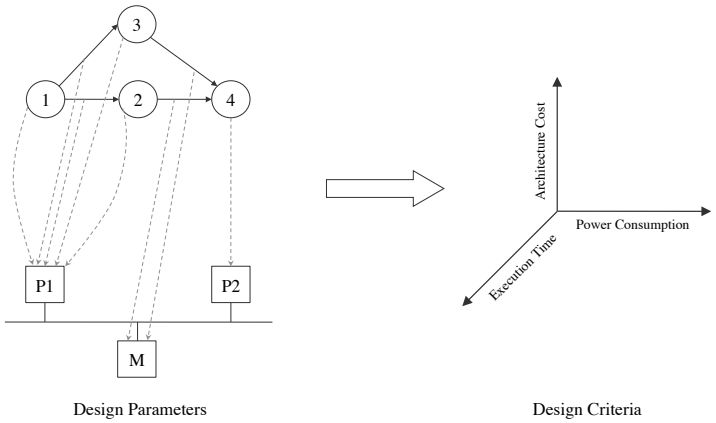


Figure 1.1: Illustration of design space exploration process on a simple example

combinations of design parameters have to be considered and tuned to find the ”best” tradeoff in terms of multiple design criteria. Such design space exploration, during which multiple criteria should be considered simultaneously, is called Multi-Objective DSE. Since objectives are often in conflict, there cannot be a single optimum solution, which simultaneously optimizes all objectives. Therefore, optimal decisions need to be taken in the presence of trade-offs between multiple design criteria. A set of optimal solutions is denoted as the ”Pareto optimal set”. This is the set of those solutions for which one objective cannot be improved further without causing a simultaneous degradation in at least one other objective.

In Figure 1.1 we illustrate the design space exploration problem on a simple example. In this example, the application description consists of four computational tasks and four communication channels. The architecture platform contains two processors and one shared memory. Let us assume three design criteria to be minimized, which are execution time, architecture cost and power consumption. Different mappings of application tasks and communication channels to various architectural components should be evaluated by simulation to find the optimum mapping solutions under the aforementioned design criteria. Each mapping decision corresponds to a single point in the design space. Moreover, each mapping decision consists of two parts: allocation and binding. The allocation determines which architectural components are actually used. The binding indicates which application task is executed by which allocated processor and which communication channel is mapped on which allocated processor/memory. Note that if two communicating processes are mapped onto the same processor, then their communications are done internally and therefore communication channel(s) between them are mapped onto the processor in question.

It is clear from Figure 1.1, even without taking into account any system criteria, that the number of alternative mappings (i.e. the size of the design space) is extremely large as it grows exponentially with the size of the application model (i.e. number of compu-

tational tasks and communication channels) as well as with the number of architecture components that can be used for the eventual implementation. As a consequence, an exhaustive evaluation of all potential mappings quickly becomes computationally intractable. Thus, heuristic search techniques such as evolutionary algorithms, ant colony optimization and simulated annealing can be used to search the design space for optimum solutions using only finite number of design point evolutions [1–4].

1.2 The Need for Visualization

System-level simulation frameworks that aim for early design space exploration may produce vast amounts of data on various characteristics (performance, power consumption, reliability, etc.) for the architecture(s) under investigation. Interpreting and drawing the right conclusions from such copious simulation results may be extremely cumbersome. Typically, the results of simulations are summarized in a number of quantities, such as average load, throughput, latency, or power consumption. Such numbers are useful to compare different configurations, but from these numbers alone it is often impossible to understand the reasons behind these differences and to spot anomalous behavior. So, they do not facilitate the understanding of the landscape of the design space, i.e., understand the often subtle relations between design decisions and their effects. For this, the data has to be studied in much more detail, and visualization is the most effective approach to accomplish this. Because of exactly this reason, other domains that also struggle with interpreting massive amounts of data, such as scientific computing, data visualization has become an invaluable tool to facilitate the data analysis. As a result, visualization has become a research field in its own right in these domains. Such visualization, however, is often domain specific and therefore not necessarily applicable to other fields.

Despite the clear benefits, little research is being conducted on methods and techniques for visualizing the results of (system-level) computer architecture simulations. Even worse, visualization approaches to support the process of architectural DSE is very much an unexplored research field. Existing visualization work in the context of computer architecture simulation mainly focuses on visualization technology for educational purposes (e.g., [5–7]), tightly couples visualization to one particular architecture simulation environment (e.g., [8–11]), visualizes only one specific aspect of embedded applications, often lower than system-level, such as memory access patterns [12], cache behavior [13–15] and data dependencies [16], or only provides some basic support for the visualization of simulation results in the form of 2D (and sometimes 3D) graphs (e.g., [17]). However, as we will demonstrate in this thesis, visualization can be extremely helpful to a system designer in interpreting and analyzing the DSE results.

1.3 Problem Description

The purpose of this research study can be summarized in the following question:

How can interactive visualization help to improve and accelerate system-level design space exploration of computer architectures?

Visualization of computer architecture simulations, and especially of those that target system-level design space exploration, is a largely unexplored research area. We expect, however, that by embracing visualization techniques, as has been done in other domains such as scientific computing and software engineering, much can be gained to improve computer architecture design technology. To this end, in this thesis, we introduce the structural use of visualization techniques in the design of embedded systems, and specifically for supporting the design space exploration of embedded systems. We incorporated all the developed visualization techniques into a tool, called VMODEX[1]. Actually, we build a bridge between two previously separate research fields: data visualization and embedded system design, and show how our proposed visualization techniques can help designers for better understanding and analyzing the DSE results.

An important challenge in this respect is how to define suitable metaphors to represent the abstract concepts, such as execution time, energy consumption, power, etc, in a visual form that is clear and understandable. For visualizing the abstract information, there is no natural counterparts in the real world and therefore the user has no predetermined mental model to which the data can be automatically mapped. Another important issue that needed to be addressed is that the visualization approach should not be limited to only two or three variables. It should be extendable to show multivariant data. Furthermore, the visualization technique should be scalable to support large design spaces. It is essential to be able to handle the large-scale experimental and simulation data sets. The efficiency of the visualization should not be dependent on the size of the data (i.e. the usefulness of the visualization should not be reduced by increasing the amount of the visualized data). Our proposed visualization techniques address all the above challenges.

For developing an effective visualization of data and information, user interaction plays an essential role. It allows users to directly manipulate the representation and to explore the data actively in order to investigate the data from different perspectives and discover new insights. In VMODEX, we provide various interactive capabilities, which allow designers to play with data and find out interesting and important features that may not be found just by looking at the static visualization.

In this thesis, we propose visualization techniques that support system-level multi-objective design space exploration process. The designer can use our visualization tool to gain insight into the landscape of the design space and look at the explored data from different perspectives and at multiple levels of abstraction. Furthermore, additional capabilities are developed to understand the dynamic search behavior of heuris-

[1] Visualization of Multi-Objective Design spacE eXploration

tic searching algorithms that are typically used in the DSE process (e.g. which parts of the design space are not visited by the algorithm). Moreover, we have proposed new visualization approaches to visualize the performance of different optimization methods from various points of view, which enable researchers to do detailed analysis of the quality of the outcome of searching algorithms that are used for exploring the design space. In addition, we define new visual representations of the results of decision making methods. These visualizations can help to gain a better understanding of the trade-offs between different design criteria and provide some guidelines for choosing the most preferred solution for the final implementation. For performing a comprehensive study of the DSE process, we define three separate stages and for each stage, we have developed several methods and visualization techniques to provide users a rapid and more accurate analysis. In the next section, we explain these stages in details.

1.4 Multi-Objective DSE Stages

The process of multi-objective design space exploration is not just evaluating different design points in terms of design criteria. There are some other issues that need to be addressed to perform a comprehensive DSE process. One important issue is developing an efficient searching algorithm for exploring the design space. This is because the size of the design space is usually too large to be explored in an exhaustive manner. Another problem is how one particular design point among the Pareto optimal points can be selected for the final decision. This question arises since in the presence of multiple conflicting objectives, there cannot be a single design point that simultaneously optimizes all objectives. Instead, a set of Pareto optimal points need to be found that provides trade-offs between multiple objectives. In the mathematical sense, all Pareto optimal solutions are regarded as equally desirable. Thus, some multi-objective decision making methods should be used for choosing the most preferred design point.

In this thesis, we define three stages for doing a comprehensive multi-objective design space exploration:

1. Developing the best optimization algorithm that efficiently and effectively explores the design space.
2. Exploring the design space using the best optimization algorithm, finding the Pareto optimal points and analyzing the DSE results (i.e. gaining insight into the landscape of the design space).
3. Selecting the most satisfying solution from the Pareto optimal set for the final implementation (i.e. multi-objective decision making).

Figure 1.2 shows the stages of multi-objective DSE. For each stage, we have developed several methods and visualization techniques for performing a rapid and more accurate analysis. Thus, using our proposed techniques, one can perform a comprehensive

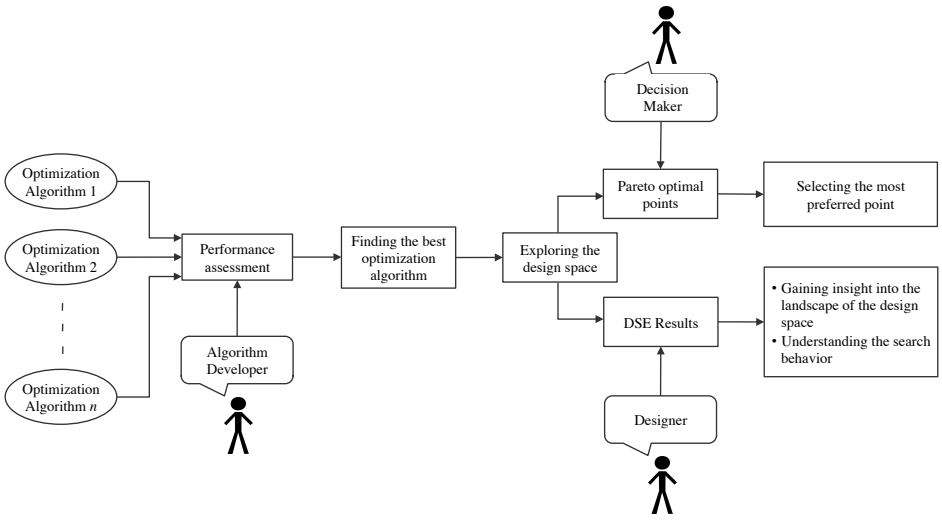


Figure 1.2: The stages of multi-objective design space exploration

study on the entire design space exploration process. In the following subsections, we describe each stage with more details.

1.4.1 Developing the Best Optimization Algorithm

Due to the sheer size of the design space in real problems, an exhaustive exploration of all possible alternatives is not feasible. Furthermore, usually multiple criteria need to be optimized simultaneously. Therefore, heuristic search techniques, such as Multi-Objective Evolutionary Algorithms (MOEAs), are often used for pruning an exponential design space and guiding the search process toward the most promising regions.

Although the goal of heuristic multi-objective optimization techniques is to find the Pareto optimal solutions with respect to the design criteria, there is no guarantee to reach real optimal solutions. This is because of the heuristic nature of these methods. They try to find optimal solutions. However, typically they are only able to find good approximations of optimal solutions that are not far away from the true optimal solutions. Furthermore, many different multi-objective optimization algorithms are proposed in literature, such as SPEA2 [18], NSGA-II [19], PAES [20], ACOG [21], AMOSA [22] etc., which may have a different performance on different problems, and there is no conclusive answer regarding to which algorithm is the best for a specific problem. On top of that, optimization algorithms are highly sensitive to the parameters being used, such as mutation rate, repair strategy, individual encoding, etc. These parameters have major effects on the performance of the algorithm and have to be fine-tuned by hand. Therefore, coming up with the best searching algorithm, which efficiently and effectively explores the design space and finds high quality solutions is

a big challenge.

Therefore, we have developed techniques to help algorithm developers to find the best optimization algorithm for their specific problem. These techniques have been incorporated in VMODEX. Thus, using VMODEX, algorithm developers can easily evaluate and compare the results of different optimization methods, with respect to their efficiency and effectiveness, in order to find the best approach. Then, the best optimization algorithm is used for exploring the design space and the results are delivered to the designers for analyzing the DSE process.

1.4.2 Exploring the Design Space

After finding the best optimization algorithm for a specific problem, this algorithm is used for exploring the design space and finding the Pareto optimal solutions. As the searched design space still is vast, interpreting all evaluation data and understanding how the optimization algorithm searches through or prunes the design space is cumbersome. Such analysis is, however, essential to the designer as it provides insight into the landscape of the design space (e.g., indicating which design parameters are more important than others). Therefore, we have developed a multi-objective DSE visualization, to understand how a heuristic optimization algorithm, searches the design space, where the optimum design points are located, how design parameters influence each objective, and provides insight into the relationship between the different objectives.

In addition, several interactive capabilities are provided, which enable the designer to analyze the data and explore the search result from different perspectives and at multiple levels of abstraction in order to discover interesting and important features that may not be found just by looking at the raw data or by using the traditional 2D/3D graphs.

1.4.3 Multi-Objective Decision Making

Finding the Pareto optimal points does not completely solve a multi-objective DSE Problem. The decision maker still has to choose one solution from this set to be implemented. However, the process of choosing the most preferred solution among the several Pareto optimal solutions is not trivial. Systematic approaches are needed to express preference information related to the multiple objectives and aid the decision maker to identify the most preferred solution. Without a systematic approach one cannot be sure that the proper decision has been made. In VMODEX, we have provided several decision making methods and visualization techniques to assist the decision makers to better understand the trade-offs between different criteria and guide them to make better decisions.

1.5 Thesis Organization and Contributions

In this thesis, we address the visualization of multi-objective design space exploration of multi-processor system-on-chip architectures. Actually, we build a bridge between two previously separate research fields. We introduce the structural usage of data visualization into the field of embedded systems design. This has not been done before, and in fact, we extend the boundaries of visualization to include new applications from other domains. We propose new visualization techniques for interpreting and analyzing the DSE results.

The main contributions of the thesis are presented in Chapters 3, 4 and 5, each focusing on a different stage in the process of multi-objective DSE, as explained in the previous section. The organization of the thesis and the main contributions of each chapter are as follows.

Chapter 2 gives an overview of the background information necessary for the rest of the thesis. We first describe the basic knowledge about multi-objective optimization problems. Then, we explain the multi-objective optimization problem in the context of design space exploration of embedded systems. This is followed by two subsections that describe two distinct issues in our DSE problem: the evaluation of a single design point and the searching strategy for exploring the design space. The first subsection introduces the Sesame system-level simulator [23, 24], which is used for evaluating design points. In the second subsection, we describe multi-objective evolutionary algorithms as a heuristic searching method for solving the second issue. Afterwards, we discuss the benefits of using visualization techniques for exploring and analyzing the data. Finally, we conclude this chapter and illustrate the need for employing efficient visualization methods for interpreting and gaining insight into the DSE results.

Chapter 3 introduces our interactive visualization tool, which is specially developed for understanding the multi-objective DSE process of embedded systems that are based on heterogeneous multi-processor system-on-chip architectures. It provides insight into the search process of heuristic searching algorithms that are typically used in the DSE process. Several interactive capabilities are provided, which allow designers to play with data and find out some interesting and important features that may not be found just by looking at the static visualization of the data. This chapter supports the second stage of multi-objective DSE, as explained in Section 1.4.2. In the following, we summarize our main contributions in this chapter:

- We model the design space as a tree in which both the design parameters and criteria are shown in a single view.
- The proposed tree model enables us to visualize multivariate data. There is no limitation on the number of neither design parameters nor criteria.
- Several techniques are provided in VMODEX to be able to handle large design spaces.
- In our DSE tree model, the concepts of subspaces and local Pareto points are

proposed, which are new concepts in the multi-objective DSE process and have not been considered before.

- Various methods and visualization techniques are proposed for evaluating and comparing different subspaces of the design space.
- Additional visualization approaches are defined for showing different aspects of the characteristics of each design point. These aspects are referred to as secondary objectives in this chapter and allow designers to interpret the data from different perspectives and gain additional insight into the underlying information.
- Additional capabilities are provided to help designers to understand the dynamic search behavior in heuristic based design space exploration.
- We present an extensive case study to show the benefits of using our tool in the DSE process.

Chapter 4 is dedicated to the first stage of multi-objective DSE (see Section 1.4.1), which is evaluating the performance of different multi-objective optimizers and finding the best one for a specific problem. In this chapter, we introduce various performance metrics and their visualization methods we have provided in VMODEX for comparing the outcomes of different optimization methods from several points of view. In multi-objective optimization problems, several distinct goals need to be achieved and therefore there cannot be a single quality measure that indicates the performance of an optimization algorithm in an absolute sense. Thus, various metrics need to be used to perform a comprehensive analysis of the performance of an optimization approach. Our main contributions in this chapter are as follows:

- Proposing three new metrics for assessing the performance of optimization algorithms in the objective space, namely: WSGR, σ_{mst} and DFPOS.
- Turning the focus of attention from exclusively evaluating optimization success in the objective space to also considering the decision space.
- Defining new goals and subsequently new metrics to evaluate the behavior of optimization methods in the decision space.
- Proposing new visualization approaches for both existing and new metrics to enable algorithm developers a deeper and more accurate analysis of the performance assessment.
- Presenting a case study, which demonstrates the usefulness of using VMODEX for performing a comprehensive study on evaluating and comparing the performance of different optimization approaches.

Chapter 5 addresses the last stage of multi-objective DSE (see Section 1.4.3), which is decision making process. As the last step of the multi-objective DSE process, the decision maker should select the most preferred design point from the set of Pareto optimal points. In this chapter, we explain the Multi-Objective Decision Making (MODM) methods that are provided in VMODEX and can help decision makers to understand the trade-offs between different criteria and select the final solution for the implementation. Furthermore, new visualization approaches are proposed, which provide the visual interpretation and detailed analysis of the results of the MODM methods. In this chapter, we describe the four basic problem formulations in MODM, which are: choice, classification/sorting, clustering and ranking problems. For each problem formulation, some decision making methodologies and their proposed visualizations are explained. Our main contributions in this chapter are as follows:

- We define a new method addressing the choice problem. Our method is based on the fuzzy dominance relations between the Pareto optimal points.
- We propose a new preference similarity measure for clustering the solutions. In our proposed similarity measure, unlike the conventional measures, the decision maker's preferences are integrated in the multi-objective cluster analysis.
- We introduce a scheme for constructing the cluster centers, considering the properties of all solutions inside the same cluster. Some clustering approaches utilize the concept of cluster center for assigning the objects to different clusters.
- For each MODM method, we propose a new visualization approach. These visual representations allow decision makers to find out how and why a particular solution is considered as a most preferred solution with respect to a specific MODM method.

Finally in Chapter 6, we first look back and summarize what we have achieved, and then look ahead to outline what can be accomplished next.

This thesis work has resulted in several reviewed publications. The content of this thesis is mostly based on these publications. At the end of the thesis, a list of all our publications is given.

2

Background Information

This chapter provides overviews of the key concepts necessary for the rest of the thesis. The readers need to know these relevant concepts to be able to understand the discussed issues. The first section describes the basic knowledge about multi-objective optimization problems. In Section 2.2, we explain the multi-objective optimization problem in the context of Design Space Exploration (DSE) of embedded systems. The DSE problems generally deal with two distinct issues. The first one is evaluating a single design point regarding all design criteria. In this thesis, the Sesame system-level simulator [23, 24] is used for evaluating each design point. An outline of the Sesame simulation environment is given in Section 2.2.1. The second issue in DSE problems is developing a search strategy for covering the design space during the exploration. This is because an exhaustive search of all possible design points is usually prohibitive due to the sheer size of the design space. Thus, heuristic search techniques are typically used to search the design space for optimum solutions using only finite number of design point evolutions. In this work, we use Multi-Objective Evolutionary Algorithms (MOEAs) as the searching strategy. Section 2.2.2 describes the general information about MOEAs. In Section 2.3 we discuss the benefits of using visualization techniques for exploring and analyzing the data. The significant aspects that should be considered for developing effective visualizations are also discussed. Finally, in the last section we conclude this chapter and illustrate the need for using visualization in interpreting and gaining insight into the results of the design space exploration process.

2.1 Multi-Objective Optimization

Real world design problems often are multi-objective optimization problems in which several incommensurable and often conflicting objectives should be optimized simultaneously. An improvement in one objective often causes deteriorations in other objectives. For example, consider the design of an embedded system such as a mobile phone, digital television, etc. Often the cost of such systems should be minimized, while maximum performance is desirable. Here, high performance and low cost are

generally conflicting. High-performance architectures can be quite expensive, while cheap architectures usually provide lower performance. Therefore, optimal decisions need to be taken in the presence of trade-offs between objectives. On the other hand, in multiple conflicting objectives problems, there cannot be a single optimum solution, which simultaneously optimizes all objectives; instead, a set of optimal solutions, denoted as the "Pareto optimal set", with a varying degree of objective values has to be found.

Definition 1: (Multi-Objective Optimization Problem) A general multi-objective optimization problem with n decision variables (parameters) and m objective functions is defined as:

$$\begin{aligned} &Minimize && y = f(x) = (f_1(x), \cdots, f_m(x)) \\ &Where && x = (x_1, \cdots, x_n) \in X \\ & && y = (y_1, \cdots, y_m) \in Y \end{aligned}$$

Without loss of generality, we assume a minimization problem. The duality principle [25], in the context of optimization, can be used to convert a maximization problem into a minimization problem by multiplying the objective functions by -1. By using the duality principle, handling mixed maximization/minimization problems is also easy. When an objective is required to be maximized, the duality principle can be used to transform the original objective for maximization into an objective for minimization. The objective function $f(x)$ maps a decision vector (solution) x in decision space (X) to an objective vector y in objective space (Y). Therefore, in multi-objective optimization problems, two different spaces should be taken into account:

- Decision variable space or parameter space
- Objective space or criterion space

One of the difficulties of multi-objective optimization is dealing with two spaces. In single-objective optimization, there is only one space (the decision space) and a searching algorithm works in this space by accepting/rejecting solutions based on their objective values. However, in multi-objective optimization, the objective functions constitute a multi-dimensional space and this additional space should be considered as well. This is because, for each solution, the position of its projection in the objective space determines how superior it is in the competition with other solutions. Although the search process takes place in the decision variables space, the objective space information is used in search operators to guide the searching algorithm towards the optimum solutions. Although these two spaces are related by a unique mapping between them, usually the mapping is not linear and therefore, proximity of two solutions in one space does not imply proximity in the other space. For instance, two close solutions in the decision space may have completely different projections in the objective space or two solutions that are far from each other in the decision space may lead

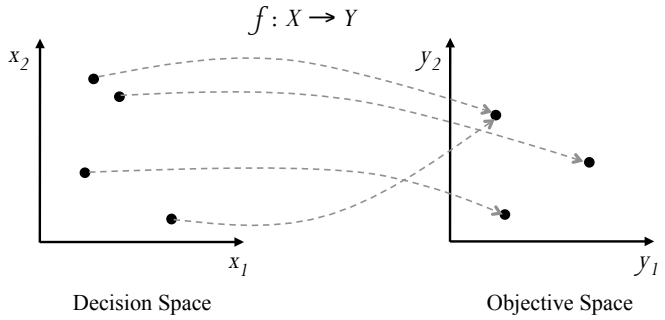


Figure 2.1: Representation of the decision space and its corresponding objective space in a two-objective optimization problem

to the same objective vector. Figure 2.1 illustrates these two spaces and a mapping between them for the case that both decision and objective spaces are two-dimensional ($n = 2$ and $m = 2$).

Most multi-objective optimization algorithms use the concept of domination for comparing solutions. In these algorithms, two different solutions are related to each other in two possible ways: either one dominates the other or they are non-dominated with respect to each other. In the following, we will describe the key concepts of multi-objective optimization.

Definition 2: (Dominance Relation) A solution $x_1 \in X$ is said to dominate the other solution $x_2 \in X$ (also written as $(x_1 < x_2)$ if and only if both of the following conditions are true:

1. The solution x_1 is not worse than x_2 in all objectives;
   formally: $\forall i \in \{1, \cdots, m\} : f_i(x_1) \leq f_i(x_2)$
2. The solution x_1 is strictly better than x_2 in at least one objective;
   formally: $\exists j \in \{1, \cdots, m\} : f_j(x_1) < f_j(x_2)$

It is intuitive that if x_1 dominates the solution x_2, the solution x_1 is superior to x_2 in the context of multi-objective optimization. It is also common to say, "x_2 is dominated by x_1" instead of saying "x_1 dominates x_2". A more rigorous definition of *strict dominance* requires x_1 to be strictly better in all objectives compared to x_2 and is written as $x_1 \ll x_2$, while the less strong definition of *weak dominance* only needs the trueness of the first condition and is denoted by $x_1 \leq x_2$. Two solutions x_1 and x_2 are said to be non-dominated with respect to each other ($x_1 \sim x_2$) if in some objectives x_1 is better than x_2 and in some other objectives x_2 is better than x_1, while on the other hand none of them is dominated by another one; formally:

$$\exists\, i \neq j \in \{1, \cdots, m\} : such\ that\ f_i(x_1) < f_i(x_2) \wedge f_j(x_1) > f_j(x_2)$$
$$\Rightarrow x_1 \not\leq x_2 \wedge x_2 \not\leq x_1 \Rightarrow x_1 \sim x_2$$

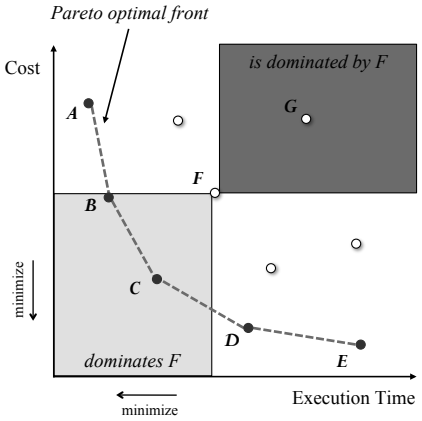


Figure 2.2: Illustration of the Pareto optimality and dominance relations between solutions in a two-dimension objective space

To illustrate the dominance relation, an example is given in Figure 2.2. The two-dimensional objective space is defined by execution time and cost of evaluated design points, both to be minimized. The solution F dominates solution G ($F < G$) since it is better in both objectives: it has a lower execution time and a lower cost. It would be even preferable if only one objective is improved, as in the case of comparing solutions B and F: they have exactly the same cost, but B achieves better execution time than F. Therefore B is superior to F and we say B dominates F ($B < F$). However, when comparing B and D, neither can be said to be superior, since $B \not< D$ and $D \not< B$. Although solution D is cheaper, it has a higher execution time than solution B.

In Figure 2.2, the dark gray rectangle encapsulates the region in objective space that is dominated by solution F. We can see that solution G is inside this region and therefore is dominated by F. The light gray rectangle shows the area in objective space, which dominates the solution F. This part contains two solutions B and C that dominate solution F. All the other solutions, which are not in these two rectangles are non-dominating with respect to F.

Definition 3: (Pareto Optimality) Let $x_1 \in X$ be an arbitrary solution (a decision vector)

- The solution x_1 is said to be non-dominated regarding a set $X' \subset X$ if and only if there is no solution in X' which dominates x_1;
  formally: $\nexists\, x_2 \in X' : x_2 < x_1$
- The solution x_1 is called Pareto optimal if and only if x_1 is non-dominated regarding the whole decision space X.

Definition 4: (Non-Dominated Set and Front) The set X' is called a non-dominated set if it contains only solutions, which are non-dominated regarding the set X', this

means that no solutions are dominated by other solutions in this set; formally $\forall x_1 \in X', \nexists\, x_2 \in X' : x_2 < x_1$. The image of a non-dominated set in the objective space is called the non-dominated front.

Definition 5: (Pareto Optimal Set and Front) The entirety of all Pareto optimal solutions is called the Pareto optimal set. Solutions in this set cannot be improved further in terms of a certain objective without causing a simultaneous degradation in at least one other objective. They represent in that sense globally optimum solutions. Any solution, which does not belong to the Pareto optimal set, is dominated by at least one Pareto optimal solution. The set of objective vectors corresponding to a set of Pareto optimal solutions is called the "Pareto optimal front" or "Pareto front".

In Figure 2.2, the black points construct the Pareto optimal front that contains five solutions: A, B, C, D and E. These solutions represent the optimum solutions but they are non-dominated with respect to each other. None of them can be identified as better than the others. We can see that there are trade-offs from one of these solutions to others; there is improvement along one objective and deterioration along another one. Therefore, further analysis, such as preference information, is needed to decide which one is the best for the final implementation. In Chapter 5, we explain different multi-objective decision making methods for comparing the Pareto optimal solutions and choosing the most appropriate solutions among them.

Definition 6: (Euclidian Distance) The Euclidian distance between two solutions a and b (of dimension n) is defined as:

$$\|a - b\| = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

If the objective functions have different scales of measurement, they should be normalized before calculating the distance measure. This prevents objectives with initially large ranges to overcome objectives with initially small ranges. For example, if there are two objectives, y_1 and y_2, in which y_1 can have values from 1 to 1000, and y_2 has values from 1 to 10, then the influence of y_2 on the distance function will usually be overpowered by the influence of y_1. Therefore, some form of normalization is necessary to make objective values scale independent and balance out the contribution of objectives on the distance measurement. There are many methods for normalization such as min-max normalization, z-score normalization and normalization by decimal scaling [26]. In this thesis, we use min-max normalization to obtain a "common scale" for all objectives. Min-max normalization performs a linear transformation on the original data values so that it does not change the initial distribution type. It transforms the data values into a desired range, which is usually $[0, 1]$. Thus, values of all objective functions are linearly transformed from their original ranges to lie within the desired range $[min_{target}, max_{target}]$. Suppose that f_i^{min} and f_i^{max} are the minimum and maximum value of the i^{th} objective. We would like to map interval $[f_i^{min}, f_i^{max}]$ into a target interval $[min_{target}, max_{target}]$. Consequently, for each solution $x \in X$ the value of i^{th} objective ($f_i(x)$) from the original interval will be

mapped into its normalized value (written as $\bar{f}_i(x)$) using the following formula:

$$\bar{f}_i(x) = \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}} \times (max_{\,target} - min_{\,target}) + min_{\,target} \tag{2.1}$$

In the case of using $[0, 1]$ as the desired range, the above formula can be simply written as:

$$\bar{f}_i(x) = \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{2.2}$$

After applying min-max normalization, each objective value will fit in the desired range of values. However, the underlying distribution of the corresponding objective within the new range will remain the same. Min-max normalization preserves exactly all the relationships among the original objective values and it does not introduce any bias.

2.2 Multi-Objective Design Space Exploration of Embedded Systems

Designers of modern embedded systems face several emerging challenges. Since embedded systems often target mass production and battery-based devices, they should be cheap and power efficient. In addition, they must, increasingly, support multiple applications and standards, for which they need to provide real-time performance. For example, mobile devices must support different standards for communication and coding of digital contents. Furthermore, modern embedded systems should also be flexible to enable easily extending them to support future applications and standards. Such flexible support for multiple applications calls for a high degree of programmability.

However, performance requirements as well as cost and power-consumption constraints require implementing substantial parts of these systems in dedicated hardware blocks. As a result, modern embedded systems often have a heterogeneous system architecture. They consist of components that range from fully programmable processor cores to fully dedicated hardware components for time-critical application tasks. Increasingly, such heterogeneous systems reside together on a single chip, yielding heterogeneous Multi-Processor System-on-Chip (MPSoC) architectures that exploit task-level parallelism in applications.

The heterogeneity of these highly programmable embedded systems and the varying demands of their target applications greatly complicate system design. The complexity of these systems forces designers to simulate systems and their components early during the design process to explore the wide range of design choices. Such Design Space Exploration (DSE), during which multiple criteria should be considered simultaneously, is called multi-objective DSE. Since objectives are often in conflict, a set of optimal solutions denoted as the Pareto optimal set has to be found. This is the set of

those solutions for which one objective cannot be improved further without causing a simultaneous degradation in at least one other objective (see Section 2.1).

In order to find the Pareto optimal design points with respect to multiple design criteria, the designer should ideally evaluate and compare every single point in this space. However, such exhaustive search of all possible design points is usually prohibitive due to the sheer size of the design space. When the design space is too large to be explored in an exhaustive manner, heuristic search techniques such as evolutionary algorithms can be used to search the design space for optimum solutions using only finite number of design point evolutions.

The DSE problem generally deals with two distinct issues:

1. The evaluation of a single design point regarding all objectives.
2. A search strategy for covering the design space during the exploration process.

Methods for evaluating design points range from solely analytical approaches to detailed cycle-accurate simulations. The results of simulation-based methods are more accurate but the evaluation process usually takes longer. A review of different approaches for evaluating a single design point is provided in [27]. In this thesis, the evaluation of design points is performed by the Sesame system-level simulator [23, 24]. Models in Sesame are defined at a high level of abstraction and capture only the most important characteristics of the components in the system. As these high-level models minimize the modeling effort and are optimized for execution speed, they can be applied for DSE at the very early design stages. Section 2.2.1 gives an outline of the Sesame simulation environment.

The searching strategies iteratively walk through the design space and try to properly cover the design space during their explorations. In general, these methods can be divided into two types, based on their progress from iteration to iteration: guided search and unguided search. The guided search methods, such as hill climbing [28, 29], evolutionary algorithms [30, 31], ant colony optimization [32–34] and simulated annealing [35, 36], use information learned so far to guide the search process. The unguided search methods such as random walk aim to provide an unbiased view of the design space. Each design point is chosen randomly and entirely by chance. In [27] a survey of different searching strategies used for design space exploration of system-on-chip architectures is given. In this work, we use multi-objective evolutionary algorithms as the searching strategy. MOEAs have several advantages; they evolve over a population rather than a single solution and in this way the search is performed in a parallel manner. Furthermore, MOEAs are generic and can be applied to a wide variety of search and optimization problems. Moreover, they are based on guided search methods. So, they learn by experiments and solve an optimization problem by successive refinement. Additionally, MOEAs are efficient at searching large and complex design spaces [37]. In Section 2.2.2 we explain the basic concepts of MOEAs and describe the searching procedure of a simple MOEA.

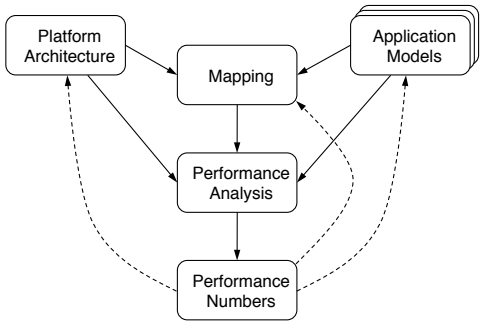


Figure 2.3: The Y-chart methodology

2.2.1 Sesame Environment

Sesame (Simulation of Embedded Systems Architectures for Multi-level Exploration) is a modeling and simulation framework geared towards efficient performance evaluation of heterogeneous Multi-Processor System-on-Chip (MPSoC) platforms in the multi-media domain [23, 24]. Models in Sesame are defined at a high level of abstraction and capture only the most important characteristics of the components in the system. By omitting detailed component properties, the simulation of an entire system can be much faster than with traditional simulation approaches. This allows for the (performance) assessment of a large number of design options.

Sesame follows the Y-chart design methodology [38, 39] which is depicted in Figure 2.3. A key aspect of the Y-chart approach is the separation of two main concerns: application models and architecture models. The application model describes the functional behavior of the system in an architecture independent manner. That means the application model is free from architectural issues, such as timing characteristics, resource utilization or bandwidth constraints. The architecture model represents the hardware components in the system, such as processors, reconfigurable modules, memories, etc. The Y-chart modeling methodology relies on independent application and architecture models in order to promote reuse of both simulation models to the conceivable largest extent.

In order to bind the application tasks onto the architectural resources an explicit mapping step is needed, after which the system performance can be evaluated quantitatively. In this step, different mappings of application processes and communication channels to various architectural components are evaluated by simulation to find the optimum mapping solutions under the design criteria. Each mapping decision taken in this step corresponds to a single point in the design space. Each mapping decision consists of two main parts: allocation and binding. The allocation determines which architectural components are actually used. The binding indicates which application task is executed by which allocated processor and which communication channel is mapped on which allocated processor/memory. Note that if two communicating processes are mapped onto the same processor, then their communications are done in-

ternally and therefore communication channel(s) between them are mapped onto the processor in question. In Sesame [1], the mapping decision problem is formulated as a multi-objective optimization problem in which three criteria are considered: the processing time, energy consumption and cost of the architecture. To solve this problem, an Evolutionary Algorithm (EA) has been used to achieve a set of best alternative mapping decisions under the aforementioned multiple criteria.

Application Model

The application model purely describes the functional behavior of the system. Applications in Sesame are modeled using the Kahn Process Network (KPN) [40] model of computation in which parallel processes, implemented in a high-level language, communicate with each other via unbounded FIFO channels. Reading on a FIFO channel is blocking and writing is non-blocking. The semantics of a Kahn process network state that a process may not examine its input channel(s) for the presence of data and that it suspends its execution whenever it tries to read from an empty channel. Unlike reads, writing to channels is always successful as the channels are defined to be infinite in size. Each of the Kahn Processes are instrumented with annotations. During execution, the annotated Kahn processes will generate three types of events: read, write and execute. Hence at any time, a Kahn process is either enabled, that is executing some code or reading/writing data from/to its channels, or blocked waiting for data on one of its input channels.

The advantages of KPN are their determinisim and Turing completeness. The deterministic property ensures that the same input will always produce the same output irrespective of the scheduling policy employed in executing the Kahn process network. Therefore, the deterministic feature of Kahn process networks provides a lot of scheduling freedom to the designer. Turing completeness means that the KPN can model any calculation that any programmable computer can perform.

The topology of the KPN is described in the Y-chart Modeling Language (YML). YML is an XML- based language which describes the Kahn Processes and their communication channels.

Mapping Layer

Because of the separation of application model and architecture model, an explicit mapping step is needed to relate these models for co-simulation. In Sesame there is an additional layer between the application model and architecture model layers, acting as a supporting interface in the process of mapping Kahn processes, i.e. their event traces, onto architecture model components. The objective of the mapping layer is illustrated in Figure 2.4.

The mapping layer comprises of virtual processors and FIFO buffers for communication between the virtual processors. As illustrated in Figure 2.4, there is a one-to-one relationship between the Kahn processes in the application model and the virtual

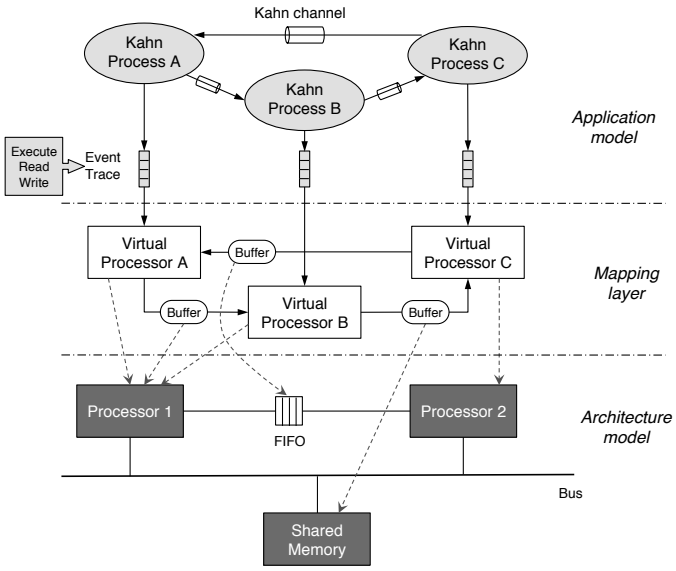


Figure 2.4: A close-up of the layers in Sesame: application model layer, architecture model layer, and the mapping layer which is an interface between application and architecture models

processors in the mapping layer. The same is true for the Kahn channels and the FIFO buffers in the mapping layer. However, the unbounded Kahn FIFO channels are mapped onto bounded FIFO buffers in the mapping layer. The size of the FIFO buffers in the mapping layer is parameterized and dependent on the architecture.

The virtual processors use trace driven simulation, processing the generated events in the annotated Kahn process node. Since the events are simulated, the virtual processors implement an event queue and provide scheduling for the events. In practice this means that the events of a single virtual processor are performed sequentially. An event is scheduled if the corresponding architectural element is available. In case of an execution event this means that the architectural element must be available for processing. For reading, the requested data must be available in the channel, whereas writing requires space in the bounded physical channel.

Unlike the projection from the application onto the mapping layer, the projection of the mapping layer onto the architectural layer does not need to be bijective. For each of the virtual processors, one of the architectural processors is selected. Similarly, for each virtual FIFO buffer a corresponding memory element is nominated. Obviously the mapping must be feasible. This means that the architectural processor is capable of executing all the annotated events, which are dispatched by the connected virtual processor. Moreover, the mapping of a virtual channel onto a memory element has the requirement that the memory connects the processors on which both of the virtual processors are mapped.

When multiple virtual processors are mapped onto the same architectural processor

scheduling is required. Sesame provides customizable schedulers to schedule events originating from different virtual processors. An example can be seen in Figure 2.4. In this case, both the virtual processors A and B are mapped onto the same architectural processor (Processor 1). As a consequence, Processor 1 must first have a scheduler in order to determine if it processes an event of virtual processor A or B.

Architecture model

An architecture model is constructed from generic building blocks provided by a library, which contains template performance models for processors, co-processors, memories, buffers, busses, and so on. The elements are implemented in a discrete event simulation language like Pearl [41] or SystemC [42]. Using the input event traces, which may come from different virtual processors, execution is simulated. From the simulation results, several metrics can be acquired, like performance and utilization. Like the application model and the mapping model the topology and the characteristics of the architecture model are also described using YML. In this case, the parameterized individual components and the connections between them are described.

2.2.2 Multi-Objective Evolutionary Algorithms

In the last few years, Multi-Objective Evolutionary Algorithms (MOEAs) have been recognized to be well suited to solve multi-objective optimization problems and have become very popular. They are widely used in many different fields such as computer science, engineering, economics, finance, industry, chemistry, ecology and etc. Over the years, many MOEAs have been proposed such as SPEA [43], SPEA2 [18], NSGA-II [19], PAES [20], MOGA [44], MAREA [45], etc. The main motivation for using EAs to solve multi-objective optimization problems is because EAs operate on a set of design points (solutions) rather than only one solution at each time. Thus, in a single run of the algorithm, several solutions are evaluated and in this way the search is performed in a parallel manner. Furthermore, EAs are less sensitive to the shape or continuity of the Pareto front. For example, they can easily deal with discontinuous, multi-modality and concave Pareto fronts. Additionally, EAs are able to search large and complex decision spaces. Moreover, it is not necessary to choose an accurate starting point. The results of EAs do not rely on the initial population and they aim at finding the global Pareto optimal solutions, rather than to get stuck in local optimums. These features play a critical role in the efficiency of EAs in solving multi-objective optimization problems. These issues are, on the other hand, major concerns when using traditional mathematical programming techniques for finding optima, such as linear programming, nonlinear programming, stochastic programming, robust optimization, combinatorial optimization, etc.

Evolutionary algorithms are heuristic search methods that take their inspiration from natural biological evolution. By analogy to natural evolutions, solution candidates

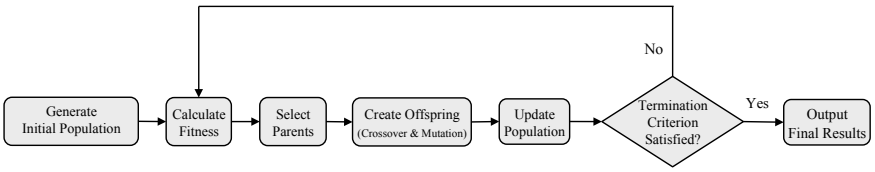


Figure 2.5: Basic procedures of evolutionary algorithms

are called individuals and a set of solutions is called a population. Each individual represents a possible design point in the decision space. However, an individual is not a vector in the decision space. Instead, it is an encoded representation of a decision vector.

MOEAs evaluate a population of individuals over several iterations, called generations. With the help of genetic operators, a MOEA progresses iteratively towards the best possible solutions. Its progression (optimization process) is comparable to natural selection in the evolution of living organisms in that the most proper elements in the population have a higher chance to survive, reproduce and thus transfer their genetic material to the consecutive generations. The algorithm usually starts with a randomly generated population and calculates the fitness value for each individual within the population. The fitness function is problem specific and determines how good the individual is. Individuals with high fitness in the current population are selected for reproduction and referred to as parent individuals. Such individuals are modified (recombined and randomly mutated) to create new individuals (offspring). This is motivated by the hope that the new individuals will be better than the old ones. The new individuals are then considered as the current population in the next iteration of the algorithm. Generally, the algorithm terminates when a predefined number of generations has been reached. But, also other conditions may be used as the termination criterion of the search process. For instance, when there is no improvement in the fitness values of the individuals, or some individuals with sufficient quality have been found. At the end, the best individuals found in the final population or during the entire search process are considered as the outcome of the MOEA. Figure 2.5 shows the sequence of basic steps in a simple evolutionary algorithm. In the remainder of this section, we will briefly describe the main concepts used by MOEAs.

Individual Encoding

The first step in the design of an evolutionary algorithm is to translate the real problem into *biological terms*. A method is needed to represent the design variable values in the allowable sets, so that they can be used and manipulated by an evolutionary algorithm. Thus, design points in the decision space should be encoded based on an appropriate structure and these encoded representations are used as individuals in EAs. The encoding mechanism is highly dependent on the problem. Binary encoding is the most common and simplest method. In binary encoding each individual is a

set of bits, 0 or 1. Each bit represents a parameter of the design point. An important example of a problem that is solved using binary encoding is the *knapsack problem*: given a set of items, each with a weight and a profit, select some number of items to be carried in a knapsack with certain capacity. The objective is to choose the set of items that maximizes the profit but does not exceed the knapsack capacity. As the encoding concerns, for each item, there is one bit in the binary encoding. Therefore, the total number of bits representing an individual is equal to the total number of items that we have. The value of each bit $(0/1)$ indicates whether the corresponding item is in knapsack or not. The benefit of a binary encoding is that the implementation of the genetic operators (crossover and mutation) is rather straightforward. A downside is that binary encodings are unnatural and unmanageable for many problems.

A simple alternative is a many-character encoding. Instead of only having the 0 or 1, a larger alphabet is available. This alphabet can contain characters, strings, integers or even real values. The large freedom in choosing an alphabet makes this encoding applicable to many problems. A designer can also choose to use a problem specific encoding. These are encodings, which are not commonly used and mostly invented for a specific problem. An example is the tree encoding, which can, for example, represent computer programs.

Fitness Calculation

The quality of an individual with respect to the optimization task is calculated using the fitness function. The fitness value indicates how well an individual is and determines its desirability in comparison to the other individuals. Note that since the quality is related to the objective functions, before calculating the fitness, an individual should be decoded to its corresponding vector in the decision space. Thus, the fitness function can be applied on decision vectors not on their encoded representations (individuals). The fitness assignment method is a major component of MOEAs and may significantly affect their performance. Therefore, different fitness assignment strategies are proposed in literature to improve the performance of MOEAs. Two popular schemes are objectives-based fitness assignment such as VEGA [46], and domination-based fitness assignment such as PAES [20], SPEA2 [18], and NSGA-II [19].

Parents Selection

Selection method indicates how the individuals from the current population are selected to produce offspring for the next generation. The purpose of the selection process is to remove low-quality individuals from the population, while reproducing high-quality individuals, in the hope that their offspring will have an even higher quality. The goal is to focus on particular portions of the search space, which contains fitter solutions and to increase the average fitness within the population. The probability that an individual is selected as a parent for further reproduction is determined by a probability function with respect to its fitness value. The probability that the fittest individual is chosen must be large enough, otherwise the algorithm will not converge

to optimal solutions. However, if the probability of choosing the fittest individual is too high, the algorithm can have a tendency to get stuck in a local optimum. Selection is done with replacement, so an individual can be selected multiple times as a parent. Like the fitness assignment, various selection mechanisms have been suggested in literature such as roulette wheel, elitism, rank, tournament selection and so on. A detailed description of how these methods work can be found in [37].

Variation Operations

The aim of variation operations is generating new individuals by modifying the existing ones. In general, two variation operators are typically applied: crossover and mutation. The variation operators work on individuals, not on the decoded decision vectors. The implementation of these operators is highly dependent on the individual encoding. The general view is that crossover is supposed to exploit the current individuals to find better ones and is helpful for rapidly exploring the search space and finding optimum points, while mutation provides a small amount of random search, and helps to ensure that no point in the search has a zero probability of being examined. Therefore, mutation is effective to investigate new and unknown areas in the search space. In the following paragraphs crossover and mutation will be described with respect to a binary encoding.

Crossover The crossover operator takes two individuals as parents and creates two different offspring individuals by recombining the parents. To take into account the stochastic nature of evolution, a crossover probability is assigned to this operator. Therefore, substrings from two parents are swapped between these parents with a fixed probability. Crossover can be done in many ways.

One-point crossover is the simplest form. In the one-point crossover (see Figure 2.6(a)), two parent individuals are cut at a random point and the segments after the cut point are swapped to create the offspring. The one-point crossover has some drawbacks. Based on the location of the bits in the individual, some schemas cannot be generated. An example is that 10001 and 00100 cannot generate 10101. Another shortcoming is that the head and tail of one individual cannot be passed simultaneously to the offspring. If both the head and the tail of an individual contain good genetic information, none of the offspring obtained directly with one point crossover will share the two good features.

Using a two-point crossover avoids the second drawback of the one-point crossover. In this case, two positions are chosen at random and the segment between these two points is exchanged. Figure 2.6(b) shows an example of two-point crossover. Note that in both one-point and two-point crossover, only a single segment is exchanged. The two-point crossover is generally considered better than one-point crossover, but it still has the problem that certain schemas cannot be generated.

A method, which is capable of generating any schema, is uniform crossover. In the uniform crossover each bit in the offspring is selected randomly, either from the first

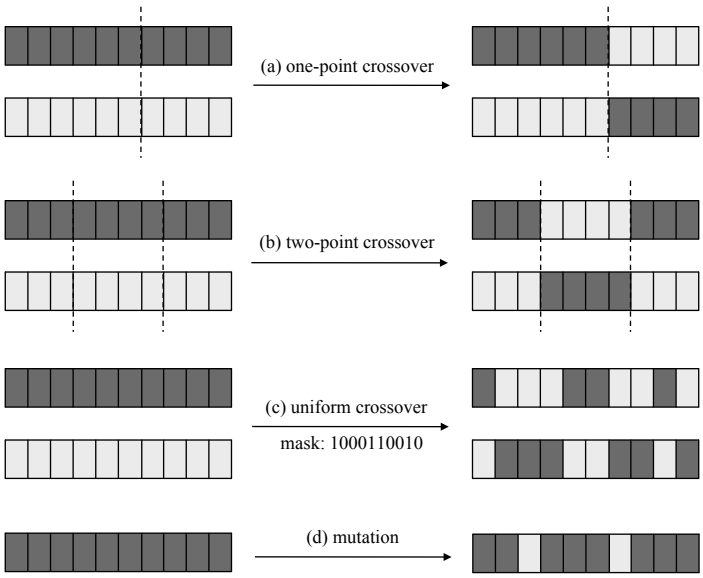


Figure 2.6: BIllustration of variation operations

parent or from the second one. A crossover mask with the same length as the individual structure is created at random and the parity of the bits in the mask indicates which parent will supply the offspring with which bits. Where there is 1 in the crossover mask, the corresponding bit is taken from the first parent and where there is 0 in the mask, the bit is taken from the second parent. Usually, for producing the second offspring the inverse of the crossover mask is used. In, Figure 2.6(c) an example of uniform crossover is shown. Unlike the one-point and two-point crossover, multiple segments can be exchanged between the parents in the uniform crossover. There are also some other types of crossover such as N-point crossover, three parent crossover, shuffle crossover, ordered crossover, etc. Here, we describe only the most popular ones. For more information about these crossover operators, the reader is referred to [47]. The question of which of the crossover methods should be used is not easy to answer. The success or failure of a crossover operator depends on the particular fitness function, encoding and other details in the evolutionary algorithm.

Mutation The mutation operator randomly alters each bit of an individual according to the mutation probability. It prevents the algorithm to be trapped in a local optimum. Furthermore, mutation helps to maintain genetic diversity in the population. It creates new genetic structures in the population by randomly modifying some of its building blocks. A simple mutation for binary encoding is shown in Figure 2.6(d). In this method, the value of each bit is inverted (changing 0 to 1 or vice-versa) with a small probability. The mutation probability should be small otherwise the EA will in fact change to random search.

Constraint Violations

Duo to the randomness in MOEAs (in initialization, crossover and mutation steps), they are prone to violating the problem constraints and producing invalid individuals. There are two options for handling the invalid individuals: use a repair mechanism or use a custom variation operator. If the repair mechanism is used, the variation operator manipulates the individuals freely. After an individual is manipulated, the repair mechanism will verify if it is still valid. When it is not a valid individual anymore, the repair mechanism will repair it. The other option is to use a custom variation operator. The custom variation operator is built in such a way that it only generates valid individuals.

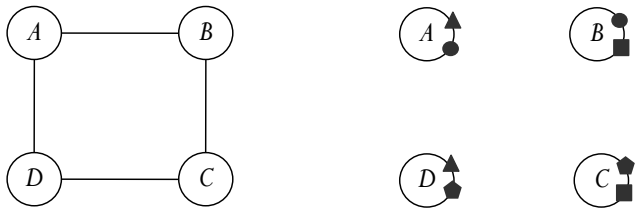
Parameters Setting

In implementing an evolutionary algorithm, various parameters need to be set, like population size, crossover type and probability, mutation probability, selection method and termination condition. The success of an evolutionary algorithm is greatly dependent on the values of these parameters. In literature, there is a lot of discussion on setting the parameters of evolutionary algorithms, However, there are no conclusive results on what the best parameters are, most people use what has worked well in previously reported cases. A study by Schaffer et al. [48] performed extensive experiments to test the success of evolutionary algorithms on a small set of numerical optimization problems with a wide range of parameter settings. They found the best settings of the parameters independent of the problem in their test suite. The best population size was between 20 and 30, the crossover probability between 0.75 and 0.95 and the mutation probability between 0.005 and 0.01.

There is also another type of evolutionary algorithm called Adaptive Evolutionary Algorithm (AEA) [49, 50] in which the parameter values are varied during the search process. A simple scheme could be as follows: the mutation probability is changed according to the quality of population. If there is no improvement in the fitness values for a long time, the mutation probability will be increased. Subsequently, the mutation probability is decreased again when an improvement occurs.

Summary

The power of EAs comes from the fact that they do exploration and exploitation simultaneously. Exploration is essential to walk through the decision space and discover new parts. Exploitation is helpful to make use of knowledge found at previously evaluated points and to guide the searching process towards better points. In EAs, the direction and chance in the search process are combined effectively and efficiently and provides a fast, useful and robust technique, which can deal successfully with a wide range of different problems. Although there is no guarantee that EAs will reach the global optimum solutions, they are generally good at finding ”acceptably good” solutions in a ”reasonable time”.



(a) Line connection (b) Symbol connection

Figure 2.7: Two different graphical methods for showing connections between entities

2.3 Visualization

Collecting information is no longer a problem, but interpreting and extracting insight from collected information has become progressively more difficult. Visualization provides a link between two potent information-processing systems: The human eyes and modern computers. Visualization is the process of transforming data, information, and knowledge into visual form making use of humans' natural visual capabilities [51]. Visualization is useful to rapidly extract insights from large amounts of information. It allows users to quickly understand the general structure and patterns in the data and the relationships between different variables.

An important issue in designing visualization systems is that visualizations are primarily developed for human interpretation. Thus, understanding how the human perceives information visually and how his mind responds to visualization is a key factor for designing an effective visualization. The human perceptual and cognitive capability must be considered in producing a visualization system, otherwise the end result may lead to a visualization that cannot be interpreted by the human. For instance, Figure 2.7 shows two different ways for demonstrating connections between entities. The connecting lines in Figure 2.7(a) are much more effective than the symbols in Figure 2.7(b).

Perhaps the best example to describe how visualizations can allow better understanding of data is Mendeleev's Periodic Table of Elements. Other examples are maps illustrating subway connections, statistical diagrams showing financial information, MRI scans for detection of health problems and weather maps showing the climatic and meteorological situation as predicted for a day or a week.

2.3.1 Visualization Classification

Visualization is commonly classified into two groups based on its application and scope: scientific visualization and information visualization.

Scientific visualization focuses on the visual display of spatial data related to scientific studies such as the bonding of molecules in computational chemistry. It inspires the

representations from the physical world as can be seen from the MRI-scan and tornado examples. The data may be generated by complex simulation models or collected by scientific instruments such as medical scans, satellites, telescopes, and microscopes. The main characteristic of scientific visualizations is the fact that they have natural counterparts in the real world to which the data can be mapped, such as the human body, earth, airplane, and molecule. Actually, scientific visualization helps to amplify the human sensory system, by showing things that are too fast or slow (on timescales) for the human eyes to perceive, or structures that are much smaller or larger than human scale, or phenomena such as X-rays radiation that cannot be directly sensed by human.

Information Visualization (InfoVis) deals with abstract data sets, that is, data without a ”natural” physical or geometric representation, such as hierarchical or textual information. Therefore, the user has no predetermined mental model to which the data can be automatically mapped. Thus, the main challenge in information visualization is developing new visual metaphors for representing the abstract data, such that they enable users to clearly and effectively understand the underlying information.

These two visualization groups are not mutually exclusive. For instance, scientific visualization may use the techniques in information visualization for representing multiple attributes of a physical object.

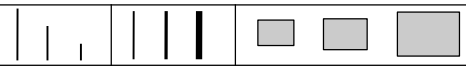
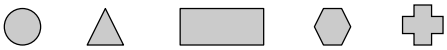
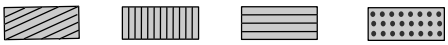
2.3.2 Information Visualization

Information visualization is about transforming abstract information into visual representations providing users the benefits of quick insight, correlation analysis and pattern recognition that are inherent in visual information. Information visualization should enable users to get information fast, make sense out of it, and draw conclusion from it in a relatively short time.

The field of information visualization is an interdisciplinary research area. It integrates multiple and diverse discipline such as computer science, mathematics, statistics, psychology, semiotics, graphic design and domain knowledge. The two main issues in computer science that are related to information visualization are computer graphics and human-computer interaction. Mathematics and statistics are necessary for accurate mapping of information to the visual elements. The field of psychology provides noteworthy scientific guidelines on how humans perceive visual information. The area of semiotics is useful for understanding the symbols and their underlying meanings. The knowledge of graphic design helps to create visualizations that are well suited for their intended purpose. It is evident that for designing a useful and effective visualization a good knowledge of domain area is essential.

Information visualization systems have two main components: representation and interaction. The representation component involves the way that data is mapped to the visual form. Subsequently, the interaction component allows a user to directly manipulate the representation and to explore the data set in order to discover additional insights. Although the representation and interaction are discussed as two separate

Table 2.1: Basic visual variables

Visual Variables	
Position Changes in the x,y,z locations	
Size Changes in length, thickness, or area	
Shape	
Color	
Saturation Changes from light to dark	
Orientation	
Border	
Texture type	
Texture density	

components, they are not mutually exclusive. For example, an interaction with visualization may lead to a change in representation. Nonetheless, these two components compose the two fundamental aspects of InfoVis systems.

Representation

In all visualizations, graphical elements are used as a visual syntax to represent semantic meaning [52]. This mapping of information to visual elements is called visual encoding, and the combination of several encodings in a single display leads to a complete visual metaphor. A simple example is the mapping of temperature to a color scale from blue to red. Cool and warm colors may be used to represent low and high temperatures, respectively. The visual variables are considered as the fundamental building blocks of any visualization. Several basic visual variables are shown in Table 2.1.

To effectively map data onto visual variables we need to consider the properties of data as well as the characteristics of visual variables. For instance, size or length is more effective to encode quantitative data, but less useful for nominal data. Contradictory, shape is ineffective in representing quantitative data, while is more suitable for nominal data.

Another essential issue in designing an effective visual encoding scheme is that the visualization system designer should be aware of standard conventions in visual representations. For example, a normal visual convention for the three colors red, orange and green is that they denote danger, caution and safety respectively (e.g. traffic light). If the use of these colors is reversed or jumbled up, then the confusion can be caused because of the contradiction with perceptual processing. The users have to redefine their mental model of what a particular color means. Utilizing unusual information coding representations makes the meaning or any understanding of the visualization virtually impossible.

Interaction

User interaction plays an essential role in effective visualization of information. It provides users the ability to directly manipulate and interpret representations. The way users interact with visualizations can strongly influence their understanding of the data as well as the usefulness of the visualization systems. Interactive visualization allows a tighter connection between users and the data being explored and enables users to rapidly and effectively discover hidden characteristics, patterns and trends in the data and gain a greater insight. This insight is associated to human cognition and the powers of reasoning within a logical framework, not to the ability of computers to process data. Furthermore, interactions can overcome some limits of representations (e.g. screen limitations) especially when the data grows larger and more complex.

In the following we discuss the general categories of interaction techniques, which are widely used in InfoVis systems.

Navigation: Interactive navigation consists of changing either the viewpoint or the zooming factor of the scene. Panning is an example for altering the viewpoint. Panning refers to movement of a camera across a scene and can simply be achieved by scrollbars. Zoom in and zoom out are powerful interaction techniques that allow users to get an overview of the entire scene or get a more detailed view on some specific parts of the scene.

Manipulating the visualized objects: The basic manipulation operations are moving the objects in the scene, rotating, translating, selecting and scaling.

Reconfigure: Reconfigure interaction techniques provide users with different perspectives onto the data set by changing the way that data items are arranged. A simple example is sorting the rows of a table by a specific column and rearranging the table rows. Another example is changing the variables assigned to the axes of a scatter plot view or rescaling the axes to exclude outliers and expand the scale to see whole data. The baseline adjustment feature in a stacked histogram is another example of reconfigure interaction. It enables users to better compare the heights (values) of subsections

that are not initially on the bottom of the histogram.

Adjusting the level of abstraction: These types of interactions enable users to modify the abstraction level of the representation from an overview down to details of individual data items and often many levels in-between. A simple example is the tool-tip interaction technique that provides detailed information when a mouse cursor hovers over a data item. Another example is expanding/collapsing nodes in tree visualization. By expanding a node its corresponding subtree is shown and allow user to do more detailed analysis on the relations between that particular node and the nodes in its subtree. However, by collapsing the node its subtree becomes invisible.

Filtering: Filtering enables users to change the set of presented data items based on some specific conditions. In this type of interaction, users specify some conditions, so that all data items meeting those criteria are presented and those data items that do not satisfy the filtering conditions are removed from the display or shown differently. For instance, a user can select interested ranges by moving sliders or particular values by clicking on check boxes and then the data items meeting those constraints are shown and others are removed from the display.

Brushing: Brushing interaction techniques are used to highlight associations and relationships between data items in multiple views. In some Infovis systems multiple views are used to show the same data set from different perspectives (e.g. scatter plot, parallel coordinate, histogram). The brushing technique is used to highlight the corresponding data items in all views. For instance, when a user selects a data item in the scatter plot, the same data item in the parallel coordinate view is highlighted simultaneously.

2.3.3 Multivariate Data Visualization

Multivariate data visualization is a specific kind of information visualization in which the correlations between multiple variables are of vital interest. For multivariate data visualization, the dataset to be visualized is of high dimensionality and its variables are correlated in some way. Formally, the data is considered multivariate if it has more than three variables. The human perceptual system is well equipped to interpret data as presented in $1D$ and $2D$ graphical representations, or even $3D$ if it is properly interfaced. The visualization techniques for multivariate data try to effectively map high-dimensional multivariate data in a $2D/3D$ visual display. Such an effective visualization should be able to facilitate users to identify, locate, distinguish, categorize, cluster, rank, compare, associate or correlate the underlying data [53].

Multivariate data visualization faces several challenges in encoding its variables in a single visual display, as follows:

- All data variables should be shown simultaneously but avoid cognitive overloading.
- Different variables can be viewed holistically for integrated analysis and, at the same time, users can judge each dimension separately and independently.
- There is a tradeoff between amount of information, simplicity and accuracy.
- Visualization of relationships is an essential issue in multivariate data visualization. The information is not just data items but also relationships, both within multidimensional data items and between them.
- In most cases certain correlations are not know in advance, and we want to discover those correlations using visualization. It is a paradox [54] that makes it hard to assess the effectiveness of an information visualization technique: we do not know what valuable knowledge exists in the data, so we hope to gain insight by visualizing it.

In the following, we describe three most popular visualization techniques used for representing multivariate data.

Scatterplot Matrix

A scatterplot is the conventional method for visualizing the relationship between two variables in which two variable values are projected along the $x - y$ axes of the Cartesian coordinates. As a visual structure, the scatterplot uses position to encode two variable values and their relationship.

Scatterplot matrix is the standard way of extending the scatterplot to higher dimensions. The scatterplot matrix contains all the pairwise scatterplots of the variables in a matrix format. That is, if there are n variables, the scatterplot matrix is an $n \times n$ matrix such that the i^{th} row and j^{th} column of this matrix is a scatterplot of i^{th} variable versus j^{th} variable. The diagonal plot is simply a 45-degree line since we are plotting a variable versus itself. Thus, it is common to use the diagonal to write the variable labels or to plot the univariate histogram. Since the scatterplot matrix is symmetrical, it is possible to omit the plots below the diagonal. Figure 2.8 shows an example scatterplot matrix for a dataset with three variables. In this figure, the label and histograms of variables have been placed on the diagonal of the matrix.

Using the scatterplot matrix, we can observe patterns in the relationships between pairs of variables, but there may be important patterns in higher dimensions, which are barely recognized in it. To address this problem, the brushing technique can be applied. Using brushing, a particular n-dimensional subspace in the visualization can be highlighted, that is, the respective points of interest in each scatterplot in the matrix are colored or highlighted.

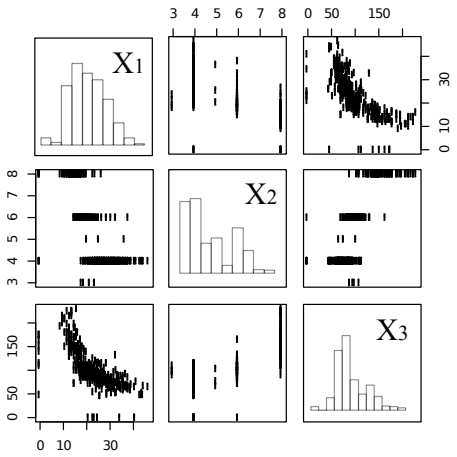


Figure 2.8: An example scatterplot matrix for 3-dimensional data

Parallel Coordinate Plot

A parallel coordinate plot [55] overcomes the limitation of the orthogonal coordinate system by placing the coordinate axes parallel to each other. With this arrangement, it is possible to plot high-dimensional data points on a plane. Figure 2.9 shows a $2D$ data point in both axis configurations: orthogonal and parallel.

In parallel coordinate plots, the maximum and minimum values of each dimension are scaled to the upper and lower points on a vertical axis. An n-dimensional data point is displayed as a polygonal line that crosses each axis at its respective data variable value. Figure 2.10 represents an example parallel coordinate plot for a data set with 7 variables. In this figure, data variables are shown as parallel axes and each data item is represented by a polygonal line. In parallel coordinate plot, the order of the axes is critical for finding features. Therefore, in typical data analysis, many reordering will need to be tried. In some cases [56], heuristics are used to create illuminating orderings.

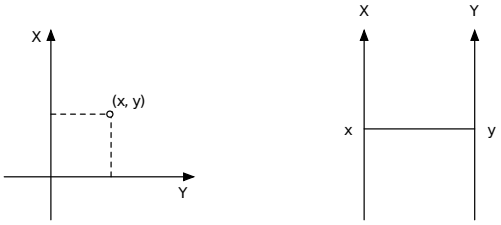


Figure 2.9: A $2D$ data point (x, y) in orthogonal axis configuration (left) and in parallel axis configuration (right)

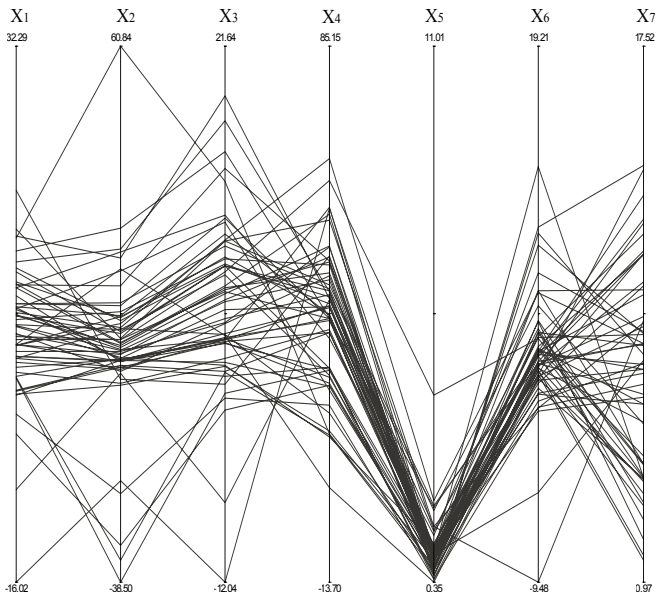


Figure 2.10: An example parallel coordinate plot for 7-dimensional data

Parallel coordinate plots have been shown to be an effective tool for an initial visual analysis of high-dimensional datasets. However, they have two main weaknesses. First, distributions of observations and patterns in the dataset become obscured as the number of observations increases. Second, to visually represent relationships between any two variables, those variables must be adjacent to each other in the plot. Many investigators have proposed solutions to overcome these shortcomings such as brushing, color density encoding, and axis reordering.

Influence Explorer

The influence explorer [57, 58] is an interactive visualization tool for any design problem in which performances can be computed from the knowledge of parameter values. This tool makes use of "Interactive Histograms" technique. Each performance and each parameter has its own histogram. A data item is represented once on each histogram by a square in the appropriate bin. The height of each column in the histogram represents the number of data items that fall within that bin. In other words, the histograms are frequency plots. The squares for a particular data item can be joined using lines. Collection of such lines can provide useful insight into the relationships implicit in the display data (parallel coordinate plots).

Figure 2.11 shows an example data set, which is displayed in the form of histograms. On the left side, the two performances P_1 and P_2 and on the right side the four pa-

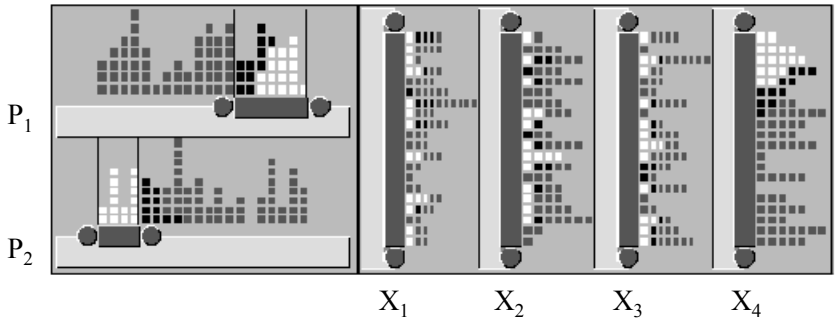


Figure 2.11: An example data set visualized by the influence explorer

rameters $X_1 - X_4$ are shown. Each square within a histogram corresponds to a single point. Selecting a point on one scale shows its corresponding values on the others. In Figure 2.11 a selection has been made on P_2 and those same points are highlighted (in white) on each of the other histograms. The circles indicate the mean of the selected points. Upper and lower limits can be set on all the performance scales. Whereupon color-coding indicates how the points satisfies those requirements. White color-coding identifies those design points that satisfy all the performance limits, black points fail one and dark grey points fail two requirements. Such color-coding provides valuable sensitivity information. Examination of the location of white squares on the parameter scales will indicate where possible designs might lie, and will often help to identify the relation between performances and parameters. The same as performance, the selection of parameter limits is also possible. However, it requires additional color-coding. The main advantage of such color-coding is that it indicates how altering the parameter or performance limits will affect the overall usefulness of the design.

2.3.4 Visualization Evaluation

Evaluating a visualization system is much more difficult than evaluating other systems. This is because InfoVis systems focus on an exploratory analysis process and this poses additional challenges. It is hard to judge whether a visualization system really helps users to get specific insights and more comprehensive understanding of the explored data in a quantitative manner. In other domains, an evaluation can be made based on whether the task completion is faster or the error rates are reduced, but how can we make the case for measuring the effectiveness criteria of a visualization system?

In the past few years, some researchers in the filed of information visualization have discussed the necessity of evaluation studies of InfoVis systems and proposed some guidelines for evaluating visualizations (e.g. see [59–61]). In [61] the authors defined four thematic areas for evaluating InfoVis systems as follows.

1. Usability assessments: typically evaluate visualizations to determine and solve user interface problems.

2. Controlled experiments comparing design elements: Investigate human visual perception. That means examine different techniques for mapping the information to visual display to see which one makes the most sense and provide the user with the right knowledge.

3. Controlled experiments comparing two or more tools: Try to compare a novel visualization technique with the state of the art.

4. Case studies of visualization tools in realistic settings: Do real tasks and demonstrate feasibility and in-context usefulness.

In the literature, controlled experiments using predefined tasks are the most common methods for identifying the effectiveness of visualization systems. In this method, the performance of users in terms of the time taken to complete the predefined task and the accuracy of the completed tasks (e.g. the number of correct/incorrect answers) is measured. Although the controlled experiments are useful for examining specific perceptual effects and provide researches a method for measuring the usability aspects of their visualizations, it has some drawbacks. It can provide results for only the limited set of predefined tasks and thus, the evaluation can not be generalized beyond these (simple) predefined tasks. Furthermore, the tasks may be chosen with some bias toward what the system is capable of doing it. Moreover, in this method, users are forced to follow the specific instructions during the experiments and are constrained to a particular line of thought developed in advance. Thus, there is little room for exploring freely and discovering unexpected insights.

The main purpose of visualization is generating insight. Discovering new things that previously went unnoticed, or looking at something familiar from a new angle. Thus, a visualization evaluation method should be able to determine how well visualizations achieve this goal. So, a measure of effectiveness of a visualization system can be its capability to generate unexpected new insights, beyond performing predefined tasks. Removing the predefined tasks provides a much broader view of insight abilities of visualizations. Therefore, instead of instructing users on exactly what insights to gain, researchers should allow users an open-ended experiment and see what insights users gain on their own.

In this thesis, for evaluating the effectiveness of our proposed visualization tool and examining how successful it is in generating insights, we did actual exploratory analysis using our tool. We performed some real case studies and used our tool to explore the results and see what new and unexpected insights we can gain from the tool that is very hard (if not impossible) to find out those insights from the raw data. The discussion about insights achieved using our tool is given in the subsequent chapters.

2.4 Conclusion

In this chapter we first introduced the preliminaries of multi objective optimization. Then we discussed that the multi-objective design space exploration problem can be considered as two distinct sub-problems: 1) how a single design point can be evaluated with respect to the design criteria and 2) how the design space can be searched during the exploration process. The latter issue arises since in real-scale problems the design space is too large to be explored in an exhaustive manner. In this work we use the Sesame system-level simulator for evaluating the design points and a MOEA is used to search and explore the design space for optimum solutions using only finite number of design point evolutions.

By utilizing an MOEA for multi-objective design space exploration of embedded systems, on one hand, we should define the specification of the DSE problem (i.e. application model, architecture model and design criteria) and on the other hand we should set the MOEA parameters such as selection method, mutation rate, crossover rate, etc. Setting the MOEA parameters properly, according to the DSE specification, can minimize the exploration time while achieving better results. Then, it is needed to define a way of encoding the DSE problem (e.g. mapping process networks onto heterogeneous multiprocessor architecture) as a multi-objective optimization problem, known as individual encoding. Therefore, each individual in the MOEA (which can be a string of numbers) represents a design point in the design space (e.g. possible mapping). The workflow of using MOEAs in multi-objective DSE of embedded systems is illustrated in Figure 2.12.

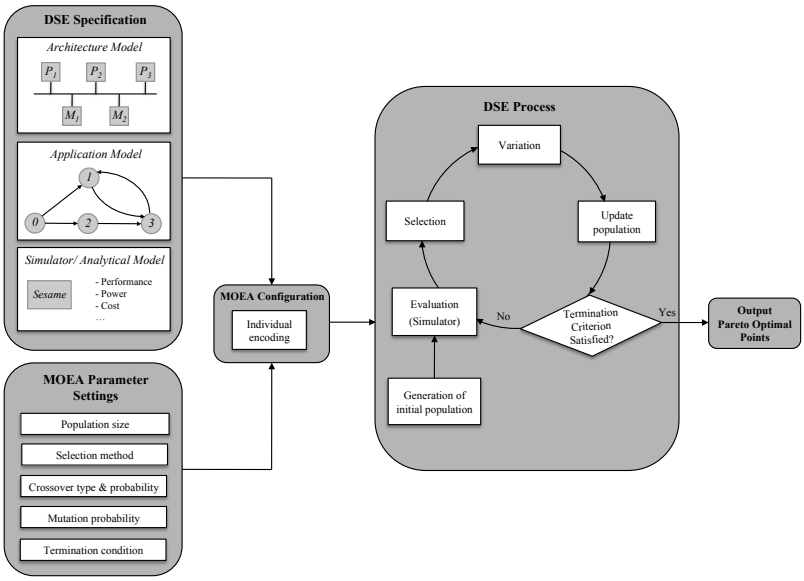


Figure 2.12: Workflow of using MOEAs in multi-objective DSE of embedded systems

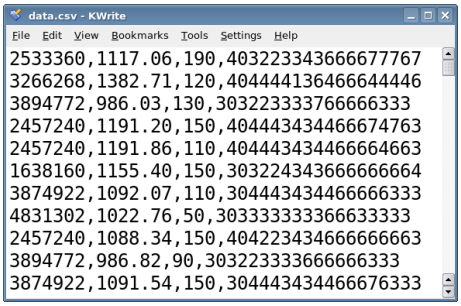


Figure 2.13: Example of raw data generated by a MOEA

As the searched design space still is vast, interpreting all evaluation data and understanding how the MOEA searches through or prunes the design space is cumbersome. Such analysis is, however, essential to the designer as it provides insight into the ”landscape” of the design space (e.g., indicating which design parameters are more important than others).

To illustrate the need for good analysis tools, Figure 2.13 shows a sample of raw data generated by an MOEA. Here, each row represents an evaluated design point in which the values of objectives (processing time, energy consumption and cost) and the individual encoding are comma separated. The way that application tasks and their communications are mapped onto the architecture components is encoded in a string of digits, which is called the individual. It is evident that interpreting and analyzing the evaluated data in this format is not possible.

Therefore, we have proposed new visualization techniques (explained in the next chapters), to understand how an evolutionary algorithm searches the design space, where the optimum design points are located, how design parameters influence each objective, and understand the correlations among multiple objectives. The main challenge that needs to be addressed by such a visualization environment is how the raw data (as illustrated in Figure 2.13) can be represented in a visual form such that it is possible to analyze the data from different perspectives and for various aspects and enables users to gain new insights into the exploration results.

3

Visualization of Multi-Objective Design Space Exploration

System-level simulation frameworks that aim for early Design Space Exploration (DSE) create large volumes of simulation data in exploring alternative architectural solutions and mappings. Interpreting and drawing conclusions from these copious simulation results can be extremely cumbersome. In other domains that also struggle with interpreting large volumes of data, such as scientific computing, data visualization has become an invaluable tool to facilitate the data analysis. Such visualization is often domain specific and has not become widely studied and utilized for evaluating the results of computer architecture simulations. Here, results are usually still presented in a table or displayed in a 2D or 3D graphs and very little research has been undertaken in the use of visualization to support and guide DSE.

In this chapter, we introduce our proposed visualization techniques, which are specially designed for understanding the Multi-objective DSE process of embedded systems that are based on heterogeneous Multi-Processor System-on-Chip (MPSoC) architectures. All the developed visualization techniques are incorporated into a tool, called VMODEX[1].

We have two main motivations for visualizing the design space exploration process. The first one is to gain insight into the landscape of the design space. That is, understanding the characteristics of the optimum design points, the correlations among multiple design criteria such as performance, cost, energy consumption and so on, and the relationships between the design parameters and their effects on the criteria. These relations are often not linear and a small change may lead to a completely different result. However, our visualization tool allows designers to easily understand these relations. Our second motivation is to understand how the design space is searched by heuristic searching algorithms. Since in real problems the design space is extremely large, it is not possible to evaluate all the design points in this space. Therefore heuristic search techniques such as evolutionary algorithms are often used to prune the design space and trim down an exponential design space into a finite set

[1] Visualization of Multi-Objective Design spacE eXploration

of points, which are more interesting (or superior) with respect to the design criteria. Our tool enables designers to understand the dynamic search behaviors of such algorithms. For instance, it clearly shows which parts of the design space are not searched at all (no design point is evaluated there), which parts of the design space are searched more often by the MOEA (more design points are evaluated there), which parts of the design space are explored in the later generations, and it illustrates the progress of the searching algorithm in the design space during successive iterations.

The rest of this chapter is organized as follows. In Section 3.1 related work on visualizing the exploration results is discussed. Section 3.2 introduces techniques we have provided for visualizing multi-objective design space exploration. Section 3.3 presents a case study with a Motion-JPEG encoder application to illustrate the benefits of using visualization in the design space exploration process. The data from this case study are used in Section 3.4, which introduces several metrics and their visualization approaches for comparing different subspaces of the explored design space. Finally, section 3.5 concludes this chapter.

3.1 Related Work on Visualizing Exploration Results

Most of the work that has been performed on visualization of computer architecture simulations either focuses on educational purposes (e.g., [5–7]), tightly couples visualization to one particular architecture simulation environment (e.g., [8–11]), visualizes only one specific aspect of embedded applications such as memory access patterns [12], cache behavior [13–15] and data dependencies [16], or only provides some basic support for the visualization of simulation results in the form of 2D (and sometimes 3D) graphs (e.g., [17]). However, in the area of system-level design space exploration, little research has been undertaken on visualization of simulation results and the exploration process.

The work presented in [62, 63] provides advanced and generic visualization support, and tries to do so for a wide range of computer system related information. However, these visualizations are not necessarily applicable to computer architecture simulations and in particular to design space exploration, with its own domain-specific requirements. Vista [64] aims at visualization support for computer architecture simulations, but it does not target system-level simulations, which may have a serious impact on the scalability requirements of the visualization, nor does it address the needs for visualization from the perspective of DSE.

In [65], an interactive visual tool is presented to visualize the results from system-level design space exploration experiments. The simulation results are visualized using a coordinated, multiple-view approach, which enables designers to understand the information through different perspectives. It is possible to compare different design points with respect to various characteristics. But this tool does not provide any insight in the searching process as performed by a heuristic method (e.g. MOEA). For example, there is no way to find out which parts of the design space are not searched at all.

This tool is only useful in the case of small design spaces, which can be exhaustively searched.

There are only a few research efforts addressing the visualization of Evolutionary Algorithms (EAs). The most common method for analyzing how EA evolves is monitoring individual fitness values, and creating fitness-versus-generations graphs. Although such straightforward graphs show the quality of the solutions considered during search process, they can only provide a limited amount of information. Due to the large number of individuals in a population, it is not possible to display all of their fitness values. Therefore, displays are usually restricted to the best individual fitness and average population fitness for each generation. Genetic information regarding the position of the discovered solutions in the search space cannot be obtained from these graphs. Other approaches also usually use standard visual representations such as bar charts, line graphs, scatter plots, etc. [66–68]. These graphs can show only one feature of EAs and therefore multiple views are needed to comprehensively understand the EA process.

More complex techniques have focused on how to display the progress of the Multi-Objective Evolutionary Algorithm (MOEA) in design (parameter) space or objective space [69, 70]. Usually, they use 2D or 3D plots in which either variables or objectives are shown. Therefore, two separate views are needed to show the distribution of the solutions in both parameters and objectives spaces. Furthermore, due to the large number of dimensions in practical problems, techniques such as Sammon Mapping [71] should be used to transform higher dimensional search spaces into smaller ones.

Hart and Ross [72] proposed the GAVEL visualization tool that provides a means to examine how the genetic operations (crossover and mutation) assemble the optimal solutions, and a way to trace the ancestry of individuals. GAVEL only shows the information that is relevant to the formation of the best solutions. All individuals that do not play any part in propagating individual genes to the final optimal solutions are disregarded. So, it does not provide any information on how the problem space is explored and how an entire population behaves or changes across generations.

In [73], a visualization method is proposed, which shows the individual structure, the fitness function and the objective function in a matrix view. For each generation, an $n \times m$ matrix is used where n is the number of individuals in the generation and m is the number of bits in an individual. Each cell of the matrix is related to a bit of the individual. If the value of a bit is one the corresponding cell is represented in red. The blue cell indicates that the value is zero. The brightness of each row (individual) represents the fitness value. The hue of individuals is changed by their objective values. Although this visualization approach shows three different kind of information in a single view, it has some drawbacks. As they mentioned in their paper, it can be used for showing only binary-coded evolutionary algorithms. Furthermore, the structure of solutions in the parameter space is represented in their encoding format. So, understanding the correlation between the problem space and individual space is difficult. It is not easy to find out how EA explores the parameter space during its generations.

The multi-objective DSE visualization, which is presented in this thesis, is based on

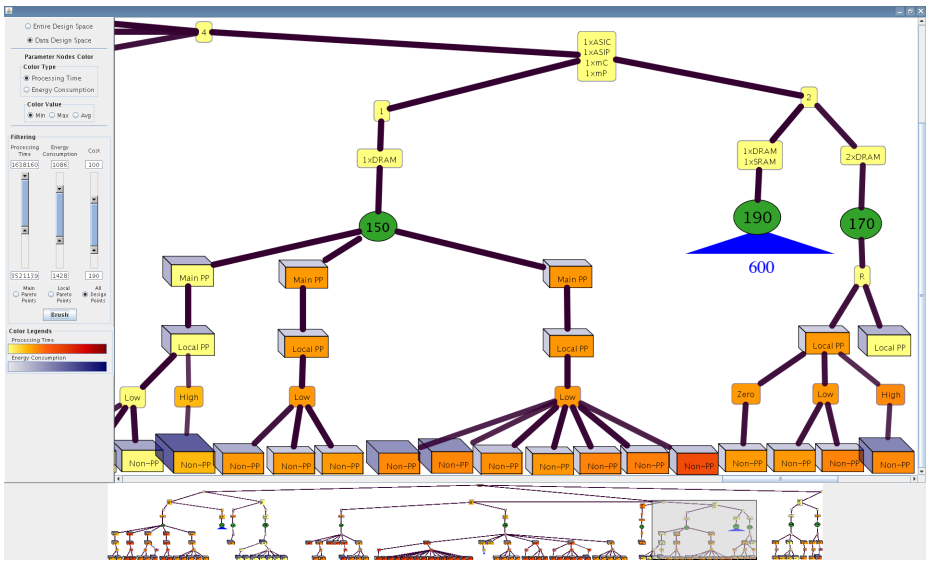


Figure 3.1: Screenshot of VMODEX visualization tool

tree visualization and enables us to visualize more than three dimensions as well as to see both the design parameters and objective values in a single view. Furthermore, it allows us to understand the dynamic search behavior in heuristic based design space exploration and gain insight into the landscape of the design space. In addition, several interactive capabilities are provided, which enable the designer to analyze the data and explore the search result from different perspectives and at multiple levels of abstraction in order to discover interesting and important features that may not be found just by looking at the raw data or by using the tradition 2D/3D graphs. To give a rough feeling of how such visualization looks like, Figure 3.1 shows a screenshot of our visualization application. Our visualization application is developed entirely in java and therefore is cross platform compatible.

3.2 Multi-Objective DSE Visualization

In this section, we introduce our proposed visualization techniques, which are specially designed for understanding the Multi-objective DSE process of embedded systems that are based on MPSoC architectures. The results of system-level design space exploration of computer architectures are used as the input data to the visualization system. For each design point, its parameter and objective values should be given. The parameter values define the characteristics of a multi-processor architecture instance (e.g. number of processors, types of the processors) and the objective values represent the performance of a design point with respect to the design criteria. The goal of our visualization tool is to help embedded systems designer to 1) gain insight

into the landscape of the design space, and 2) understand how the design space is searched by heuristic searching algorithms.

The multi-objective DSE visualization, which is presented in this thesis, is based on tree visualization. In comparison with the existing tools such as parallel coordinate, scatterplot matrix and influence explorer, our proposed DSE tree is more structured in terms of dominance relation between design points. In multi-objective optimization problems, the concept of dominance is generally used to compare the quality of different solutions and determine the trade-off among multiple objectives. As we will describe in the following sections, in the DSE tree, design points are distributed in the three levels (i.e. global Pareto, local Pareto, non-Pareto) based on their dominance relations with respect to each other. Thus, just by looking at the tree, one can understand how good a solution is in comparison to other solutions. However, in the existing tools, the dominance relation is not considered for representing the solutions. They just plot each individual design point independently from other points.

Furthermore, in our proposed DSE tree, the design points are clustered in a way that nicely matches with natural way of thinking of embedded systems designers. In this thesis, we refer to the clusters as subspaces. By modeling the design space as a tree, it is divided into exclusive subspaces, such that each subspace represents a unique instance of the architecture platform. On the other hand, solutions inside a subspace have exactly the same architecture components but the way that the application is mapped onto those components is different. This kind of clustering of design point is quite helpful for embedded systems designers to explore their design problems. They are mainly interested to investigate how objective values are being changed by modifying the mapping of applications within a unique architecture instance (solutions in the same subspace). Moreover, we have provided several techniques in our tool that allow designers to evaluate and compare the properties of the discovered design points in different subspaces (different architecture instances) from various perspectives.

Note that in this thesis, we use the conventional tree diagram to describe the concepts of our tree model of the design space. However, to more effectively utilize the screen space, other alternatives for representing the tree structures such as hyperbolic tree, cone tree, space tree, etc. could also be used.

3.2.1 Modeling the Design Space as a Tree

As it is conceptually shown in Figure 3.1, we model the design space as a tree. In this section, we explain how a design space can be modeled as a tree. Table 3.1 presents the set of mathematical symbols that are used for constructing the tree model of the design space. The tree has three segments: the parameters, architecture-dependent objectives and design points. Equation 3.1 formally describes tree segments.

$$T_{DS} = Seg_{par} + Seg_{arch-dep-obj} + Seg_{dp} \tag{3.1}$$

The parameters and design points segments are explained in this section while the architecture-dependent objectives segment will be described in the next section.

Table 3.1: Table of symbols for the tree model of the design space

Symbol	Definition
T_{DS}	Tree model of the design space
Seg_{par}	Parameters segment
$Seg_{arch-dep-obj}$	Architecture-dependent objectives segment
Seg_{dp}	Design points segment
$\|Seg\|$	Number of levels in a segment
LVL	Abbreviation for level
par	Abbreviation for parameter
n	Total number of parameters
DP	The set of all evaluated design points
GP	The set of global Pareto optimal points
LP	The set of local Pareto optimal points
NP	The set of non-Pareto optimal points
Sub	Abbreviation for subspace
k	Total number of subspaces in the T_{DS}
$map(p)$	The mapping of the application onto the architecture components for the design point p

Parameters Segment

In this segment, each level shows one parameter of the design space, such as the number of processors in the MPSoC platform. So, the number of levels in this segment is equal to the total number of parameters in the design space; formally:

$$\begin{aligned} &Seg_{par} = LVL_{par_1} + LVL_{par_2} + ... + LVL_{par_n} \\ &|Seg_{par}| = n \end{aligned} \tag{3.2}$$

For better understanding the process of modeling the design space as a tree, an example is shown in Figure 3.2. In the tree illustrated in this figure, the design space has four parameters thus there are four levels in the parameters segment, which are:

1. *Number of processors level:* this level shows the number of processors (independent of their type), which are allocated for executing the application tasks. In this particular example, the desired maximum number of processors is two and therefore there are two nodes at this level indicating one or two used processors.

2. *Processors type level:* nodes at this level represent different combinations of processor types. Each node becomes a child of the node at the previous level that shows the number of processors used in its combination. In this example, the platform architecture consists of one Application Specific Instruction Processor (ASIP) and two microprocessors (mPs). Note that only those combinations are shown, which lead to the platform instances that are capable of executing the

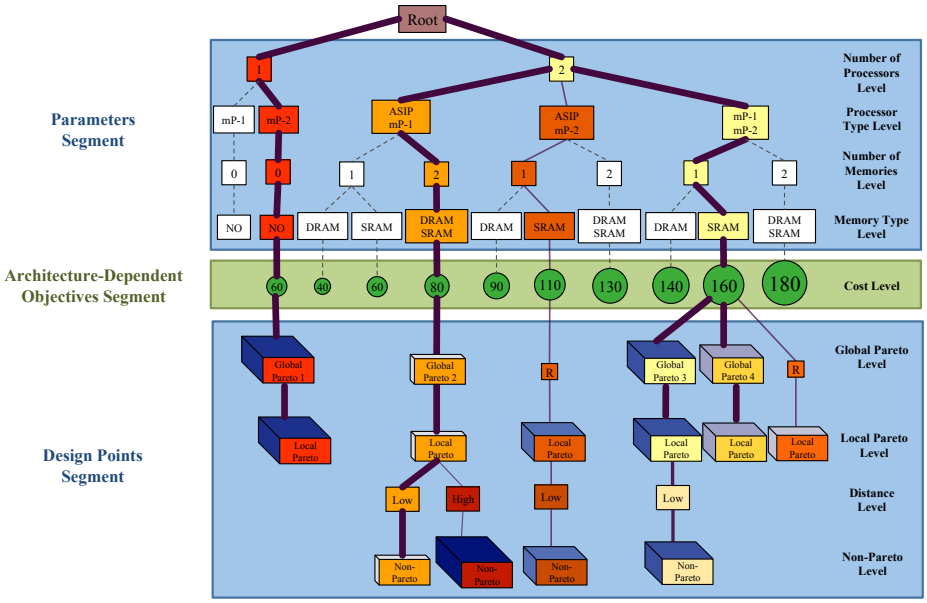


Figure 3.2: Modeling the design space as a tree

application (feasible combinations). For instance, in this example, using only one ASIP is not sufficient for executing the application and thus there is no node in the tree that indicates one ASIP.

3. *Number of memories level:* this level shows the number of memories used in a platform instance.

4. *Memory types level:* at this level, the type of memories is shown. In this example, there are two types of memory: one Static RAM (SRAM) and one Dynamic RAM (DRAM).

By modeling the design space as a tree, there is no limitation on the number of design variables as each parameter is located at one level of the tree. It should be mentioned that, in principle, the designer has total freedom of ordering the parameters in the levels of the tree. However, putting more important parameters higher up in the tree facilitates the information organization in such a way that it produces sub trees, which are more likely to show a better view of the design space characteristics. Because the more important design points (according to the design parameters) are clustered in only one sub tree, the designer can easily select that sub tree to investigate and compare these design points. On the other hand, by putting more important parameters down in the tree, the design points with the same parameter are distributed in several sub trees.

Design Points Segment

this segment includes the design points searched by the MOEA. Here, a design point is defined as a specific instance of the architecture platform as well as a task and communication mapping. Each point is shown as a node, which is a child of its corresponding architecture. This means that, for each solution, its parents at the previous levels show its design parameters. For instance, the solution labeled by "Global Pareto 2" has the following architectural components: two processors, of which one is an ASIP and the other one is mP, and two memories, of which one type is DRAM and another type is SRAM. Design points are distributed in three levels: global Pareto, local Pareto and non-Pareto. There is also another level in this segment, called distance level, which is used for categorizing the non-Pareto points based on their distance from parents. Thus, the design points segment consists of four levels, as follows:

$$\begin{aligned} &Seg_{dp} = LVL_{gp} + LVL_{lp} + LVL_{dis} + LVL_{np} \\ &|Seg_{dp}| = 4 \end{aligned} \tag{3.3}$$

Global Pareto Level This level shows the global Pareto points found by the MOEA. The solutions at this level are better than all other solutions in the entire design space; formally:

$$\begin{aligned} &LVL_{gp} = \{p \mid p \in GP\} \\ &GP = \{p \in DP \mid \nexists p' \in DP : p' < p\} \end{aligned} \tag{3.4}$$

The solutions in the global Pareto optimal set has the following two properties:

1. They are non-dominated with respect to each other (no one is absolutely better than another one); formally: $\forall\, p, p' \in GP : p \sim p'$
2. Each point that is not part of the global Pareto set, is dominated by at least one global Pareto point; formally: $\forall\, p \in (DP - GP)\, \exists\, p' \in GP : p' < p$

In this thesis, we simply refer to the global Pareto optimal solutions as Pareto optimal solutions.

Local Pareto Level By modeling the design space as a tree, it is divided into exclusive subspaces. Each subspace represents a unique combination of design parameters (in our case, a unique instance of the architecture platform). On the other hand, solutions inside a subspace have exactly the same architecture components (have the same parents at the parameter levels) but the way that the application is mapped onto those components is different. Figure 3.3 illustrates the concept of subspaces. It shows the same tree as Figure 3.2. However, in this figure, the four subspaces of the design space that contain some evaluated design points are indicated by contour lines and are labeled from Sub_1 to Sub_4. For instance, all design points in the subspace Sub_4

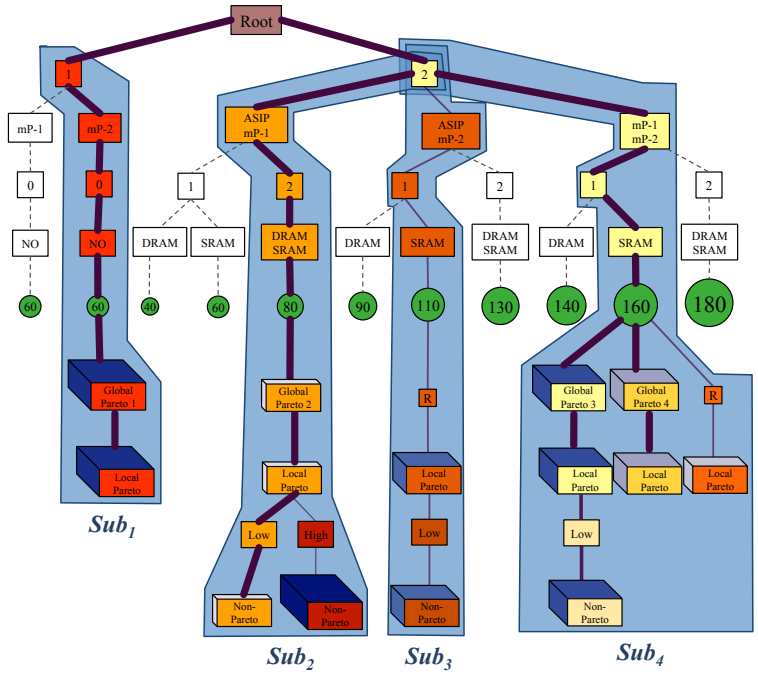


Figure 3.3: Illustration of the subspaces of the design space

have two microprocessors and one SRAM memory in their underlying architecture. However, the binding of application tasks and channels to these resources is different. In total, there are 11 subspaces in the example shown in Figure 3.3. Actually, The number of nodes at the last level of the parameters segment (memory type level in Figure 3.3) indicates the number of subspaces. Thus, we can consider the tree model of the design space as a union of several subspaces:

$$T_{DS} = \bigcup_{i=1}^{k} Sub_i \tag{3.5}$$

Where k is the total number of subspaces (in our example $k = 11$).

For each $p, p' \in Sub_i$ following conditions are true:

1. $\forall\ j \in \{1, 2, .., n\} \quad par_j(p) = par_j(p')$
2. $map(p) \neq map(p')$

At the local Pareto level, the local Pareto points are shown. In each subspace, the Pareto optimal solutions are called local Pareto. Thus, a design point is called a local Pareto point if within the design points with the same architecture components (same

allocation) but with different mappings (different binding), there is no point dominating that one. So, it is an optimal solution with respect to a specific architectural instance. However, in the entire design space, a design point might exist which dominates the local Pareto point. Equation 3.6 gives a formal definition of the local Pareto concepts:

$$\begin{aligned} LVL_{lp} &= \{p \mid p \in LP\} \\ LP &= \bigcup_{i=1}^{k} LP_i \\ LP_i &= \{p \in Sub_i \mid \nexists\, p' \in Sub_i : p' < p\} \end{aligned} \tag{3.6}$$

It is clear that all the global Pareto points are local Pareto points as well. However, not all the local Pareto points are global Pareto points and therefore we use a relation node at the global Pareto level to make a connection between them and the previous level. These nodes are labeled with R in Figure 3.2. Equation 3.7 describes this relation between the global Pareto and local Pareto sets:

$$\begin{aligned} p \in GP &\Rightarrow p \in LP \\ p \in LP &\nRightarrow p \in GP \end{aligned} \tag{3.7}$$

Non-Pareto Level All the other design points are placed at the non-Pareto level. Each non-Pareto point is dominated at least by one point in the local Pareto set of its corresponding subspace (LP_i); formally:

$$\begin{aligned} LVL_{np} &= \{p \mid p \in NP\} \\ NP &= \{p \in DP \mid \exists\, p' \in LP_i : p' < p\} \end{aligned} \tag{3.8}$$

Each non-Pareto point becomes a child of a local Pareto point, which dominates it. If a design point is dominated by more than one local Pareto point, we calculate the Euclidean distance, in the objective space, between the dominated point and each dominating local Pareto point and the design point becomes the child of the local Pareto point with the smallest distance. A smaller distance means that the points are more similar according to the objectives.

Distance Level For easier interpretation and better analysis of the design points, the children of a local Pareto point are categorized into two groups according to their Euclidian distance from their parent. If the distance between a non-Pareto point and its corresponding local Pareto point is more than a certain threshold (determined by the designer), it becomes a child of a "High" distance node, otherwise it becomes a child of a "Low" distance node. Thus, there are two types of nodes at the distance level; formally:

$$LVL_{dis} = \{d \mid d \in \{Low, High\}\} \tag{3.9}$$

3.2.2 Showing Objectives

Various features can be considered as objectives for design space exploration of embedded systems, such as processing time to complete a particular task or application, power consumption, energy consumption, architecture cost, latency, utilization, temperature, physical size, weight and so on. We classify objectives, which are considered in the DSE process, in two groups: primary objectives and secondary objectives. The primary objectives are directly used in optimization process and depending on their values the optimization algorithm walks through the design spaces to find optimum design points with respect to the primary objectives. The secondary objectives are not considered as optimization goals. However, they provide supportive information and help designer to do further analysis on the characteristics of design points.

The primary objectives are directly shown in the DSE tree and therefore just by looking at the DSE tree the designer can see their values. However, the secondary objectives are shown just after the designer requests to see them. If the designer is interested to know about the secondary objectives for some specific solutions, it is possible to select those solutions to see their secondary objective values.

Showing Primary Objectives in the DSE Tree

For representing the primary objectives in the DSE tree, we have divided these objectives into two groups:

1. Objectives that are only dependent on the architectural components, denoted as $obj_{arch-dep}$
2. Objectives that are dependent on the mapping, denoted as $obj_{map-dep}$

Those primary objectives that are only dependent on the architectural components, not on the mapping decision are shown in the architecture-dependent objectives segment ($Seg_{arch-dep-obj}$). An example is the architecture cost as shown in Figure 3.2. These objectives can be computed after the parameters segment, since all components are known. Furthermore, all design points with the same architecture have the same value in terms of these objectives. Therefore, we add an extra segment between the parameters and design points segments, which shows the values of architecture-dependent objectives for different architectures. The definition of the architecture-dependent objectives segment is as follows:

$$Seg_{arch-dep-obj} = LVL_{obj_{arch-dep_1}} + LVL_{obj_{arch-dep_2}} + ... + LVL_{obj_{arch-dep_{m_1}}}$$
$$|Seg_{arch-dep-obj}| = m_1 \tag{3.10}$$

Where m_1 is the number of architecture-dependent objectives. In Figure 3.2, the cost is shown with a different shape; a circle, since it is an objective and not a design parameter. For a better view, the size of the circle becomes bigger as the cost increases.

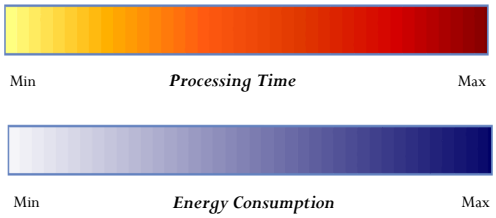


Figure 3.4: Color legends

In the case that more primary objectives exist, which are only dependent on architectural components, such as weight, physical size, etc., extra levels can be added in the segment $Seg_{arch-dep-obj}$ in which each level shows one objective. For better distinguishing different objectives, each one can be shown with a different color and shape, like rectangle, trapezoid, pentagon, etc. In each subspace, the values of architecture-dependent objectives for all design points are the same. For instance, in Figure 3.3, the cost of all design points in Sub_4 is 160. Formally, For each $p, p' \in Sub_i$ the following condition is true:

$$\forall j \in \{1, 2, .., m_1\} \quad obj_{arch-dep_j}(p) = obj_{arch-dep_j}(p')$$

Those objectives that are dependent on the mapping (binding application tasks and channels to the hardware components) are shown in a design point node. In Figure 3.2, there are two primary objectives that are dependent on the mapping: the processing time (i.e. the time needed to execute the given application) and energy consumption of the whole system. The color of the node itself represents the processing time. Colors are varied from yellow to red with all color grades in between. Nodes with the lowest processing time are yellow and nodes with the highest processing time are red. The size and color of the third dimension of a design point node shows the energy consumption. As the energy consumption increases, the size of the third dimension becomes bigger and its color becomes darker. The color legends for the processing time and energy consumption are shown in Figure 3.4.

Parameter nodes, however, do not represent single design points and therefore do not have the direct notion of processing time or energy consumption. For this reason, there are some options to color the parameter nodes: based on the average, minimum, or maximum of either processing time or energy consumption of the design points in their sub trees. The color of parameter nodes that have no data node (i.e., do not have any DSE data) is white. In Figure 3.2, the minimum processing time is chosen for coloring parameter nodes.

It should be mentioned that although we only show three objectives (architecture cost, processing time and energy consumption) in this thesis, VMODEX is able to easily visualize more than three objectives. For those objectives that are only dependent on the architectural components, extra levels can be added in the architecture-dependent objectives segment. Showing more objectives that are dependent on the mapping is

(a) (b)

Figure 3.5: An example of representing solutions with 6 objectives

also easy. Each node has some attributes like shape, orientation, size, color, transparency, texture, border, glow, etc. Each attribute can be assigned to one objective. For the purpose of illustration, Figure 3.5 shows an example of representing a solution with six objectives. In this figure, texture density indicates average utilization (denser texture means higher utilization), the size of the glow shows the temperature (bigger glow means higher temperature) and the size of the trapezoid indicates the weight (bigger trapezoid means heavier weight). The other three objectives are the same as Figure3.2. Note that the examples shown in Fig. 5 would become the tree nodes in our DSE tree (For the case that we have 6 objectives). Since in the DSE tree the architecture-dependent objectives (e.g. cost and weight) are shown as separate nodes in the segment $Seg_{arch-dep-obj}$, the corresponding nodes are drawn above the 3D rectangle. Figure 3.5(a) represents a solution that is superior to the solution shown in Figure 3.5(b) with respect to the all six objectives. Thus, the solution in Figure 3.5(a) has a lower processing time and energy consumption, and its average utilization is higher. It also produces less heating and it is cheaper and lighter.

As we described above, our visualization tool is extendable to show more than three objectives. However, multi-objective design space exploration problems are usually based on only two or three objectives, typically performance, cost and power. The maximum number of objectives that we found in related work is six [74].

Showing the Mapping Decision

In the Sesame simulator as well as in many other system-level simulation frameworks, the application behavior is modeled as a process network. A process network is a computational model of the application and uses a directed graph notation, where each node represents a process and each edge represents a one-way (FIFO) communication channel between two processes. Figure 3.6(a) represents an example process network graph, which has five processes (A-E) and six communication channels (1-6).

We visualize the process network graph in a way that shows the mapping decisions as well. That means that it shows how the application is being mapped to the underlying architecture both in terms of processes and communication channels. The shape and the color of each node in the graph represent the type of the processor executing the

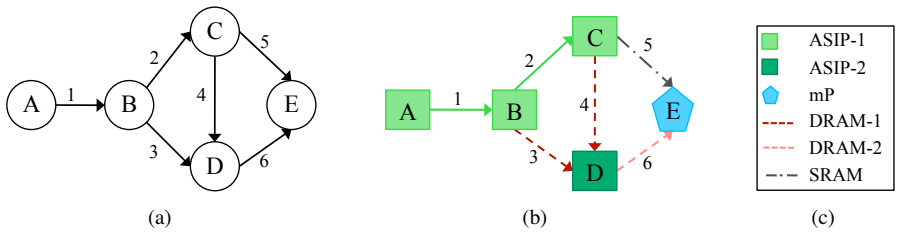


Figure 3.6: (a) An example of process network, (b) Mapping decision visualization, (c) Mapping legend

corresponding process. For example, a green rectangle for one processor type and a blue pentagon for another type. If there are multiple processors of the same type in the platform architecture, then they are differentiated using different variants of the same color such as light green and dark green.

If two communicating processes are mapped onto the same processor, then their communications are done internally and therefore communication channel(s) between them are mapped onto the processor in question. In the process network graph these internal communications are represented by a solid line with the same color as the corresponding processor. In the case that a channel is mapped onto an external memory, a dashed line is drawn. The color of the line and style of the dash represents the memory type. Memories with the same type are shown by the same dash style but with different variants of the same color.

Figure 3.6(b) shows how our visualization model shows the process network graph from Figure 3.6(a). As can be seen in this figure, processes A, B, C and channels 1 and 2 are mapped to the same processor (ASIP-1). Process D is executed on the same processor type but on a different processor as process A (ASIP-2). The type of the processor executing process E is different from the others since it is shown with a blue pentagon (mP). Channels 3,4,5 and 6 are mapped to memories (not processors) as they are shown with dashed lines. Channels 3 and 4 are mapped to the same memory (DRAM-1). Channel 6 is mapped to another memory but with the same type (DRAM-2) and Channel 5 is mapped onto a different memory type because it has a different dash style and different color (SRAM).

Note that the colors assigned to each processor type or memory type in the visualization of the mapping decision are the same as colors used in the visualization of utilization, latency and processes execution time (these visualizations will be explained later). For instance, in all of them the ASIP processor is shown by a green color. Thus, these visualizations are consistent with each other and the designer can easily understand the effect of different mapping decisions on the utilization of architecture components as well as processes execution times and read/write latencies.

Showing Secondary Objectives

Since the secondary objectives are not used in the optimization process and do not have any effect on the exploring procedure, we do not show them directly in the DSE tree. However, if the designer is interested to know about the secondary objectives for some particular solutions, it is possible to select those solutions to see their secondary objective values. The secondary objectives provide additional information and help the designer to do more in-depth analysis on the characteristics of design points. VMODEX is designed to show various properties of design points, which are considered as secondary objectives. Thus, the designer is able to look at the evaluated data from different perspectives and thereby gets a comprehensive view on the problem under study. Furthermore, for better understanding and easier analysis, we propose several visualization techniques for showing the secondary objectives. In the following subsections the provided secondary objectives and their visualization methods are described.

Showing Utilization For better understanding the utilization of each separate hardware component and also for easier comparison between different design points, we have visualized the utilization property. Here, we define utilization as the percentage of the time that a hardware component was busy. For visualizing the utilization, the platform architecture is shown as a directed graph in which each node represents an architectural component and the edges show connectivity between components.

Each node (component) is filled with its corresponding color (see subsection 3.2.2) in a way that the size of the colored part represents the percentage of the time that the corresponding component was busy. The components in the platform architecture that are not allocated are shown by gray borders. The average utilization is written at the bottom of the visualization.

Figure 3.7 shows an artificial example of utilization visualization for a platform architecture consisting of two Application Specific Instruction Processors (ASIPs), two microprocessors (mPs), one Static RAM (SRAM) and two Dynamic RAMs (DRAMs). In this example, the mP-1 is not allocated since it is shown by gray border. The utilization of ASIP-1 and DRAM is 100% while for the Bus it is almost 75% and for the other components it is less than 50%. By visualizing the utilization, besides understanding the utilization of each architecture component individually, the designer is able to see how these components are connected together, operate with each other and which resources are shared.

Showing Processes Execution Time VMODEX also allows to modify the mapping visualization in such a way that it shows the processes execution time as well. Thus, it also shows the amount of time that each process was executed by the processor onto which it has been mapped. In the mapping visualization, the shape and color of each process in the process network graph represent the type of the processor executing the corresponding process. To add information about how long a processor was busy with

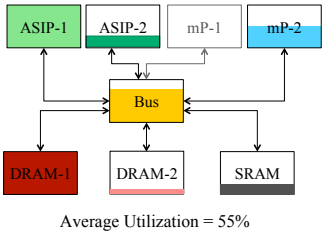


Figure 3.7: Utilization visualization

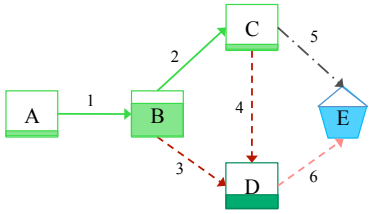


Figure 3.8: Visualization of processes execution time

executing a process, instead of fully coloring the process nodes in the graph, the size of the colored part represents the percentage of time that its assigned processor was busy executing it. Figure3.8 shows the same mapping decision as shown in Figure3.6(b), but the processes execution time information is added. From this figure we can see that processor ASIC-1 was executing process B in most of its busy time, while executing process A and C took relatively a small amount of time. The processes execution time visualization enables designers to explore the effect of different mappings on the execution time of processes.

Showing read/ write latency Similarly, VMODEX is also capable to show the amount of time that each process is waiting for read and write communications. Since in process network graph the communication channels between processes are shown, we visualize it in a way that shows the read and write latencies as well. The color-coding, from blue to red, is utilized to represent the waiting time. In Figure3.9(a) the color legend for latency is shown. In the process network graph, each process node is divided into two halves. The color of the top part shows the write latency and the color of the bottom part represents the read latency. By clicking on each part, the exact value of latency is shown. The mapping information is also added to the latency visualization as well. Therefore, the designer can easily investigate the effect of different mappings on the latency. The shape and color of the third dimension of each process in the graph represent the type of the processor executing the corresponding

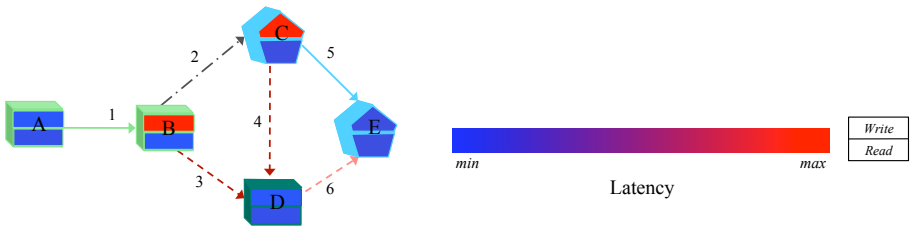


Figure 3.9: Latency Visualization

process (similar to the mapping visualization). Figure 3.9(a) shows an example of latency visualization. In this figure, processes B and C have high write latencies since the color of their top parts are red. By including mapping information in the latency visualization we can understand that the reason of these high latencies is because of mapping channels 3 and 4 on the same memory (both of them are shown by dashed red lines). Thus, processes B and C are competing with each other to write to the shared memory. For the other processes both the read and write latencies are relatively low.

Showing Processing Time of Processor Operations Basically, processors perform three operations: read, write and execution [23]. There is an option in VMODEX to see the percentage of time that a processor was doing each operation separately. To show this, for each processor in the architecture platform, a stacked bar chart is drawn. Each operation is shown as a stack in the bar with a different color. The stack height represents the percentage of time that the processor was performing the corresponding operation. Figure 3.10 represents an example of visualizing the processing time of different processor operations for a platform architecture consisting of two ASIPs and one mP.

Showing Generation Numbers In some cases, the designer wants to know what interesting design points are evaluated in which search generations. Therefore, we have developed a method in VMODEX that allows the designer to easily find out this kind of information. During the process of design space exploration using an MOEA, some design points that are near to the optimal solutions may be regenerated in different generations. There is an option in the VMODEX user interface to show the generation numbers. By selecting that option, for each design point node in the DSE tree, a hexagon will be drawn at the upper left corner of the node. The number of the last/first (chosen by the designer) generation in which the corresponding design point was generated is written inside the hexagon.

Moreover, the designer is also able to select a specific design point and see all the generation numbers in which the design point was evaluated. Figure 3.11 shows the

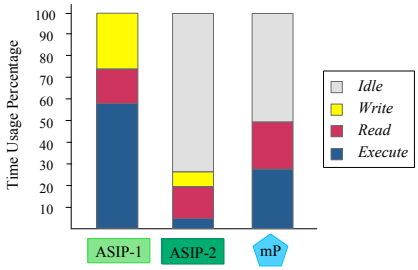


Figure 3.10: Visualization of processing time of processor operations

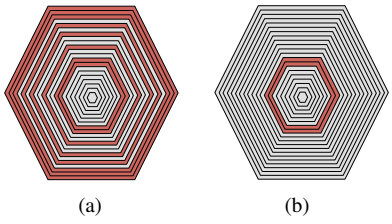


Figure 3.11: Visualization of the generation numbers

visualization approach for showing the generation numbers. For each generation, a hexagon is drawn. The size of the hexagons increases from the first to the last generations. To save space, these pentagons are nested together. If within a generation the selected design point is found, then the color of the pentagon representing that generation is red. Otherwise, a gray pentagon is drawn. Figure 3.11(a) shows the situation that the corresponding design point is close to the optimum. Since it is regenerated in many search generations during the entire search. But in Figure 3.11(b) the corresponding design point is generated in only two generations. This indicates that the design point is far from the optimal solutions and after a few generations it is not regenerated any more.

In the evolutionary algorithms, we expect better solutions at the later generations, as they gradually converge to the optimal solutions during successive generations. So, the later generations are more important. In our proposed visualization, we emphasize the importance of later generations by increasing the size of their corresponding hexagons.

Showing Exact/Normalized Objective Values Instead of showing objectives with visual variables (color and size), VMODEX can also show the exact values of processing time and energy consumption. It also represents the normalized value of the objectives. We normalize objective values to make them scale independent. At the end of normalization, all design points get values in the range [0, 1] for their objective values. Before normalization, it is not possible to compare e.g. processing time and energy consumption with each other since they have different magnitudes. However, after normalization, comparing them is possible. Furthermore, for calculating the Euclidean distance between any two solutions (in the objective space), the normalized objective values are used (see Chapter 2, Section 2.1).

3.2.3 Edge Visualization

Edge visualization helps designers to navigate through the DSE tree and easily find more important parts. One feature of the design points is chosen as an importance factor and then the tree edges are visualized according to that factor, as follows:

- A minimum and maximum edge width is defined, and this range is linearly mapped against the range of importance factor values. For example, the following linear mapping can be used:

$$\frac{\Delta IF}{\Delta Width} = \frac{IF^{max} - IF^{min}}{Width^{max} - Width^{min}} = \frac{IF(e) - IF^{min}}{Width(e) - Width^{min}}$$

$$Width(e) = Width^{min} + \frac{\Delta Width}{\Delta IF} \times \left(IF(e) - IF^{min}\right)$$

  Where IF^{max} and IF^{min} are respectively the maximum and minimum values of the importance factor, and $Width^{max}$ and $Width^{min}$ are respectively the

maximum and minimum widths that are defined. $IF(e)$ denotes the importance factor of the edge e and $Width(e)$ indicates the appropriate width for representing the edge e. Wider edges lead toward more important subtrees. The effect is a bit like a network of roads, in which the more important roads are wider.

- A specific color with various saturations is chosen. Similar to the line width, a linear mapping is done between the maximum and minimum saturation and the importance factor values range. Darker edges represent more important parts and lighter edges show less important subtrees.

In VMODEX, two importance factors are defined: minimum Euclidian distance and last generation number. In the following we explain these factors.

Minimum Euclidian Distance For each design point, the Euclidian distance (in the objective space) between a solution and the nearest global Pareto optimal point is calculated. A smaller distance indicates that the solution is closer to the optimal solutions and therefore is better. Thus, in the DSE tree, the edges in the path from the root to the global Pareto optimal points are the thickest and darkest since the distance is zero (see Figure 3.2). As the distance increases the edges become thinner and lighter. In this manner, just by looking at the DSE tree, the designer can easily determines which parts of the design space contain optimal and near optimal solutions and which parts contain solutions that are far away from the optimal solutions.

Last Generation Number The number of the last generation in which a design point is evaluated can be considered as an importance factor. As the MOEA gradually converges to a set of Pareto optimal points, we expect better design points in the later generations. The edge visualization can show the progress of the searching algorithm in exploring and covering the design space during its generations. The edges with a higher generation number in their subtrees (i.e. the design points in their subtrees are evaluated in the later generations) are thicker and darker. As a result, the paths from the root to the last generated data nodes are the darkest and thickest paths.

As the importance factors are not applicable for edges that have no data nodes in their sub tree, these edges are shown by gray dashed lines.

3.2.4 Visualization of the Design Space Coverage

In VMODEX, we provide a technique to show how a heuristic searching algorithm walks through the design space and accesses new parts of the design space during its exploration process. To do this, a color-coding scheme is used to color the parameter nodes based on the generation number in which a design parameter is explored for the first time. Thus, the color of each parameter node represents the first generation that a design point, containing that parameter, is evaluated. On the other hands, for each parameter node, the number of the generation in which the first design point is

added to its subtree, is used for coloring the node. For visualizing the design space coverage, the green color with variations in lightness and saturation is used, such that the color of parameter nodes, which are searched in the earlier generations, is light green. The color of nodes becomes darker as they are investigated at the later generations. Therefore, the designer can easily see how a searching algorithm covers different parts of the design space during its successive iterations. In Section 3.3.1, an example of this visualization is shown for our case study results (Figure 3.15). In Figure 3.15, the color legend for visualizing the design space coverage is also shown.

3.2.5 Interactive Exploratory Techniques

User interaction plays an essential role in the effective visualization of data and information. It allows users to explore and work with the data actively to investigate the data from different perspectives. Interactive visualizations are very useful in analyzing data and can provide new insights that could not be obtained using static graphics or statistical methods. Interactivity means that users cannot only observe the visualization but moreover play with it. The visualization changes in response to the users actions and enable users to create customized views that are more interpretable and informative to be able to find hidden patterns in the data and complex relationships among variables. Colin Ware [75] notes,

> *"The best visualizations are not static images to be printed in books, but fluid, dynamic artifacts that respond to the need for a different view or more detailed information"* (p. 385).

Shneiderman [76] identified two aspects of visualization technology that should be considered for visualizing information in the most comprehensive and efficient way. One focuses on mapping abstract information to a visual representation and the other provides interactions between the visualization system and the users for more effective data exploration and help users to deal with large volumes of information. A user cannot be expected to deal with a single image consisting of thousands of information elements. Therefore, we have provided a set of interaction techniques in VMODEX to help designers to handle large design spaces and to analyze the data and explore the search results from different perspectives and at multiple levels of abstraction in order to find out interesting and important features that may not be found just by looking at the DSE tree visualization.

Zoom

Zooming and scrolling are traditional tools in visualization; they are quite indispensable when large trees are explored. In our visualization tool, besides normal zooming and scrolling, we provide two extra zooming features for augmenting the exploration process: *bird view* and *satellite view*.

Bird view provides overview (context) and detail (focus) on the same screen by allowing users to view items at different scales. The aim of bird view is to enlarge items closer to the users point of interest while compress the objects that are farther away from the interested region. Bird view is a window moving with the mouse-pointer and works like a magnifier. That means, those parts of the main scene that are under the bird view window are shown in enlargement with a higher zoom factor. So, by simply hovering over the DSE tree with the mouse-pointer, it is possible to zoom in on an area of interest to see its details without losing the global context. Bird view is helpful when the tree is big and the user zooms out to see an overview of the entire tree in one scene and still wants to see the details of some specific nodes such as labels.

Satellite view, shown at the bottom of Figure 3.1, gives an overall, smaller scale view of the entire scene, which allows the user to navigate quickly across the view. It also enables the user to zoom in on certain parts of the scene to focus on certain nodes in the tree without losing track of the position in the entire scene.

Hiding Nodes

In VMODEX, two options are provided to reduce the number of visible nodes in the DSE tree in order to make its size smaller for better and more effective exploration. It should be mentioned that by hiding some nodes, we recalculate the location of visible nodes in the tree to optimize their fit on the screen. This means that, the empty spaces from the hidden nodes are occupied by the visible nodes. Note that the order of the visible nodes in the tree will not be changed. So, the structure of the tree remains the same.

Hiding Sub Trees without Exploration Data Since we use heuristic search techniques, some areas of the design space may not be visited by the searching algorithm (e.g., they are not interesting enough). So, we do not have any evaluated design points for those parts. In VMODEX it is possible to hide the sub trees of the nodes that have no evaluated data. This way, the designer can focus on the explored sub trees, which are more important. However, if the designer is interested to see the entire design space (both evaluated and not evaluated parts), there is another option in the tool that shows all parts of the design space. In this case, the color of parameter nodes that are not visited by the searching algorithm is white. So, the designer is able to recognize which parts of the design space are not searched during the exploration process.

Hiding Uninteresting Sub Trees If the designer is not interested in some parts of the tree, then he is able to hide them in order to make the tree smaller and pay more attention to other nodes. By double clicking on a node, its sub tree collapses (becomes invisible) and a blue triangle appears at the bottom of the node specifying that the children of the node are hidden. The size of the triangle represents the size of the sub tree. The bigger the triangle is, the more nodes in the sub tree exist. By double clicking again, the sub tree expands (becomes visible) and the blue triangle is removed. If the

designer is interested to know the exact number of nodes in a hidden sub tree, by clicking its corresponding triangle the amount of nodes is shown. In Figure 3.1, the blue triangle at the bottom of the cost node "190" indicates that its corresponding sub tree is hidden, which contains 600 nodes (the number written at the bottom of the triangle).

Filtering

The filtering option in VMODEX allows designers to filter out parts of the solutions, which are not required for further investigation, and view only preferred design points. Therefore, the designer can focus on the more interesting design points and find the similarities/dissimilarities between them. Various kinds of filtering are available based on different designer preferences: 1) primary objective values, 2) design parameters, 3) distance from the Pareto optimal set and 4) secondary objective values. The combinations of different filtering approaches are also possible. The filtering results can be shown in three ways: view all design points that fall within the filtering conditions or only see local Pareto optimal points or only global Pareto optimal points.

Filtering Based on the Primary Objective Values Primary objectives are those objectives that participate in the optimization process and based on their values, the searching algorithm explores the design space. In some cases, the designer wants to consider only design points with some specific objective values (e.g. design points with the best processing time). The value of each objective is controlled by a range slider bar, in which the designer can set upper and lower limits on that objective. Design points with objective values inside the selected ranges are visible and the others become invisible. Therefore, the designer has the ability to view only design points with preferred objective values and find out the relationships between them.

Filtering Based on the Design Parameters VMODEX allows designers to hide the parts of the design space that are not being considered for further analysis to pay more attention to the more interesting parts. For instance, the designer may not be interested in design points with more than four processors. Then it is possible to hide those subtrees in the DSE tree that contain more than four processors and focus only on those parts of the design space that contain design points with the preferred parameters.

Filtering Based on the Distance from the Global Pareto Optimal Set In the multi-objective context, the goodness of a design point can be evaluated by its distance from the Pareto optimal set. The smaller the distance the better the solution. This distance measure is the Euclidean distance (in the objective space) between the design point and the nearest member of Pareto optimal set. In VMODEX, we provide a filtering option based on this distance measure. The designer is able to define a distance threshold and then filters the design points to see solutions that have lower/higher distance

values than the threshold. Furthermore, as in VMODEX both the design parameters and objectives are shown in a single view, the designer can easily understand which parts of the design space contains solutions that are close to/far away from the Pareto optimal set.

Filtering Based on the Mapping Decision For better analyzing the effect of different mappings on the objective values, we provide the following filtering option. The designer can specify some constraints on the mapping and then filters the design points to see only those solutions that not violate the specified mapping constraints. After that, it is possible to investigate the values of different objectives for the design points, which satisfy the mapping restrictions and find out the influences of mapping decisions on the objective values. There are three ways for defining a mapping constraint:

1. Some specific tasks (determined by the designer) are mapped on the same processor, without specifying the type of the processor. In this case, we can study the effectiveness of the internal communications between them as well as comparing the results of using different processor types for executing those tasks.
2. Some specific tasks are not mapped on the same processor. So, we can investigate the effect of using (shared) memories for their communications.
3. For one or more tasks a specific processor type is determined to be mapped to. In this case, we can investigate the effect of other tasks mappings on the objective values.

For instance, the designer may want to know what objective values can be obtained with the following mapping constraint: tasks 1 and 2 are mapped on the same processor, task 4 is executed by processor ASIP-1 and task 5 and 6 are mapped on different processors.

Filtering Based on the Secondary Objective Values Secondary objectives are not considered as the optimization goals, but they provide more detailed information about the evaluated design points. Thus, we also provide filtering options based on the secondary objectives. For instance, it is possible to filter design points based on their average utilizations or specify a minimum desired utilization for each architectural component.

Step by Step Animation

Since exploring the design space is an iterative process, it is important to know how a searching algorithm walks through the design space and trace its progression in finding new design points and covering the design space during successive iterations. In VMODEX, we provide an interactive user interface that allows designers to follow

the evolutionary exploration process during numerous generations. This interface is called step-by-step animation. The designer chooses an arbitrary generation number as a start point. At this step, the DSE tree shows only those design points that are found in the specified generation number or in earlier generations. Then, the designer can move forward or backward in generations to see the dynamic exploration.

When moving through the generations (i.e., replaying parts of or even the entire search process), it is important to know in each generation which new design points are found (added to the DSE tree). To show this, the design points generated in the current generation are blinking. If a parameter node, which had no data node in the previous step, receives its first data node in the current generation, it starts blinking as well. Therefore, the designer can easily notice that some new parts of the design space are visited for the first time by the algorithm in the current generation. For the hidden sub trees, in case of any data node addition, the blue triangle starts blinking and if the user is interested in viewing that sub tree, it can be expanded as well.

Details on Demand

Because of limited screen space as well as high data complexity, showing every detail for all design points is impractical. Therefore, generally, in proposing a visualization method, only the most important data features are shown directly in the visualization and detailed information for interesting parts is shown just after the user requests them. In VMODEX, a variety of detailed information are provided, enabling designers to interpret the evaluated design points from different aspects and gaining additional insight into the underlying information. This detailed information has been described earlier in this section and are called the secondary objectives. By right clicking on an interesting design point, a pop-up menu appears that shows the available detail options. The user can simply select the desired detailed information from the list and see it.

Adjusting DSE Tree Appearance

In VMODEX, there are some options for modifying the way that the DSE tree looks, based on alternatives for showing/hiding sub tress, coloring the parameter nodes and tree edges.

Showing/Hiding Sub trees In VMODEX, the designer can choose to see the entire design space or only those parts of the design space that are visited by the searching algorithm. Furthermore, it is possible to hide the sub tree of a (uninteresting) node by double clicking on it.

Coloring Parameter Nodes As we described in Section 3.2.2, in the DSE tree, the color of design point nodes represent the processing time and the color of the

third dimension shows the energy consumption. Parameter nodes, however, do not represent single design points and therefore do not have the direct notion of processing time or energy consumption. For this reason, we provide some options to color the parameter nodes. The designer is able to choose the average, minimum, or maximum of either processing time or energy consumption of the design points in their sub trees as a method for coloring the parameter nodes. The color of parameter nodes that have no data node (i.e., do not have any DSE data) is white.

Another option for coloring parameter nodes is based on the generation number in which the searching algorithm could get access to that part of the design space for the first time. (See Section 3.2.4)

Coloring and Thickness Regulation of Tree Edges As we explained in Section 3.2.3, tree edges can be visualized according to two importance factors: minimum Euclidian distance and last generation number. In the VMODEX user interface, there is an option for selecting one of these factors for visualizing the edges.

3.3 A Case Study

In this section, we present a real application case study to illustrate the benefits of using visualization in the design space exploration process. In this case study, we map a Motion-JPEG (M-JPEG) encoder to a heterogeneous multi-processor system-on-chip platform architecture consisting of a general-purpose microProcessor (mP), a micro-Controller (mC), an Application Specific Instruction Processor (ASIP) and two Application Specific Integrated Circuits (ASIC-DCT and ASIC-VLE). For communication, the platform architecture contains two dedicated point-to-point FIFOs (between mP and ASIP) and two shared memories; one Static RAM (SRAM) and one Dynamic RAM (DRAM), each one is accessible through a common bus. Note that the evaluated design instances only use a subset of the platform resources, based on the mapping of application tasks and communication channels onto the platform resources. Figure 3.12 represents our MP-SoC architecture model and Figure 3.13 shows the M-JPEG encoder application.

In our case study, an mC or an mP processor can execute all the different processes in the M-JPEG application while an ASIP can execute only three processes, namely: DCT, Q and RGB2YUV. The ASIC-DCT is designed for executing the DCT process and ASIC-VLE is dedicated for executing the VLE process.

The design space that we consider in this study has four parameters: number of processors, processor type, number of memories and memory type. Each architecture platform instance is indicated by a unique combination of these parameters. Those combinations that lead to the platform instances that are capable of executing the application are denoted as *feasible combinations*. Using the NSGA-II multi-objective evolutionary optimizer [19], we intend to find a set of optimal design points (in terms of alternative architectural solutions and mappings) under three criteria: processing

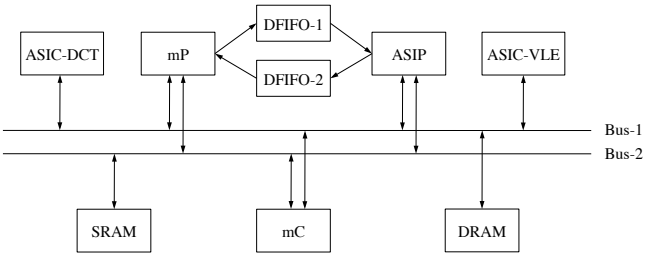


Figure 3.12: Heterogeneous multi-processor system-on-chip architecture model

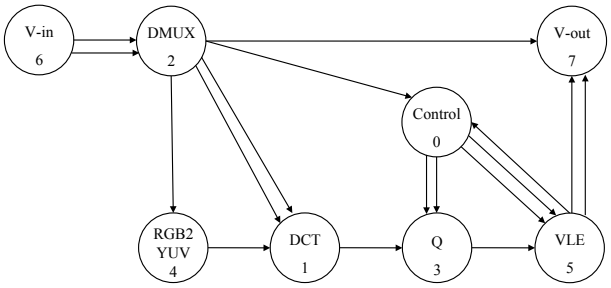


Figure 3.13: M-JPEG encoder process network graph

time, energy consumption and architecture cost. For this study, we run the MOEA for 100 generations with 100 individuals per population. Therefore, 10000 design points are searched by the MOEA. In this experiment, we have used the uniform crossover with probability 0.9. The mixing ratio between two parents is 0.5 (i.e. the offspring has approximately half of the bits from first parent and the other half from second parent). The mutation probability[2] is 0.23 and the bit mutation probability[3] is 0.01. For evaluating design points in terms of aforementioned criteria, we used the Sesame system-level simulator. In our case study, the processing time varied by factor of 6.4, energy consumption by factor of 4.45 and architecture cost by factor of 10.25. These large variations indicate the need for automated exploration and optimization.

VMODEX allows designers to look at the evaluated design points and explored design space from different perspectives and analyze the search results at multiple levels of abstraction. In the following subsections, we analyze the M-JPEG case study with respect to the following issues: 1) design space coverage, 2) the characteristics of the global Pareto optimal points, 3) investigating the absence of ASIC-VLE in the Pareto optimal set, and 4) studying the effects of executing the DCT process by different processor types on design criteria.

[2]i.e., the likelihood of mutating a particular individual.

[3]i.e., the likelihood of mutating each bit of an individual in mutation.

3.3.1 Design Space Coverage

Since we use an evolutionary algorithm (instead of exhaustive search) for exploring the design space, the searching algorithm may not visit some parts of the design space. Therefore, there is no evaluated data for those parts. However, it is essential that a searching algorithm achieves a broad coverage of the design space. This means that the algorithm is able to find solutions in as many as possible different regions of the decision space, even if those solutions are not Pareto-optimal. In this case, the exposed solutions represent the variety of possible designs in the decision space and enable the designer to do a more comprehensive study on the relationship between design parameters and their effects on the design criteria. Furthermore, a higher diversity and coverage in the searching algorithm also means that there is a higher chance of finding the global optimal solutions rather than the local optimum ones (the algorithm does not get trapped in local optimum points in the design space). The spread of design points in the design space can help the algorithm to escape from such local optima.

Using VMODEX, the designers can easily see the parts of the design space that are covered/not covered by the MOEA. This tool clearly shows which parts of the design space contain evaluated solutions and which parts are not searched at all (no design point is evaluated there). As we have described before, parameter nodes with a white color and dashed line have no data. In our case study there are 99 possible combinations of design parameters. By visualizing the exploration results, we could find out that more than 90% of possible platform instances are searched (91 out of 99). For platforms with two and three processors, all the possible combinations are searched. Further, only a few combinations of platforms with four and five processors have not been visited by the MOEA. Those parts of the design space that are not investigated during the exploration process are shown in Figure 3.14. In this figure, minimum processing time is used for coloring the parameter nodes. The white nodes represent the combinations that are not searched by the MOEA. For instance, there is no evaluated design point that consists of four processors of the types ASIC-DCT, ASIC-VLE, ASIP and mP and two memories of which one is DFIFO-1 and another one is DRAM (this platform instance is indicated by subtree "A" in Figure 3.14). However, the color of the node labeled with "2" in this subtree (at the number of memories level) indicates that the other combinations of two memories such as one DRAM and one SRAM are searched.

Next, we are going to study how our MOEA walked through the design space and found new architecture platform instances during its search iterations. We used the "visualization of design space coverage" option in VMODEX to understand the searching behavior of the MOEA in the design space. By selecting this option, parameter nodes are colored based on the generation number in which the searching algorithm gained access to that part of the design space for the first time. Therefore, the color of the parameter nodes, which are searched in the earlier generations, is lighter and those parameters that are visited in the later generations are darker. In our case study, we found that most of the architecture platform instances (possible combinations of design parameters) are searched in the 15 first generations. Actually, in

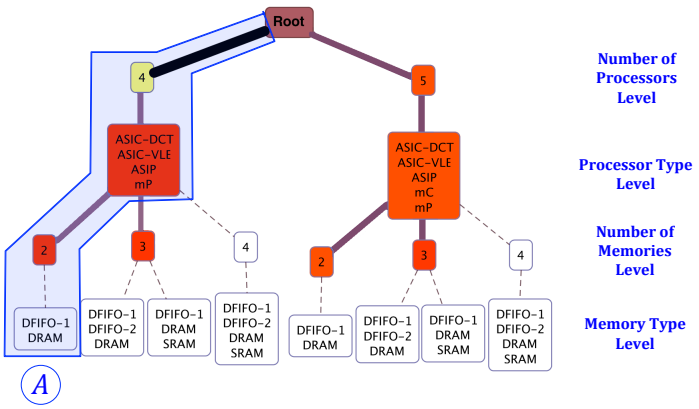


Figure 3.14: Platform instances that are not visited by the MOEA

our initial population, which is generated randomly, 58 unique instances are visited and then until the 15^{th} generation totally 86 (out of 99) possible combinations are searched. In the next generations, the algorithm could find 4 more instances and after the 65^{th} generation it could not find any new combination of design parameters. Figure 3.15 shows the visualization of design space coverage for our case study. Since the size of the DSE tree representing the entire design space is large and it is not possible to include it here, this figure shows only four processors architecture instances. Furthermore, the four last combinations that are found by the algorithm are in this part of the design space. So, we can easily see which architecture instances are searched in the later generations.

As can be seen in this figure the color of most parameter nodes is light green, which indicates that those parts of the design space are searched in the beginning generations. Only four nodes (denoted by A-D) are colored with darker green, which points out that those architecture instances are visited in the later generations. The number written at the bottom of these nodes shows the number of the first generation during which the algorithm has investigated them.

As we described before, the size of the blue triangle at the bottom of each node represents the number of design points in its subtree. From Figure 3.15 we can see that the platform instances containing four processors of the types ASIC-DCT, ASIP, mC and mP (subtree indicated by "S") are searched more often by our MOEA since nodes in this subtree have the biggest triangles.

In Figure 3.15, the minimum Euclidian distance is used as an importance factor for edge visualization. Therefore, the edges in the path from the root to the global Pareto optimal points are the thickest and darkest (black color) since the distance is zero. From Figure 3.15 we can see that among four processors platform instances only one of them leads to the global optimal design points, which is indicated by subtree S. This explains why the searching algorithm evaluated more design points in subtree S. In this part of the design space it has found design points that are closer to the opti-

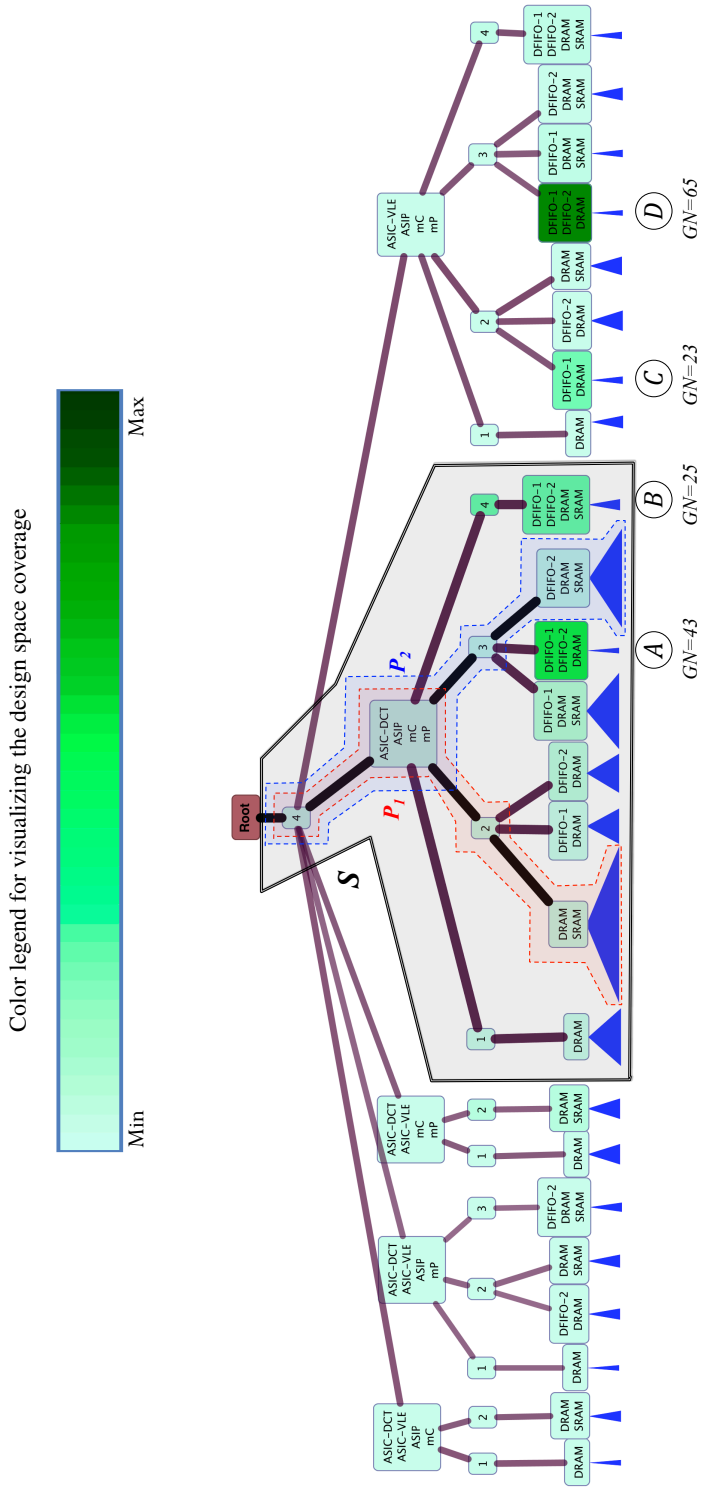


Figure 3.15: Visualization of design space coverage for our case study

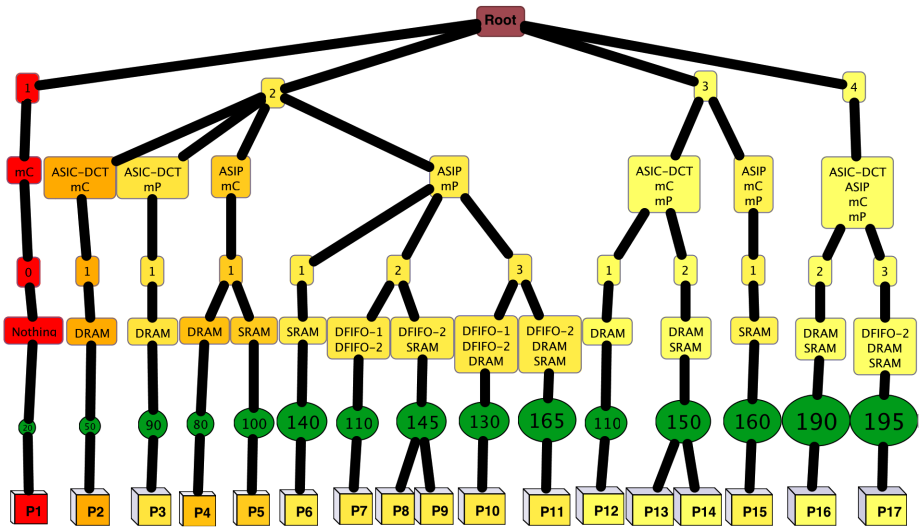


Figure 3.16: Global Pareto optimal points found by the MOEA (denoted by P_1-P_{17}). 15 unique combinations of architectural components lead to the 17 Pareto optimal points.

mum solutions. Within subtree S, two combinations of memory types yield optimum solutions. The two architecture instances that contain optimum design points in their subtrees are denoted by P_1 and P_2.

3.3.2 The Characteristics of the Global Pareto Optimal Points

Figure 3.16 shows the global Pareto optimal points found by our MOEA. By looking at the picture, the designer can immediately recognize the characteristics of the Pareto optimal points, which are the best design points with respect to the design criteria. As in the DSE tree both design parameters and objective values are shown in a single view, the designer can easily find out which combinations of architectural components yield optimum design points and what the trade off is between objective values. As can be seen in Figure 3.16, in our case study, fifteen unique combinations of architectural components lead to the seventeen Pareto optimal points. The variety of architecture cost in the Pareto optimal set is quite large. The cost is varying from the cheapest one to almost the most expensive one that have been encountered during the whole search (approximately $10x$ $(195/20)$). For the processing time, except the design point with a single processor architecture (labeled by "P1" in Figure 3.16) all the other points have relatively good processing time. The normalized value of the processing time (excluding P1) is in the interval [0, 0.21]. The energy consumption of all the found Pareto optimal points is relatively low. Its normalized value is in the range [0,0.16].

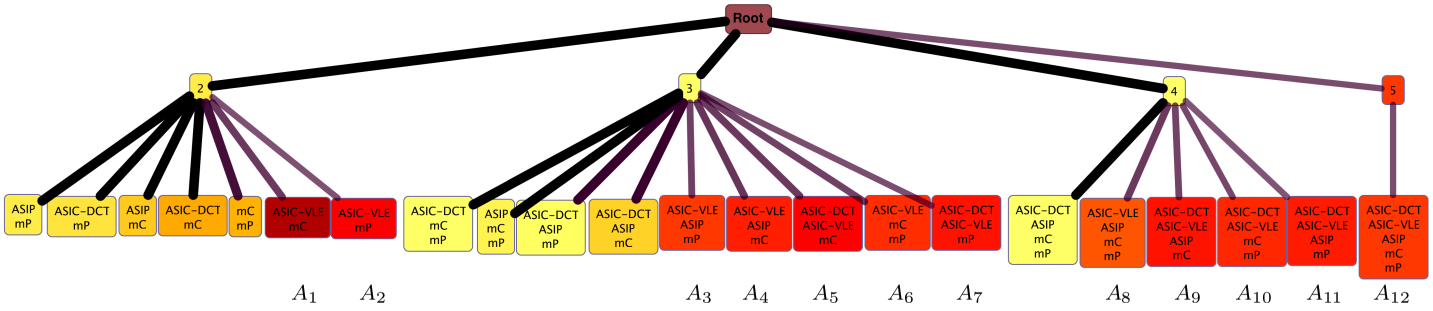
Now we are going to analyze the discovered Pareto optimal points in terms of design parameters. From Figure 3.16 we can see that, in our case study, there is no Pareto

optimal point with five processors. This means that with less processors (which is cheaper) we can get the same or better processing time and energy consumption. Therefore, using five processors is not appropriate for this application. Another interesting feature is that most of the obtained Pareto optimal points (10 out of 17) contain two processors in their underlying architecture. Thus, maybe two processors architectures are more suitable for executing our application. It can also be seen that there is no Pareto optimal point with four memories. So, the communications between processors can be done efficiently with less memories. Furthermore, we can see that the processor ASIC-VLE, which is dedicated for executing the VLE process, is never used in the Pareto optimal set. In the next subsection we are going to find out the reason of this remarkable property.

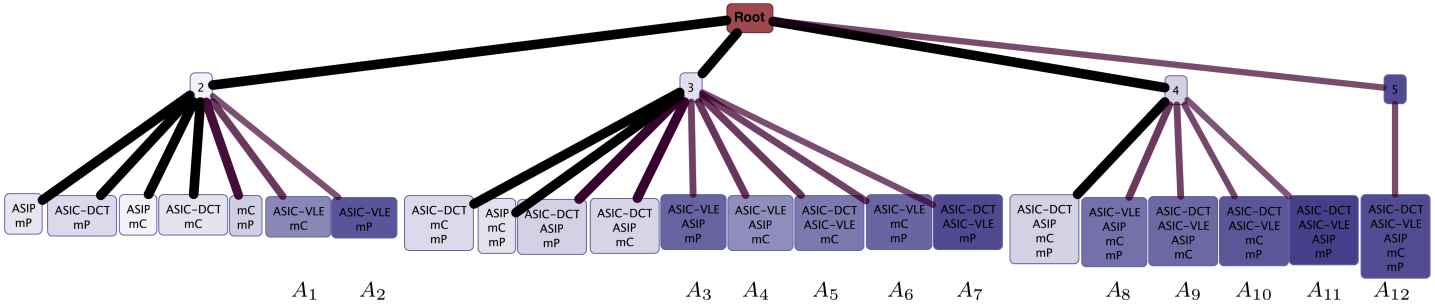
3.3.3 Investigating the Absence of the ASIC-VLE in the Pareto Optimal Set

As we can see in Figure 3.16, there is no Pareto optimal point that contains ASIC-VLE in its underlying architecture. By looking at the DSE tree representing the entire design space, we can find that there are several evaluated design points that use ASIC-VLE. Thus, our MOEA could access the parts of the design space that contain ASIC-VLE. However, the quality of design points in these parts with respect to the design criteria is not optimal and they are dominated by the solutions in the other parts. Figure 3.17 shows all the possible combinations of processor types. To make the picture fit here, different alternatives with respect to the number and types of memories are not shown. Thus, only the first two levels of the parameter segment are shown (i.e. the number of processors and processor types). In Figure 3.17(a) the minimum processing time and in Figure 3.17(b) the minimum energy consumption is used for coloring the parameter nodes. As can be seen in these two figures, in all of the platform instances containing ASIC-VLE, both the processing time and energy consumption is quite high.

We use filtering options provided in VMODEX to realize why both the processing time and energy consumption are significantly increased when using an ASIC-VLE. First, we filter design points based on their primary objective values. We were interested to see all design points which are relatively good in both processing time and energy consumption (their normalized value is in the range [0, 0.2] for both objectives). There was no restriction in the architecture cost objective. We found out that in all of them three processes V-Out, VLE and Control are mapped to the same processor. We did similar filtering to see those design points that are extremely poor in both (their normalized value is in the range [0.8, 1.0] for both objectives). Then, we could recognize that in all of them, the three aforementioned processes are mapped to different processors. Thus, we can conclude that the communications between these processes are much more intensive than the communications between other processes.The reason for this is that over these channels tables for Huffman encoding are being transmitted (see [77]). However, for the other processes, the communications are all the same as they communicate pixel blocks, which are small units of data. Therefore, using



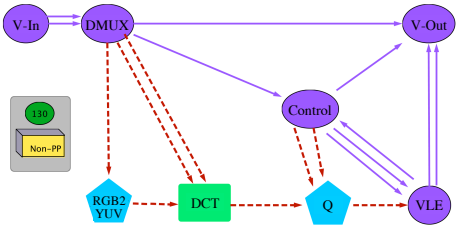
(a) Minimum processing time is used for coloring parameter nodes. For all platform instances containing ASIC-VLE (indicated by A_1-A_{12}), the processing time is quite high (the color of node is red).



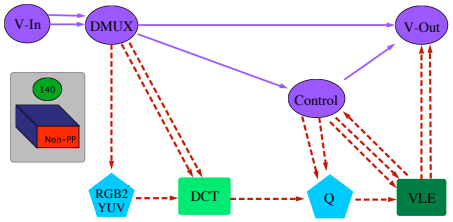
(b) Minimum energy consumption is used for coloring parameter nodes. For all platform instances containing ASIC-VLE (indicated by A_1-A_{12}), the energy consumption is quite high (the color of node is dark violet).

Figure 3.17: Possible combinations of processor types

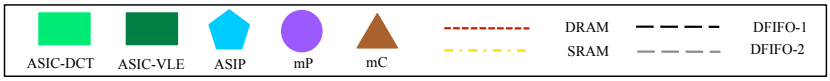
shared memories for these communications causes large overheads that can actually make the system slower. However, if these communications are done internally (mapping in the same processor) the mapping is much more efficient with respect to both processing time and energy consumption. As ASIC-VLE is dedicated for executing the VLE process, by using it, the processes V-Out and Control have to be mapped on the other processors and therefore their communications will be done externally using shared memories, which is not efficient. Thus, even though using an ASIC implementation for the VLE process is computationally efficient, because of its intensive communication, it is not an effective solution. In Figure 3.18 we illustrate this conclusion with an example. This figure represents the mapping decisions of two design points in which they have exactly the same mapping for all the processes except for VLE. In Figure 3.18(a) the VLE process is mapped to the mP (the same processor as V-Out and Control) and in Figure 3.18(b) it is mapped to the ASIC-VLE. From the visualization of their objective values we can see that they are significantly different in both processing time and energy consumption. The legend for shape and color of processor and memory types are shown in Figure 3.18(c).



(a) VLE is mapped to the same processor as V-Out and Control



(b) VLE is mapped to the dedicated ASIC-VLE processor



(c) Legend for shape and color of processor and memory types

Figure 3.18: Mapping decisions (with and without ASIC-VLE)

3.3.4 Studying the Effects of Executing the DCT Process by Different Processor Types on Design Criteria

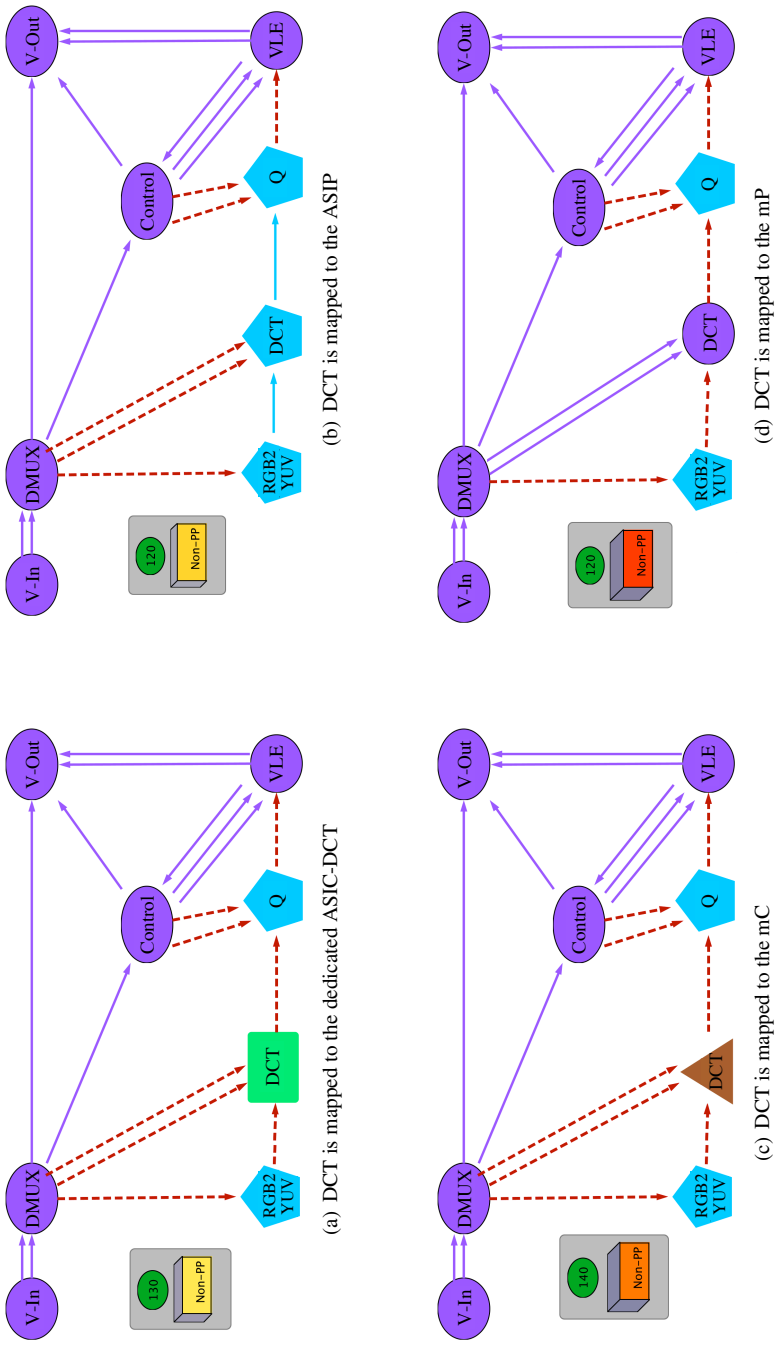
In the M-JPEG application, the DCT is the most computationally intensive process and therefore it is essential to be mapped on the processor that is optimal for executing it. We are going to investigate how the values of processing time and energy consumption change by running the DCT on different types of processors. To do that, we use mapping filtering. From the previous discussion we understand that if three processes V-Out, VLE and Control are not mapped on the same processor, regardless of the mapping of the other processes, the processing time and energy consumption are quite high. So, in this study, we assume that these three processes are mapped on the same processor and then we examine the influences of different mappings for DCT on the objective values. Using mapping filtering, we could find out that if DCT is mapped to ASIC-DCT or ASIP both the processing time and energy consumption is relatively good. But, by executing the DCT on the mC or mP, we cannot get any satisfactory solution. To illustrate this conclusion, an example is shown in Figure 3.19. This figure represents the mapping decisions of four design points in which the mappings of all processes excluding DCT are the same. The variation on mapping the DCT are shown in Figure 3.19(a) to Figure 3.19(d). From the visualization of the objective values we can see the effect of each mapping on processing time and energy consumption.

The analysis we performed in this section would have been very cumbersome and time consuming to do by only looking at the raw data or by using traditional 2D/3D graphs. Several traditional graphs are needed in order to interpret the data like we did. However, using VMODEX, a single visualization view of the design space enables very powerful and rapid analysis of the DSE data.

3.4 Comparing Subspaces

As we described in Section 3.2.1, in VMODEX, the design space is modeled as a tree and this kind of modeling causes the design space to be divided in several subspaces. Each subspace represents a unique instance of the architecture platform. On the other hand, solutions inside a subspace have exactly the same architecture components (have the same parents at the parameter levels) but the way that the application is mapped onto those components is different. We have provided additional functionality in VMODEX to allow designers to evaluate and compare the properties of interesting subspaces from various perspectives.

Figure 3.20 represents the design space of our case study, which is visualized by VMODEX. This figure shows only those parts of the design space that are compared with each other in this section. To make the picture smaller to be able to put it here, only global and local Pareto points are shown and non-Pareto points are not displayed. In the DSE tree shown in Figure 3.20 there are four subspaces, which are indicated by dashed contour lines and are labeled from S_1 to S_4. For instance, the subspace S_1 consists of one ASIP, one mC, one DRAM and one SRAM. Thus, all design points



(a) DCT is mapped to the dedicated ASIC-DCT

(b) DCT is mapped to the ASIP

(c) DCT is mapped to the mC

(d) DCT is mapped to the mP

Figure 3.19: Illustration of the effects of executing DCT process by different processor types on design criteria

in this subspace utilize these architecture components (same resource allocation) but their task and communication bindings are different. In each subspace, the Pareto optimal solutions found by a MOEA are called local Pareto optimal solutions, which are optimal with respect to a specific architectural instance. However, in the entire design space, a design point might exist which dominates the local Pareto point. In the following subsections, we explain the techniques VMODEX provides for analyzing and comparing the properties of discovered design points in different subspaces. We use these techniques for comparing the properties of the four local Pareto optimal sets, which are labeled by LPS_1 to LPS_4 in Figure 3.20.

The concept of local Pareto solutions and proposing some methodologies for evaluating and comparing different local Pareto optimal sets (like we explain here) is a new point of view in the multi-objective DSE process and has not been considered before. Such comparison is, however, essential to the designer as it provides insight into the landscape of the design space and can help him to comprehensively understand the properties of the discovered design points in different subspaces of the explored design space. Using VMODEX the designer is able to select the interesting subspaces and compare them with respect to the different aspects, which are explained in the following subsections.

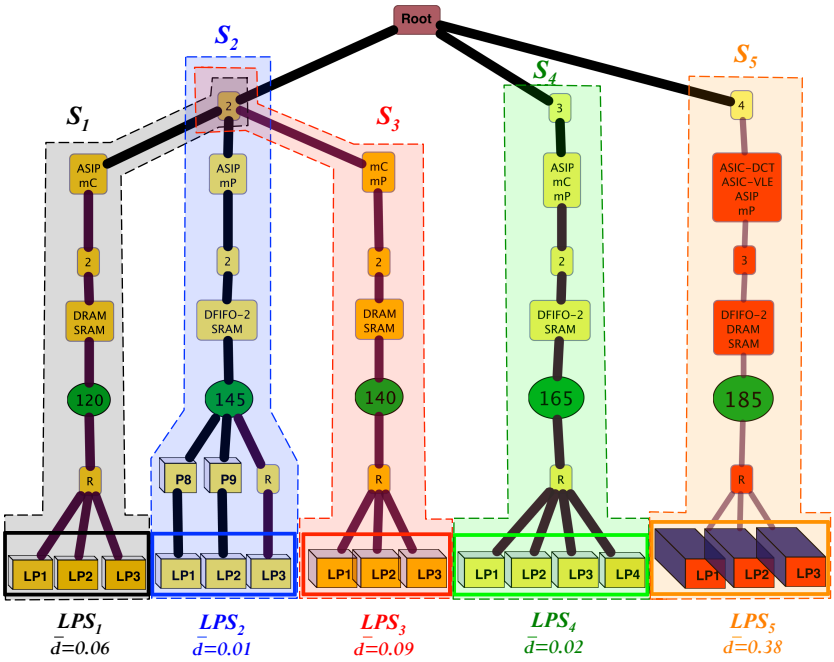


Figure 3.20: Subspaces of the design space

3.4.1 Distance from the Global Pareto Optimal Solutions

A subspace containing local Pareto optimal solutions, which are closer to the global Pareto optimal set, is more preferable. The distance can be measured in two ways: 1) the number of solutions in a local Pareto optimal set which are also in the global Pareto optimal set, and 2) the average of Euclidian distances (in the objective space) between the solutions in a local Pareto optimal set and the nearest member of the global Pareto optimal set.

In the DSE tree, both distance measures can simply be evaluated. Just by looking at the tree, one can easily recognize which solutions of a local Pareto optimal set are in the global Pareto optimal set as well. For those solutions, their parents (in the tree) are Pareto optimal nodes; otherwise they become children of a relation node. For example, in Figure 3.20, two solutions (out of three) in the set LPS_2 are globally Pareto optimal. However, in the other sets, none of the solutions are in the global Pareto optimal set (their parents are relation nodes).

The second distance measure can be seen directly in the DSE tree as well. The color and thickness of edges show the distance from the nearest global Pareto optimal solutions. The edges in the path from the root to the main Pareto optimal solutions are the thickest and darkest since the distance is zero. As the distance increases the edges become thinner and lighter. The value of the average distance between solutions in a local Pareto optimal set and global Pareto optimal set is shown at the bottom of the local Pareto set (denoted by $\bar{d}$). If the designer is interested to know the exact value of the distance measure for a particular solution, then the distance value is shown by clicking the corresponding edge. For example, in Figure 3.20, the average distance for LPS_2 is 0.01 which means that solutions in this set are quite close to the Pareto optimal set. However, the solutions in set LPS_5 are relatively far from the global Pareto optimal set since their edges are thin and light and the average distance is 0.38.

3.4.2 Coverage of Local Pareto Sets

Ziztler and Thiele [78] introduced the Coverage (C) metric, which directly compares two Pareto optimal sets with each other. The metric $C(P_1, P_2)$ calculates the proportion of solutions in Pareto optimal set P_2 which are weakly dominated at least by one solution in P_1:

$$C\left(P_1, P_2\right) = \frac{|\{p' \in P_2 | \exists\ p \in P_1 : p \leq p'\}|}{|P_2|} \tag{3.11}$$

Where $\leq$ is the weakly dominance relationship. For two solutions p and p' it is said that p weakly dominates p' , if p is not worse than p' in all objectives. In fact, the function C maps the ordered pair (P_1, P_2) to the interval [0, 1]. The value $C\left(P_1, P_2\right) = 1$ means that all members of P_2 are weakly dominated by P_1 and $C\left(P_1, P_2\right) = 0$ represents the situation where none of the solutions in P_2 are weakly dominated by P_1. Note that for fully understanding the dominance relations between two Pareto optimal sets, both directions $C\left(P_1, P_2\right)$ and $C\left(P_2, P_1\right)$ have to be considered, since

$C(P_1, P_2)$ is not necessarily equal to $1 - C(P_2, P_1)$. The C metric compares only two sets with each other. Thus, for comparing more than two sets, we propose a new metric called Total Coverage (TC), as follows:

$$TC(P_i) = \sum_{j=1, j \neq i}^{n} C(P_i, P_j) - C(P_j, P_i) \tag{3.12}$$

Where n is the number of comparing sets. $TC > 0$ means that the dominating rate is higher than the dominated rate and the $TC < 0$ implies that the solutions are more dominated by the other sets than they dominate solutions in the other sets. Therefore, a set with a bigger TC value is better. In VMODEX, the designer is able to select the interesting local Pareto sets and compare them using the TC metric. For better understanding the dominance relations between sets, we visualize this metric. A directed weighted graph is used for visualizing the TC metric. Each comparing set is shown as a node in the TC graph. For each two sets P_1 and P_2, if $C(P_1, P_2) \neq 0$ then an edge is drawn from P_1 to P_2, of which the weight is equal to the C value. In the case of $C(P_1, P_2) = 0$, there is no edge from P_1 to P_2. Due to the summation property, we can break up a summation across a sum or difference. Therefore, we can rewrite the TC formula as follows:

$$TC(P_i) = \sum_{j=1, j \neq i}^{n} C(P_i, P_j) - \sum_{j=1, j \neq i}^{n} C(P_j, P_i) \tag{3.13}$$

According to the TC graph, the first sum in Equation 3.13 is equal to the sum of the weights of outgoing edges and the second sum is equivalent to the sum of the weights of incoming edges. Thus, for each node in the TC graph, the value of TC metric is calculated by the sum of the weights of outgoing edges minus the sum of the weights of incoming edges:

$$TC(P_i) = \sum (Outgoing\, Edges) - \sum (Incoming\, Edges) \tag{3.14}$$

The size of the nodes in the graph indicates the TC value. Therefore, nodes with higher TC values are bigger. Because the TC value greater than zero (means more dominating) or less than zero (means more dominated) have completely opposite meaning, we demonstrate this property in the nodes color. Nodes with $TC > 0$ are shown in blue while nodes with $TC < 0$ are shown in red. Figure 3.21 shows the visualization of the TC metric for sets LPS_1 to LPS_5 shown in Figure 3.20. As can be seen in this figure, all solutions in LPS_5 are dominated by all the other comparing sets (there are four incoming edges from LPS_1 to LPS_4 with the weight $4/4 = 1$), while there is no solution in LPS_5 that dominates a solution in the other sets (LPS_5 does not have any outgoing edge). Thus the TC value for LPS_5 is less than zero ($TC = -4$) and this node is shown by a red color. Furthermore, we can understand that solutions in LPS_1 dominate all the solutions in set LPS_3, while no solutions in LPS_1 are dominated by any other sets, since there is no incoming edge. Moreover, we can see that half of the solutions is LPS_4 are dominated by solutions in LPS_2. As

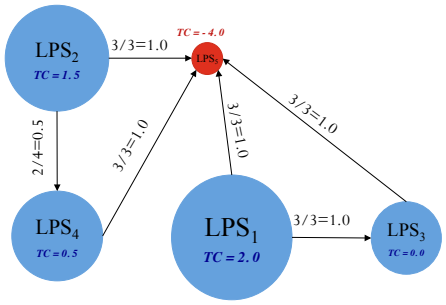


Figure 3.21: Visualization of the TC metric

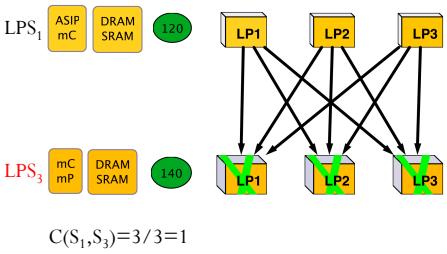


Figure 3.22: Visualization of the C metric

a result, LPS_1 has the highest TC value (biggest node in the graph) and thus is the best local Pareto optimal set among the comparing sets with respect to the TC metric.

If the designer is interested to know more about the dominance relation between each two sets, such as which solutions in one set dominate which solutions in the other set, it is possible to select those sets to see more details. To this end, we visualize the dominance relation between two sets as follows. Solutions in both sets P_1 and P_2 are shown in two different rows. If a solution in P_1 dominates a solution in P_2, an arrow is drawn between them coming out from the solution in P_1 to the solution in P_2. Furthermore, a cross is displayed at the dominated solution in P_2 to show that this solution is dominated by another one. Figure 3.22 shows the visualization of the dominance relation between two local Pareto sets LPS_1 and LPS_3. From this figure, we can understand that all solutions in set LPS_1 dominate all the solutions in LPS_3 and therefore LPS_1 is absolutely better than LPS_3. The architectural components of each local Pareto set are shown on the left side of each set. As can be seen in Figure 3.22, both local Pareto sets have the same components except that in LPS_3 the ASIP is replaced by mP. Using an mP instead of an ASIP is not beneficial since it increases the cost but does not yield a better processing time nor a better energy consumption.

3.4.3 Size of the Dominated Region

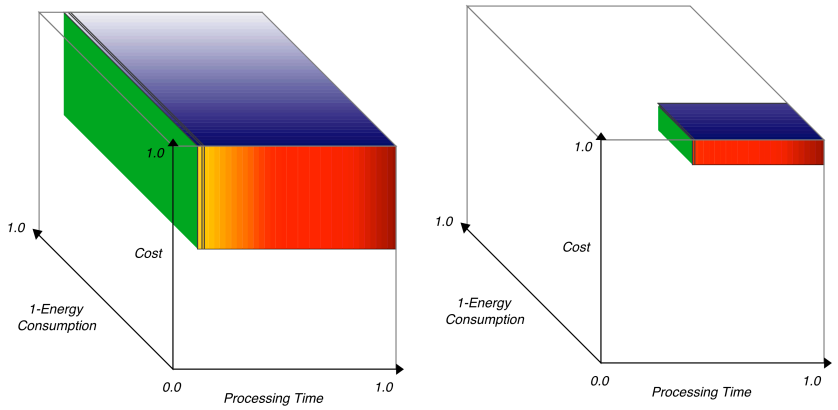
In [79, 80], the *Hypervolume* metric is proposed, which measures how much of the objective space is dominated by a given non-dominated set. A set with a larger hypervolume is desirable. We use this metric to compare the local Pareto optimal sets in different subspaces of the design space. Moreover, we visualize this metric to illustrate which area of the objective space is dominated by solutions in a local Pareto set. Figure 3.23 represents the visual from of the hypervolume metric for two local Pareto sets LPS_1 and LPS_5 shown in Figure 3.20. Each side of the cube shows one objective. The colored region indicates the dominated area in the objective space. Each point in this region is dominated at least by one member of Pareto optimal set. For better vision, each side of the cube is colored with the corresponding color scheme in the DSE tree. Black lines in the colored region denote the solutions in a local Pareto

optimal set. The normalized values of processing time (PT) and energy consumption (EC) for each solution, is written at the bottom of the cube. As can be seen in Figure 3.23, in set LPS_1 almost half of the objective space is dominated while only a small portion of the objective space is covered by the solutions in the LPS_5. Using the Hypervolume visualization, it is possible to see the dominated parts of each objective separately. For instance, in Figure 3.23 the energy consumption and processing time surfaces are almost completely covered for LPS_1 while only half of the cost surface is dominated. We also propose another approach for visualizing the Hypervolume metric, which is extendable to be used in problems with more than three objectives. This visualization method is explained in Chapter 4.

3.4.4 Sensitivity of Subspaces to Different Mappings on Design Criteria

In studying the properties of a certain subspace (i.e. specific platform architecture instance), it is a worthwhile issue to investigate how the objective values are being changed with modifying the mapping decisions. In this paper, we propose a visualization method for exploring the sensitivity of each subspace to different mappings on the objective values. To do this, the frequency distribution of objective values in each subspace is visualized to see the range as well as the frequency of objective values that can be achieved with a specific architecture instance by using different mappings. Since all solutions in a subspace have the same architecture cost, the frequency distribution is not applicable for this objective and should only be considered for the other two objectives: processing time and energy consumption.

For each objective, a horizontal axis from 0 to 1 is drawn and colored like the color-coding technique used for showing that objective in the DSE tree. For example, in our case, colors from yellow to red are used for representing the processing time and therefore, this color scheme is used for coloring the corresponding axis in the frequency distribution visualization. The height of each color bar in the (color-coded) objective axis indicates the number of design points with the objective value inside that range. Figure 3.24 shows the visualization of the frequency distribution for three subspaces. Figure 3.24(a) shows the frequency distribution of solutions in a subspace that contains an ASIP, mP, DFIFO-2 and SRAM. This subspace is indicated as S_2 in Figure 3.20. As can be seen in this figure, for all design points, both the processing time and energy consumption is relatively good (less than 0.5). Therefore, for this particular architecture, with different mappings we can get approximately good design points. So if the designer is looking for a system that can flexibly deal with different mappings, this architecture is a good solution. The subspace in Figure 3.24(b) consists of an ASIP, mC, mP, DFIFO-2 and SRAM, which is denoted as S_4 in Figure 3.20. In this subspace, both the processing time and energy consumption are varying from the best to almost the worst. However, the processing time and energy consumption of most design points are approximately good. Therefore, if the designer is interested in this platform architecture instance, he should take care about the mapping because a wrong mapping decision can make the difference between the best or the worst design



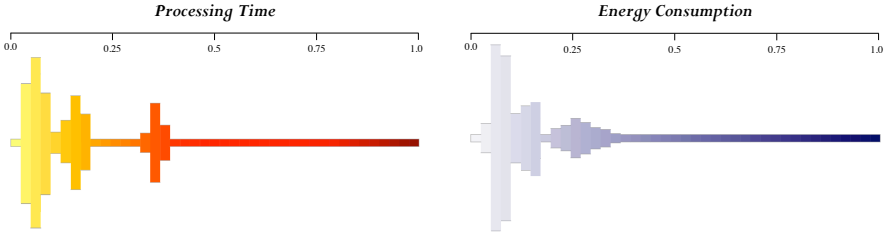
LP_1: PT=0.11, EC=0.02
LP_2: PT=0.13, EC=0.01
LP_3: PT=0.14, EC=0.01
Cost=0.54

$HV(S_1) = 0.43$

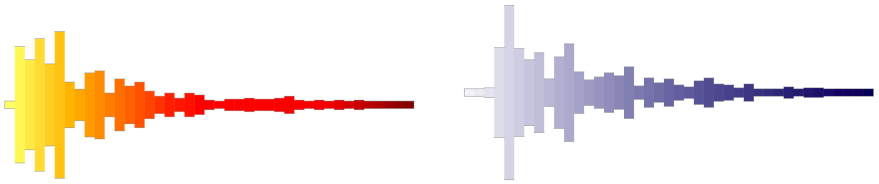
LP_1: PT=0.41, EC=0.79
LP_2: PT=0.41, EC=0.76
LP_3: PT=0.42, EC=0.74
Cost=0.89

$HV(S_5) = 0.017$

Figure 3.23: Visualization of the Hypervolume metric



(a) Subspace consisting of: ASICP, mP, DFIFO-2, SRAM



(b) Subspace consisting of: ASICP, mC, mP, DFIFO-2, SRAM

(c) Subspace consisting of: ASIC-DCT, ASIC-VLE, ASICP, mP, DFIFO-2, SRAM, DRAM

Figure 3.24: Visualization of the frequency distribution of objective values for three subspaces

point. Figure 3.24(c) represents the frequency distribution of subspace S_5 in Figure 3.20. This subspace is made up of following components: ASIC-DCT, ASIC-VLE, ASIP, mP, DFIFO-2, DRAM and SRAM. As can be seen in this figure, for all design points, both the processing time and the energy consumption are extremely poor. Even with different mappings, we cannot get an acceptable design point. Thus, this architecture is not a suitable solution for our case study.

Therefore, by using the frequency distribution visualization, the designer is able to analyze the effect of different mappings on the design criteria for a certain architecture instance. Furthermore, it is easy to compare the sensitivity of different subspaces with respect to the mapping decisions.

3.5 Conclusion

In this chapter, we introduced our proposed visualization techniques, which are specially designed for understanding the multi-objective DSE process of embedded systems that are based on heterogeneous multi-processor system-on-chip architectures. All the developed visualization techniques are incorporated into a tool, called VMODEX. Our visualization tool provides insight into the search process of heuristic searching algorithms that are typically used in the DSE process. It helps designers to understand how such algorithms explore the design space during their iterations and converge towards the optimal design points. It should be mentioned that although in this thesis we choose the Sesame simulator for evaluating design points and MOEAs for searching the design space, our visualization tool is not restricted to neither evaluating approach nor searching mechanism. It can easily be used for different types of evaluating methods as well as searching strategies.

The tree model that we propose for visualizing the design space enables us to visualize multivariate data. There is no limitation on the number of neither design parameters nor criteria. Furthermore, in our DSE tree model, the concepts of subspaces and local Pareto points are proposed, which are new points of view in the multi-objective DSE process and has not been considered before.

VMODEX also enables very powerful and rapid analysis of DSE results. Designers can look at the data from different perspectives and at multiple levels of abstraction. Several interactive capabilities are provided, which allow designers to play with data and find out some interesting and important features that may not be found just by looking at the static visualization.

To illustrate the benefits of using our visualization techniques in the DSE process, we performed an actual case study on Motion-HPEG encoder application. Then, we used VMODEX to interpret and analyze the results. In this chapter, We explained some new and unexpected insights that we could gain from the tool that is not possible to find out those insights from the raw data.

To summarize the different visualization approaches that we proposed in this chapter, Figure 3.25 shows all our proposed visualization methods together with the graphical

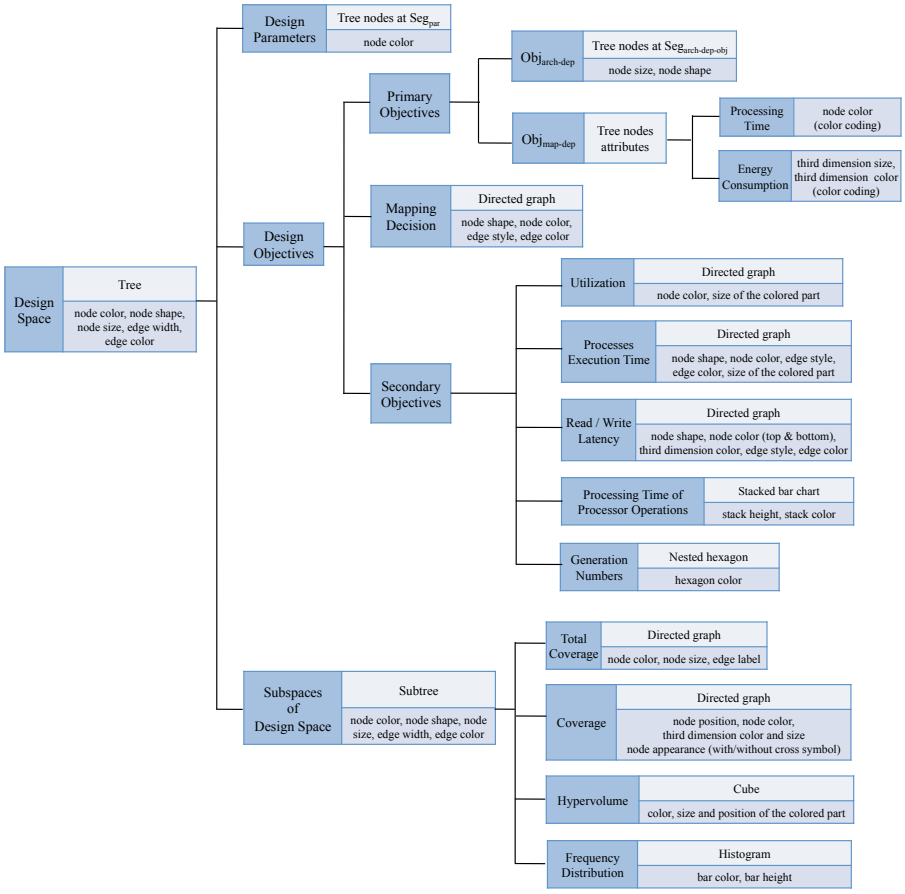


Figure 3.25: summarization of all our proposed visualization approaches together with their relations with each other

tricks that we used for visualizing them. Furthermore, in this figure, we clarify how these visualizations are related to each other.

In the following, we summarize the benefits of using VMODEX in multi-objective design space exploration process:

- It models the design space as a tree in which both the design parameters and objectives are shown in a single view. Therefore, it is easy to understand where the optimum design points are located and what objectives they have.
- There is no limitation on the number of design variables since each parameter is located at one level of the tree.
- It can easily be extended to show more than three objectives. Since each ob-

jective is assigned to one attribute of the tree nodes and nodes have various attributes such as shape, orientation, size, color, transparency, texture, border, etc.

- Several techniques are provided in VMODEX to be able to handle large design spaces.
- Various user-interactions are developed for more effective data exploration and enabling to find hidden patterns in the data and complex relationships among variables.
- Different properties of evaluated design points can be shown, allowing designers to interpret the data from different aspects and gaining additional insight into the underlying information.
- By coloring the parameter nodes, it is possible to do some statistical analysis. The designer can compare different architectures (in terms of number and type of the processors and memories) according to the minimum, maximum or average of each of the design criteria.
- Edge visualization helps designer to easily navigate through the design space and find more important parts.
- The concept of local Pareto point is proposed and in the DSE tree there is an explicit level called local Pareto level, which shows the optimal design points with respect to specific architecture instances. Therefore, the designer can easily compare the best design points of different architecture instances with each other.
- In VMODEX, each unique instance of the architecture platform is considered as a subspace of the design space. Various methods are provided for evaluating and comparing the properties of design points in different subspaces of design space.
- It helps designers to understand the search behavior in heuristic based design space exploration such as:
  - Which parts of the design space are not searched at all (no design point is evaluated there). As we mentioned before, nodes with a white color and dashed line have no data.
  - Which parts of the design space are searched more often by the searching algorithm (more design points are evaluated there). In these areas, the tree provides more nodes so the sub trees of the corresponding nodes are bigger.
  - The step by step animation option in VMODEX allows us to trace the progression of the searching algorithm in finding new design points and covering the design space during its consecutive iterations.

4

Performance Assessment of Multi-Objective Optimizers: Metrics and Visualizations

An important issue in multi-objective optimization is the quantitative comparison of the performance of different algorithms. In the case of using heuristic optimization methods, the outcome is usually an approximation of the true Pareto front and therefore a question arises on how to evaluate the quality of the discovered approximation sets. Various performance measures have been proposed to evaluate different quality aspects of the results of optimization algorithms. In this chapter, we explain some existing performance metrics as well as some new metrics that evaluate the performance of optimization algorithms from new points of view. We classify performance metrics into different groups, based on the quality aspect they measure. Furthermore, we propose several visualization approaches, which enable researchers to do detailed analysis of the quality of Pareto optimal solutions. These visualizations provide insight on the reasons behind the strength/weakness of a Pareto optimal set with respect to a particular metric.

4.1 Introduction

As we described in Chapter 2, due to the sheer size of the design space in real problems, an exhaustive exploration of all possible alternatives is not feasible. Furthermore, usually multiple criteria need to be optimized simultaneously. Therefore, heuristic search techniques, such as Multi-Objective Evolutionary Algorithms (MOEA), are often used for pruning an exponential design space and guiding the search process toward the most promising regions. Since objectives are often in conflict, there cannot be a single optimum solution, which simultaneously optimizes all objectives. Instead, a set of optimal solutions denoted as the ”Pareto optimal set” or ”non-dominated set” has to be found.

Although the goal of heuristic multi-objective optimization techniques is to find the Pareto optimal solutions with respect to the design criteria, there is no guarantee to reach real optimal solutions. This is because of the heuristic nature of these methods. They try to find optimal solutions. However, typically they are only able to find good approximations of optimal solutions that are not far away from the true optimal solutions. Therefore, metrics are needed to evaluate the quality of the discovered solutions. Furthermore, many different multi-objective optimization algorithms are proposed in literature, such as SPEA2 [18], NSGA-II [19], PAES [20], ACOG [21], AMOSA [22] etc., which may have a different performance on different problems, and there is no conclusive answer regarding to which algorithm is the best for a specific problem. On top of that, optimization algorithms are highly sensitive to the parameters being used, such as mutation rate, repair strategy, individual encoding, etc. These parameters have major effects on the performance of the algorithm and have to be fine-tuned by hand. Therefore, coming up with the best searching algorithm, which efficiently and effectively explores the design space and finds high quality solutions is a big challenge.

Therefore, we have added extra functionality to VMODEX to help algorithm developers to find the best optimization algorithm for their specific problem. Using VMODEX, algorithm developers can easily evaluate and compare the results of different optimization methods, with respect to their efficiency and effectiveness, in order to find the best approach. Then, the best optimization results are delivered to the designers for analyzing the design space exploration process.

In single-objective optimization, the quality of solutions can be defined by measuring the values of the objective function; the smaller (or larger) the value, the better the solution. However, in the context of multi-objective optimization, it is difficult to define appropriate quality measures since it is not clear what quality means in the presence of multiple objectives. In multi-objective optimization, multiple aims need to be satisfied simultaneously and thus there cannot be a single quality measure that is able to indicate the performance of an optimizer in an absolute sense. Therefore, several performance metrics have been proposed in the literature to assess different aspects of the quality of the Pareto optimal solutions obtained by heuristic multi-objective optimizers. A review of these performance metrics can be found in [81, 82]. However, most of the current performance measures concentrate on evaluating the quality of the found Pareto optimal solutions in the objective space and measuring the behavior of searching algorithms in the decision space has been mostly disregarded. However, in many applications, such as design space exploration for embedded systems, the structural property of the evaluated solutions in the decision space is also a key factor. The designer may be particularly interested in a range of structurally different solutions distributed over the Pareto optimal front. The diversity of the evaluated points in the design space represents the variety of possible designs in the decision space and enable the designer to do a more comprehensive study on the relationship between design parameters and their effects on the design criteria.

In this thesis, for comparing the outcomes of different optimization methods, we chose some performance metrics from literature that we considered most suitable for our context. We also propose three new metrics (*WSGR*, σ_{mst} and *DFPOS*), which eval-

uate the results of optimization algorithms in the objective space. Furthermore, we define some new metrics to assess the performance of optimizers in the decision space.

For evaluating and comparing different Pareto optimal sets, the results of a quality metric are typically shown in a table or displayed in a 2D graph in which the value of the quality metric is shown to compare one Pareto optimal set to another one. Although these kinds of representation are useful to find out which Pareto optimal set is better in respect to a certain quality aspect, they do not provide insight on why a Pareto optimal set is good (bad) according to the specific metric.

Utilizing visualization techniques can be helpful for further analysis of the Pareto optimal solutions. In general, a quality metric encapsulates the properties of a Pareto optimal set to one scalar value that somehow reflects a certain quality aspect. Therefore, some useful information may be lost because of this compression. However, visualizations can provide better understanding of the quality of the results and enable us to perform more accurate analysis. But, there is a significant lack of studies on representing and visualizing the different quality aspects of Pareto optimal sets. The conventional way is plotting the optimal solutions in the objective space. Although this method is simple and displays the overall quality, it is limited to a maximum of three objectives (maximal three dimensions) and does not provide a detailed description of different quality aspects. Ang et al. [83] proposed Distance & Distribution (DD) charts, in the first of which the Pareto optimal solutions are plotted against their distance to the true Pareto front and in the second of them the Pareto optimal solutions are plotted against the distance between each consecutive solutions. In these plots, only the distance values of solutions are shown and the objective values are not considered. So, they do not reveal any information about the location of the Pareto optimal solutions in the objective space.

In this thesis, we propose several visualization approaches, which enable researchers to do detailed analysis of the quality of Pareto optimal solutions. These visualizations provide insight on the reasons behind the strength/weakness of a Pareto optimal set with respect to a particular metric. To this end, we integrated various performance metrics (including both existing metrics and new ones) and their visualizations in VMODEX. Thus, using VMODEX, algorithm developers are able to perform a comprehensive study on the properties of the discovered optimal solutions and evaluate the performance of different optimization algorithms from various perspectives.

The rest of this chapter is organized as follows. Section 4.2 describes the goals that should be considered in multi-objective optimizations to yield high quality results. In Section 4.3 we introduce performance metrics and their visualization methods we have provided in VMODEX. Section 4.4 presents a case study in which the qualities of the outputs of two different optimization approaches for a specific problem are compared using various metrics. This case study illustrates the benefits of our tool, which integrates and visualizes different performance metrics in a single environment. The last section presents concluding remarks.

4.2 Goals in Multi-Objective Optimization

As we described in Chapter2, in multi-objective optimization problems, two different spaces should be taken into account: the decision space and objective space. Therefore, the performance of an optimization algorithm in both spaces needs to be assessed.

With respect to the objective space, usually two distinct goals are considered:

1. Find solutions as close as possible to the true Pareto front.
2. Discover solutions as diverse as possible.

The first goal is essential for any optimization task and therefore is common in both single and multi-objective optimizations. Finding solutions, which are far away from the true Pareto optimal set, is not appropriate. Only when the obtained Pareto front is close to the true Pareto front, one can be assured that a near-optimal set of solutions is found. In most multi-objective optimization problems, the true Pareto front is not known. Therefore, for measuring the closeness, a reference Pareto front that is the best-known approximation of the true Pareto front, is used. The reference Pareto front can be made by combining all the optimal solutions from numerous searches and then removing dominated solutions from the combined set.

The second goal is completely specific to multi-objective optimization. Diversity means covering the *entire* Pareto optimal region *uniformly*. Only with a diverse set of solutions, we can have a good set of trade-offs among objectives. The diversity can be divided in two different measures: extent (the spread of extreme solutions) and distribution (the relative distance between solutions). Figure 4.1 illustrates the concept of multi-objective optimization goals in the objective space for a problem in which two objectives (f_1 and f_2) should be minimized.

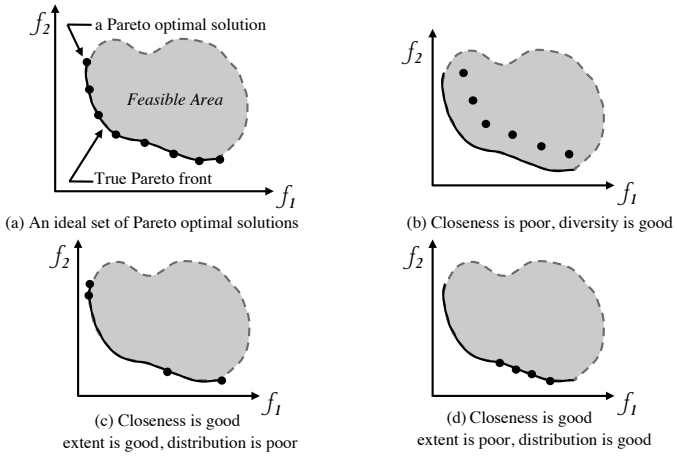


Figure 4.1: The concept of multi-objective optimization goals in the objective space

Since both goals are important, an efficient optimization algorithm must satisfy both of them adequately. Therefore, optimal solutions found by a good optimizer, are close to the true Pareto front and maintain a uniform distribution over the entire Pareto optimal region. Finding a diverse set of solutions, which are not close to the true Pareto optimal solutions, is not desirable since they cannot provide an accurate estimation of the true Pareto front. On the other hand, a close set of solutions with poor diversity cannot provide reasonable trade-offs between objectives.

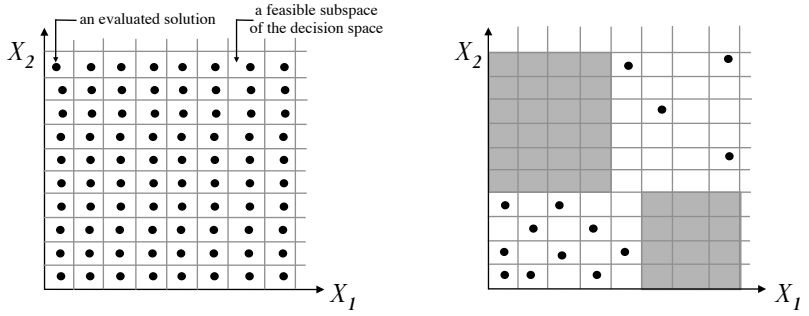
With respect to the decision space, we define two goals that should be achieved:

1. Find solutions in as many as possible different regions of the decision space.
2. Evaluate as many as possible unique design points.

When heuristic multi-objective optimization techniques are used for exploring the design space and finding the optimum solutions, it is possible that the searching algorithm does not visit some parts of the design space and therefore there is no evaluated data for those parts. However, it is essential that a searching algorithm achieves a broad coverage of the design space. This means that the algorithm is able to find solutions in as many as possible different regions of the decision space, even if those solutions are not Pareto-optimal. In this case, the exposed solutions represent the variety of possible designs in the decision space. A higher diversity and coverage in the decision space means that there is a higher chance of finding the global optimal solutions rather than the local optimum ones (the algorithm does not get trapped in local optimum points in the design space). The spread of design points in the design space can help the algorithm to escape from such local optima. Figure 4.2 illustrates the concept of diversity of evaluated solutions in the decision space for a problem with two design parameters (X_1 and X_2). In this figure, the design space is divided into exclusive cells (subspaces). Each subspace represents a partial instantiation of the design variables and includes a set of design points with similar characteristics. Each cell that contains at least one design point is considered as a searched cell. Figure 4.2(a) represents an ideal diversity in the decision space. For each subspace of the design space there is at least one evaluated solution. Figure 4.2(b) shows the situation in which the diversity is poor. The gray areas in Figure 4.2(b) indicate those parts of the design space that are not visited by the optimization algorithm and therefore there is no evaluated design point there. Thus, there is no knowledge about the properties of solutions in those parts.

The second goal comes up since during the process of design space exploration using a heuristic optimizer, some design points that have a good performance may be regenerated in different generations. Therefore, there might be some duplicate design points in different generations. An algorithm with less duplication has better performance because a greater number of design points can be investigated with the same computational effort.

Since the multi-objective optimization goals are distinct concepts, no single quality measure is able to indicate the performance of an optimization algorithm in an absolute



(a) An ideal diversity (b) A poor diversity

Figure 4.2: The concept of diversity of evaluated solutions in the decision space

sense. Therefore, for measuring the performance of different optimization approaches and comparing how well they achieved the multi-objective optimization goals, several metrics need to be used. In general, a performance metric assigns a number to a Pareto optimal set that somehow reflects how good it is with respect to a certain feature measured by that metric. However, to guarantee an exhaustive evaluation of a Pareto optimal set with respect to all the aspects of quality, a collection of different metrics is required in such a way that there is at least one metric for each aspect.

4.3 Performance Metrics and their Visualizations

In this thesis, we broadly classify performance metrics in two groups:

1. Metrics that measure the quality of found Pareto optimal solutions in the objective space.
2. Metrics that evaluate the behavior of optimization methods in the decision space.

In the next two subsections, we will introduce some metrics for each group. Furthermore, the visualization approach developed for each metric as well as the benefits of such visualization will be explained. Particularly, in proposing visualization methods for performance metrics, the following challenges have to be addressed:

- The visualization approach should not be limited to the number of objectives. It must be scalable to be used in more than three objectives problems.
- The visualization techniques should be simple and straightforward. Complex approaches do not reveal useful information clearly and some valuable issues may not be discovered. Furthermore, simple methods enable us to easily compare two or more non-dominated sets with each other.

- For showing a specific objective in different visualization methods for different metrics, the same metaphor should be used. Thus, there is no confusion because of representing one thing in different ways.

All the visualizations proposed in this chapter satisfy the above challenges. In VMODEX, for easy correlation between the visual form of performance metrics and the multi-objective DSE visualization (explained in Chapter 3), the same metaphors are used for showing the objective values and solutions. Thus, one will not be confused by the different representations of the same thing.

4.3.1 Performance Metrics in Objective Space

In this chapter, we consider the same objectives as in Chapter 3 that are processing time, energy consumption and architecture cost. The visual representations of these objectives are also the same. Processing time is shown by the color of the node representing a solution. Colors are varied from yellow to red with all color grades in between. Nodes with the lowest processing time are yellow and nodes with the highest processing time are red. The size and color of the third dimension of a solution shows the energy consumption. As the energy consumption increases, the size of the third dimension becomes bigger and its color becomes darker. The architecture cost is shown as separate nodes at the cost level. Cost nodes are represented with circle symbol. The size of the circle becomes bigger as the cost increases. Figure 4.3 represents an example of a set of Pareto optimal solutions that is visualized by VMODEX and we are going to evaluate the quality of this set with respect to various metrics in this section. In this figure, solutions are labeled by an index, in the order of increasing processing time.

Since objective functions may have different scales of measurement, one should map the limits of the objective function values to a unique interval, before doing any arithmetic operation. To make calculations scale independent, we normalize objective values using min-max normalization (see Section 2.1 in Chapter 2). At the end of normalization, all design points get a value in the range [0, 1] for their objective values.

In the following subsections, we categorize the performance metrics into four groups: 1) closeness metrics to evaluate the distance of the obtained Pareto front to the true Pareto front, 2) diversity metrics to evaluate the spread or distribution of the solutions in the found Pareto front, 3) combined metrics to evaluate both the closeness and diversity of the discovered solutions in an implicit manner, and 4) dynamic metrics to

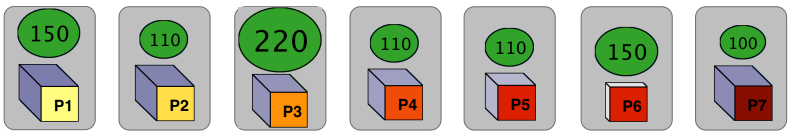


Figure 4.3: An example of Pareto optimal solutions visualized by VMODEX

show how the quality of solutions with respect to a specific metric is varying during the searching process.

Closeness Metrics

These metrics compute a measure of the closeness of a Pareto front found by an optimization algorithm (PF_{known}) from the true Pareto front (PF_{true}). PF_{true} can either be a set of (theoretical) true Pareto optimal points (if known) or a reference set, which contains the best-known non-dominated points from a combination of numerous runs.

Van Veldhuizen [84] proposed two metrics for measuring the distance of the found Pareto optimal set to the true Pareto front: Error Ratio (*ER*) and Generational Distance (*GD*). These metrics are widely used in literature because of their simplicity.

Error Ratio (*ER*) This metric indicates the proportion of solutions in PF_{known} that are not member of the PF_{true} as follows:

$$ER = \frac{1}{|PF_{known}|} \sum_{i=1}^{|PF_{known}|} e_{s_i} \qquad e_{s_i} = \begin{cases} 0 & s_i \in PF_{true} \\ 1 & s_i \notin PF_{true} \end{cases} \tag{4.1}$$

Where $||$ denotes the number of elements in a set and s_i indicates the i^{th} solution in the PF_{known}. A smaller value of *ER* means a better approximation of the true Pareto front. This metric takes a value between zero and one. $ER = 0$ means that all solutions of PF_{known} are member of the PF_{true} and $ER = 1$ means that no solution is a member of the true Pareto front. The drawback of this metric is that if no member of the PF_{known} is in the PF_{true}, it does not distinguish the relative closeness. Another drawback of this metric is illustrated in Figure 4.4. In this figure, algorithm A_1 finds only two Pareto optimal solutions, of which one is in PF_{true}, so its error ratio is 0.5. Algorithm A_2 finds 8 Pareto optimal solutions of which 3 of them are in PF_{true} and other solutions are relatively close to the true Pareto front. However, its error ratio is $5/8 = 0.625$, which is bigger than the error ratio of solutions in A_1. Clearly, the algorithm A_2 performs much better but the error ratio indicates that it performs worse. To overcome this drawback, we propose a new metric called Weighted Sum Generation Ratio (*WSGR*), as will be explained later in this section.

Generational Distance (*GD*) Instead of determining whether or not a solution of PF_{known} belongs to the PF_{true}, this metric finds an average distance of the solutions of PF_{known} from PF_{true}:

$$GD = \frac{1}{|PF_{known}|} \sqrt{\sum_{i=1}^{|PF_{known}|} {d_i}^2} \tag{4.2}$$

The distance measure (d_i) is the Euclidean distance (in the objective space) between the solution $s_i \in PF_{known}$ and the nearest member of PF_{true}. It is clear that a value

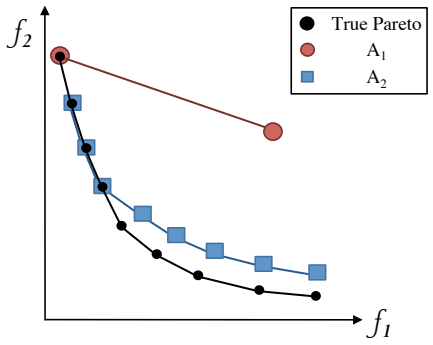


Figure 4.4: Illustration of the drawback of *ER*

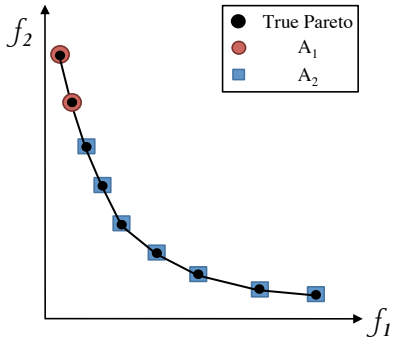


Figure 4.5: Illustration of the drawback of *GD*

of $GD = 0$ indicates that all the solutions of PF_{known} are in PF_{true}. A Pareto optimal set having a smaller value of *GD* is better. The drawback of this metric is that when all solutions in PF_{known} are in the true Pareto front ($GD = 0$), it does not determine how good the algorithm is in finding true Pareto points. This shortcoming is illustrated in Figure 4.5. In this figure, algorithm A_1 finds 2 Pareto optimal solutions of which both of them are in PF_{true} and algorithm A_2 finds 7 Pareto optimal solutions of which all of them are in PF_{true}. The value of *GD* for both of them is zero. However, it is clear that the algorithm A_2 performs better since it could find more true Pareto optimal points. Our proposed metric, *WSGR*, overcomes this drawback as well.

Weighted Sum Generation Ratio (*WSGR*) As we discussed above, both *ER* and *GR* metrics have some drawbacks. To overcome those shortcomings, we propose a new metric called *WSGR*. This metric combines the proportion of the true Pareto optimal solutions found by the algorithm with the proportion of solutions in PF_{known}, which are relatively close to the PF_{true}, via a weighted sum, as follows:

$$WSGR = \frac{w_1}{|PF_{true}|} \sum_{i=1}^{|PF_{true}|} f_{s_i} + \frac{w_2}{|PF_{known}|} \sum_{i=1}^{|PF_{known}|} c_{s_i} \tag{4.3}$$

$$f_{s_i} = \begin{cases} 1 & s_i \in PF_{known} \\ 0 & s_i \notin PF_{known} \end{cases} \qquad c_{s_i} = \begin{cases} 1 & d_i < \theta \\ 0 & d_i > \theta \end{cases}$$

Where $w_1 + w_2 = 1$ and d_i is the same as distance measure in the *GD* metric. For measuring the relative closeness, a threshold (θ) needs to be defined. If the minimum distance of solution $s_i \in PF_{known}$ from PF_{true} is less than the threshold, then it is considered as a relatively close solution and the value of its c_{s_i} is one. The weights w_1 and w_2 show the relative importance of two combined components of Equation 4.3. Since finding true Pareto optimal points has a higher priority than discovering

relatively close solutions, w_1 should be higher than w_2. A bigger value of ***WSGR*** is better since it indicates that the searching algorithm has obtained more solutions of PF_{true} and also more solutions in PF_{known} are close enough to the true Pareto front. Using each element of Equation 4.3 separately, as a performance metric, has some drawbacks. By considering only the first part, the value of the metric for two algorithms that could not find any solutions in PF_{true} is zero. So the metric does not distinguish the relative closeness. However, the solutions found by one algorithm may be much closer to PF_{true} than the solutions in another one. In the case of utilizing only the second term in Equation 4.3, the performance of two algorithms of which one finds solutions that are all in PF_{true} and the other one finds solutions of which all of them are close enough to PF_{true}, but none of them is member of PF_{true}, is the same. For both of them the calculated value is one. By combining these two components, the above shortcomings can be addressed.

To illustrate how our proposed metric can solve the drawbacks of ***ER*** and ***GD*** metrics, consider the examples in Figure 4.4 and Figure 4.5 again. In Figure 4.4, the performance of algorithm A_2 is better than A_1, in terms of closeness, while the error ratio indicates that the performance of A_2 is worse. With respect to our proposed metric, their performance can be calculated as follows ($w_1 = 0.6, w_2 = 0.4$):

$$WSGR(A_1) = 0.6 \times \frac{1}{9} + 0.4 \times \frac{1}{2} = 0.066 + 0.2 = 0.266$$
$$WSGR(A_2) = 0.6 \times \frac{3}{9} + 0.4 \times \frac{8}{8} = 0.198 + 0.4 = 0.598$$

Thus, the ***WSGR*** metric implies that solutions found by algorithm A_2 are closer to the true Pareto front. In Figure 4.5, algorithm A_2 has found more solutions of PF_{true} than A_1, but ***GD*** metric indicates that the performance of both algorithms, in terms of closeness, is the same. However, the ***WSGR*** metric implies that the performance of A_2 is better.

$$WSGR(A_1) = 0.6 \times \frac{2}{9} + 0.4 \times \frac{2}{2} = 0.132 + 0.4 = 0.532$$
$$WSGR(A_2) = 0.6 \times \frac{7}{9} + 0.4 \times \frac{7}{7} = 0.467 + 0.4 = 0.867$$

$WSGR = 1$ indicates that all the true Pareto optimal solutions are found by the optimization algorithm and all the other solutions in PF_{known} are close enough to the true Pareto front according to the distance threshold. $WSGR = 0$ means that none of the solutions in PF_{known} is close enough to the PF_{true} and thus the performance of the optimization algorithm, in terms of the closeness, is not desirable. Since we are interested in how well an algorithm performs, in the case of two algorithms with $WSGR = 0$ we can conclude that the performance of both of them is not acceptable with respect to the closeness aspect. It does not really matter which one performs worse. However, our visualization method for showing the closeness aspect enables us to do a more detailed analysis on the distance between solutions in PF_{known} and solutions in PF_{true}.

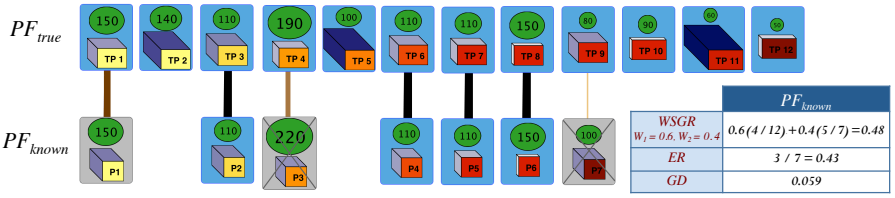


Figure 4.6: Visualization of the closeness

Visualization of the closeness We propose a visualization method that simply and clearly illustrates the closeness relations between PF_{known} and PF_{true}. Our visualization represents all the closeness metrics in a single view. It not only shows the values of different closeness metrics and how these values are achieved, but also represents some information about the outliers and fluctuation in the distance values (if they exist). One of the disadvantages of the *GD* metric is that if there is a large fluctuation in the distance values or there exist some outliers, the metric may not reveal the true distance. However, using our visualization, these properties can be recognized easily.

Figure 4.6 represents the visualization of the closeness aspect for the Pareto optimal set shown in Figure 4.3. The solutions in the true Pareto optimal set (or reference set if PF_{true} is not known) are shown in the first row and solutions of PF_{known} are shown in the second row. In the case of comparing the performance of different algorithms, for each of them, the solutions in its PF_{known} are drawn in a separate row. The background color of the true Pareto optimal solutions is blue. If a solution in PF_{known} is also a member of PF_{true} its background color is blue as well. Otherwise, a light grey background is used, and a cross is displayed at the dominated solution in PF_{known} to show that a solution in PF_{true} dominates this solution. However, if the distance of a dominated solution in PF_{known} is less than the defined threshold in the *WSGR* metric, then the cross is not displayed to indicate that the solution is close enough.

Each solution in PF_{known} is connected to the nearest solution in PF_{true}, in which the Euclidian distance (in the objective space) between them is minimum. This minimum distance is the same as the distance measure used in the *GD* and *WSGR* metrics. The color and thickness of each edge shows the distance between two connecting solutions. As the distance increases the edges become thinner and lighter. So the edges connecting true Pareto optimal solutions in PF_{known} to their corresponding solutions in PF_{true} are the thickest and darkest since the distance is zero. This edge visualization allows us to recognize the outliers and fluctuation in the distance values.

As can be seen in Figure 4.6, four members of PF_{known} are also in PF_{true}. For these solutions, the background color is blue and their edges are the thickest and darkest (black color). The other three solutions in PF_{known} are dominated by solutions in PF_{true}. Each one is connected to the nearest solution in PF_{true} that dominates it. The solution $P1$ in PF_{known} is close enough to PF_{true} according to the distance

threshold and therefore the cross is not drawn. The solution $P7$ in PF_{known} is an outlier since its distance value is too far from the other distance values. Its edge color and thickness differ a lot in comparison with other edges. It is much lighter and thinner than others. The visualization of the closeness aspect has following advantages:

- It enables us to find out which solutions in PF_{known} are dominated by which solutions in PF_{true}.
- It allows us to understand which true Pareto optimal solutions have been found by the optimization algorithm.
- Since for each solution its objective values are shown as well, it is easy to find out in which parts of the objective space the obtained solutions are near to the true Pareto front and in which parts they are far away. For example, from Figure 4.6 we can see that none of the low cost (less than 100) Pareto optimal solutions in PF_{true} have been found.
- It is possible to recognize outliers.
- It shows the fluctuation in the distance values.

Diversity Metrics

The Diversity metrics can be divided into two groups: 1) metrics evaluating the spread of solutions along the Pareto front and 2) metrics evaluating the distribution of solutions. In this section, we explain one metric for each group.

∇-Metric (for measuring spread) The ∇-metric [1] calculates the volume of a hyperbox formed by the extreme objective values observed in the Pareto optimal set:

$$\nabla = \prod_{m=1}^{M} \left(f_m^{max} - f_m^{min} \right) \tag{4.4}$$

Where M is the number of objectives, $f_m^{max}$and f_m^{min} are respectively the maximum and minimum values of the m^{th} objective in the Pareto optimal set. For two objective problems, this metric refers to the area of the rectangle formed by the two extreme solutions in the objective space, as shown in Figure 4.7.

A bigger value spans a larger portion and therefore is better. In the normalized version of this metric (as we use), if the value of the ∇ metric is one, then the widest spread set of solutions is obtained. This metric does not reveal the exact distribution of intermediate solutions, so we have to use another metric for evaluating the distribution.

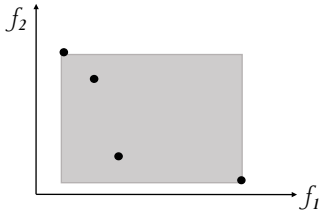


Figure 4.7: Illustration of the ∇-metric

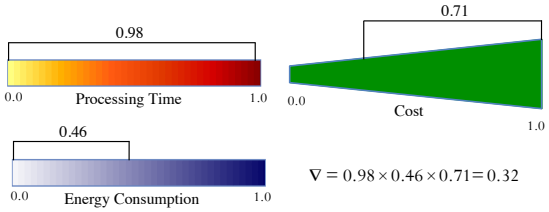


Figure 4.8: Visualization of the ∇-metric

Visualization of the spread For each objective, a horizontal axis from 0 to 1 is represented. Each axis is drawn in such a way that it demonstrates the same visual variable, which is used for showing that particular objective in the multi-objective DSE visualization. For example, in our case, colors from yellow to red are used for representing the performance of design points (in terms of processing time); therefore, this color scheme is used for coloring the corresponding axis in the spread visualization. However, for representing the cost of design points, instead of the color, the visual variable size is used. In the DSE tree, the size of the cost nodes becomes bigger as the cost increases. Thus, for demonstrating this visual variable in the cost axis, the height of the axis is increased by going from zero to one.

For each axis, the range between the minimum and maximum value of the corresponding objective in the Pareto optimal set is determined and shown above the objective axis. Figure 4.8 represents the visualization of the spread for the Pareto optimal set shown in Figure 4.3. As can be seen in this figure, the Pareto optimal set has an almost perfect extent in processing time and its spread in terms of cost is fairly good. However, only a small portion of the energy consumption is covered by the solutions in this Pareto optimal set. As a result, the value of ∇-metric is relatively small and indicates that this set does not have a good extent. The spread visualization has the following advantages:

- It is not limited to the number of objectives. Only one axis is added for each objective.
- For each objective, it can easily be seen that solutions are located in which part of the objective space. Both in terms of spread width (difference between the minimum and maximum) and objective values (the value of the minimum and maximum). For example, in Figure 4.8 the spread width of the energy consumption is small. However, the covered portion is located near the optimum part of this objective space.

$\sigma_{\mathbf{mst}}$-Metric (for measuring distribution) Most of the metrics suggested for measuring the distribution of solutions, are only applicable for problems with a two-dimensional objective space. It is not possible to use them in higher dimensions so

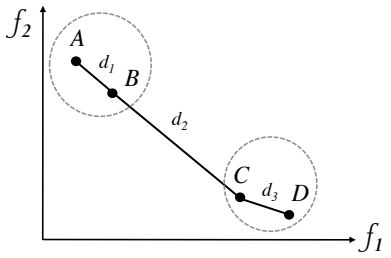


Figure 4.9: Illustration of the drawback of *SS*-metric

easily. Most of these metrics, such as the Δ [1] and Δ' [19] metrics, are based on calculating the distance between consecutive solutions. However, the concept of consecutive solutions in higher dimensional spaces is ambiguous. Our proposed metric for measuring the distribution (σ_{mst} metric) can easily be used for any number of objectives.

In [85], the well-known Schotts Spacing metric (SS) is proposed which tries to assess how evenly the non-dominated solutions are distributed. It is based on computing the shortest distance between solutions. The drawback of this metric is that in the case that solutions are clustered in small groups along the Pareto front, the distance between the groups are not considered since only the shortest distances are computed. This situation is illustrated in Figure 4.9. To calculate the SS metric, the distance between A and B (d_1) and the distance between C and D (d_3) are used twice, while the distance between B and C (d_2) is disregarded. As a result, it considers some information more than once (distance between solutions inside a group) while ignoring some useful distribution information such as the distance between the groups. Therefore, for measuring the distribution of solutions in a Pareto optimal set, we propose a new metric, called σ_{mst}, which is the standard deviation of the edges weights in the Minimum Spanning Tree (MST) generated by Pareto optimal solutions. The weights of the tree edges are the Euclidian distances (in the objective space) between solutions.

A minimum spanning tree is a subgraph of a weighed graph, which is a tree and contains all of the graphs nodes and a subset of its edges, such that all nodes are connected and the total weight of the edges is minimal. In Algorithm 4.1, we define the procedure of constructing an MST from a Pareto optimal set. When the MST is made, σ_{mst} can be computed using Equation 4.5.

$$\sigma_{mst} = \sqrt{\frac{1}{|E|-1}\sum_{i=1}^{|E|}\left(\bar{w}-w_i\right)^2} \tag{4.5}$$

Where $|E|$ is the number of edges in the MST, w_i is the weight of the i^{th} edge and $\bar{w}$ is the average weight of the edges in the MST. The σ_{mst} metric measures the standard

Algorithm 4.1 Constructing MST from a Pareto Set

Input: A set of Pareto optimal solutions
Output: MST

1. Compute the Euclidian distance (in the objective space) between any two solutions in the Pareto optimal set
2. Create a fully connected weighted graph (G) in such a way that each solution in the Pareto optimal set indicates a node in the graph G and the edge weight between two nodes is the Euclidian distance between the corresponding solutions
3. Generate a minimum spanning tree for the graph G (Prim's algorithm [86]):
   a) Let MST be an empty tree
   b) Select a random node in G and add it to MST
   c) While MST has fewer nodes than G do:
      i. Find the smallest edge connecting a node in MST to a node in G-MST
      ii Add the corresponding edge and node to the MST

deviation of the edges weights in the MST. The edges weights denote the minimum distances between connecting solutions. Therefore, a smaller value indicates that the distribution of solutions is closer to the uniform distribution and thus is better.

In Algorithm 4.1, we use Prim's algorithm to calculate the MST. Prim's algorithm is a deterministic greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph. It (greedily) builds the spanning tree by adding at each step the minimum weight edge that has exactly one endpoint in the MST. One can implement this algorithm efficiently by maintaining a heap of the edges incident to the MST that have not been added yet, ordered by weight. Every time a new vertex is added, the heap of candidate edges is updated with the incident edges that the new vertex introduces. If the graph has E edges and V vertices, the running time is $O(E\ log\, V)$. Alternatively, Prim's algorithm can be implemented using Fibonacci Heaps [87] to obtain $O(E + V\ log\, V)$ complexity. As it is explained in Algorithm 4.1, we work with fully connected graphs. Thus, we use Prim's algorithm as it is more efficient for dense graphs. However, other algorithms for computing the minimum spanning tree such as Kruskal [88], Boruvka [89], etc. can be used. Figure 4.10 illustrates how Prims algorithm constructs the MST from an original weighted graph.

Visualization of the distribution We propose a visualization method, which clearly shows how discovered Pareto optimal solutions are distributed in the objective space. For visualizing the distribution, the constructed MST is drawn in such a way that the length of the edge between two nodes (solutions) represents the edge weight (the

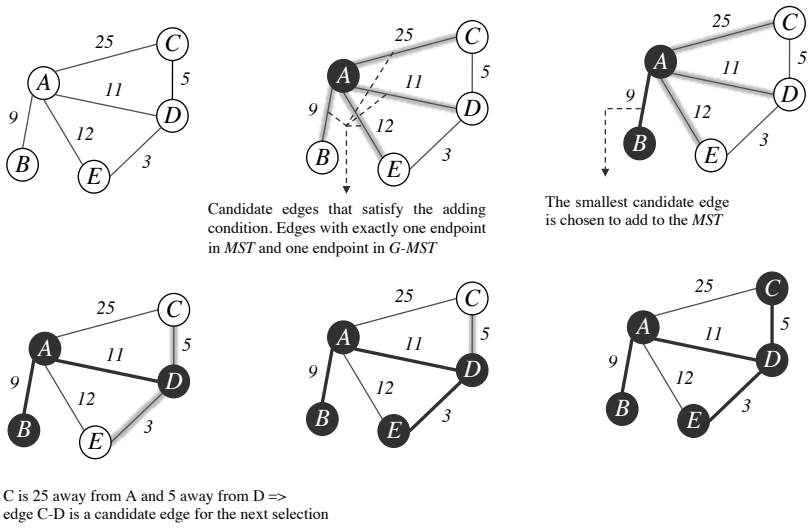


Figure 4.10: An example that shows how the Prim's algorithm works

Euclidian distance between two solutions). A longer edge implies a larger distance. Therefore, if in a Pareto optimal set all the edges have almost the same length, then this means that the solutions are distributed (nearly) uniformly. The same as in the DSE tree, the objective values of solutions (nodes in the MST) are shown by node attributes.

For better viewing and analyzing the distribution of solutions, it is possible to cluster the nodes (solutions) in the MST according to their edge weights (distance). If the distance between a solution and its parent is less than a certain threshold (determined by the user) then it is in the same cluster as its parent. Otherwise, it becomes a member of a new cluster. The solutions in the same cluster have the same background color. A Pareto optimal set with a smaller number of clusters has a better distribution. For a better view, the edges connecting two different clusters are drawn by dashed lines.

Figure 4.11 shows the visual representation of the distribution for the Pareto optimal set shown in Figure 4.3. In this figure, $\bar{w} + \sigma_{mst}$ is chosen as a threshold for clustering the solutions. Here, the solutions are distributed into two clusters, since there are two different background colors. Excluding the solution labeled by "P3", all the other solutions are in the same cluster, which means that the distance between them is less than the threshold and therefore their distribution is fairly uniform. However, the large distance between P3 and its parent causes the value of the σ_{mst} metric to become slightly bigger and thus indicates that the distribution of solutions is suboptimal. Using distribution visualization has the following advantages:

- For each two connected solutions, besides the distance value, the amount of difference between their objective values can be seen. Therefore, it is easy

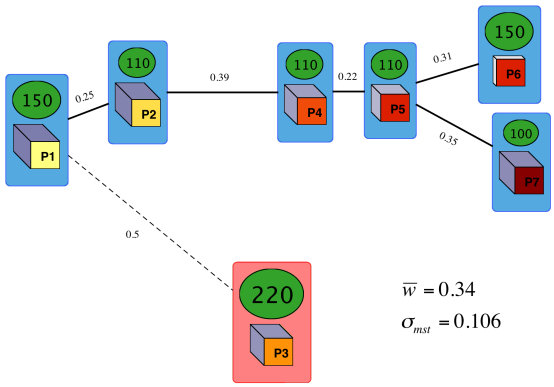


Figure 4.11: Visualization of the σ_{mst}-metric

to find out which objective value(s) have a high impact on the distance value. For example, in Figure 4.11, solutions P2 and P4 have exactly the same cost and their energy consumption is almost the same too. But their difference in processing time is significant. Thus this objective has the highest impact on the distance value.

- It can easily be used for problems with more than three objectives because objective values are shown by node attributes.
- By clustering the solutions, it is easy to understand which parts of the objective space are properly covered by well-distributed Pareto optimal solutions and in which parts the coverage and distribution is poor.

Combined Metrics

The combined metrics provide a measure of closeness as well as diversity in an implicit manner. Here we explain two combined metrics: weighted sum and hypervolume.

Weighted Sum A simple way to evaluate both goals with a single metric is to define a weighted sum metric, which combines one or more closeness metric(s) with one or more diversity metric(s), such as follows:

$$WS = w_1 GD + w_2\, \sigma_{mst}\,, \quad where \quad \sum_{i=1}^{N} w_i = 1 \tag{4.6}$$

Where N is the number of combined metrics. The user can choose appropriate weights (w_1 and w_2) for combining metrics. Here, we have combined the generational distance

(*GD*) metric for measuring the closeness with the σ_{mst} metric for evaluating the diversity. As we described before, *GD* takes a small value for a good closeness and σ_{mst} takes a small value for a good distribution. Therefore, a set of Pareto optimal solutions having an overall small value of *WS* indicates that both goals are satisfied. Since this metric does not provide a new measurement and only combines other existing metrics, a new visualization approach is not needed. We can simply use the visualization of each combined metric separately.

Hypervolume (*HV*) This metric [79, 80] measures the hypervolume of the objective space covered by members of a Pareto optimal set and a reference point. The hypervolume represents the size of the region dominated by the solutions in the Pareto optimal set. In Figure 4.12, the gray region represents this metric for two objectives (f_1 and f_2) where these objectives are to be minimized. The reference point (W) can simply be found by constructing a vector of worst objective values. A Pareto optimal set with a large value for the hypervolume is desirable.

The hypervolume metric is interesting because it is sensitive to the closeness of solutions to the true Pareto optimal set as well as the diversity of solutions across the Pareto front. Figure 4.13 illustrates this property of the hypervolume metric. In this figure, the dark gray region represents the hypervolume metric for the true Pareto front (PF_{true}) and the light gray area shows this metric for the Pareto optimal solutions found by an optimization algorithm (PF_{known}). In Figure 4.13(a), the closeness of solutions in PF_{known} is good (all of them are in PF_{true}), but the diversity of solutions is poor. As a result, the dominated region in PF_{known} is less than the dominated region in PF_{true}. In Figure 4.13(b) the situation is reversed. The diversity of solutions in PF_{known} is good. However, they are relatively far away from the PF_{true}. Consequently, less area in objective space is dominated by solutions in PF_{known}. Therefore, the hypervolume metric captures in a single scalar both goals of multi-objective optimization in the objective space. The hypervolume value is calculated by summing the volume of hyper-rectangles constructing the hypervolume. In Figure 4.12, dashed lines separate the hyper-rectangles, which are rectangles in two objective problems.

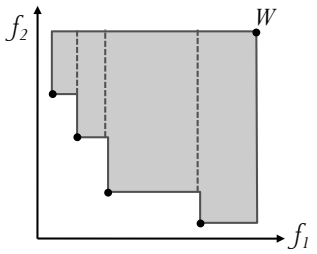
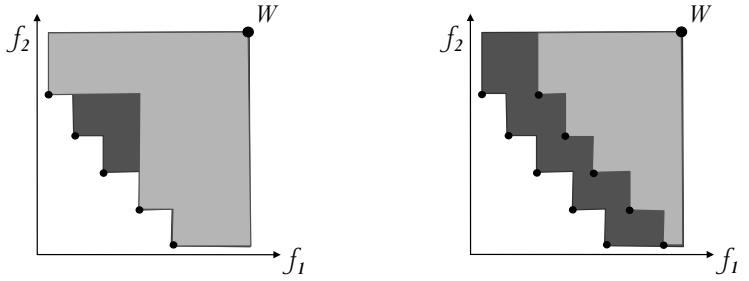


Figure 4.12: Illustration of the Hypervolume metric



(a) The effect of diversity on *HV* (b) The effect of closeness on *HV*

Figure 4.13: Illustrating the sensitivity of Hypervolume metric to both closeness and diversity

Visualization of the Hypervolume As we mentioned before, one of the main challenges of proposing a visualization method for a performance metric is that it can be used for problems with more than three objectives. The visualization of the hypervolume metric, which is typically used in literature, is limited to three objectives. However, we propose a new visualization method, which is not restricted to the number of objectives. To do that, we divide the *m*-dimensional objective space to *m-1* two-dimensional spaces. One certain objective (determined by the user) is chosen as a base and is a member of all *2D* spaces, while all the other objectives are in separate *2D* spaces. For each two-dimensional space, an *x*-*y* graph is drawn in which the *x*-axis shows the base objective and therefore is the same in all graphs and the *y*-axis represents the other objective. In each *x*-*y* graph the colored region shows the dominated part. As an example, consider a problem with three objectives: f_1, f_2 and f_3. We divide this 3-dimensional objective space to two *2D* spaces and represent them with two *x*-*y* graphs: f_2 versus f_1 and f_3 versus f_1. Here, f_1 is chosen as the base objective.

By dividing the objective space to two-dimensional spaces, each *m*-dimensional hyper-rectangle in the hypervolume region, is divided into *m-1* rectangles (two dimensional hyper-rectangle). Each rectangle is drawn in its corresponding *x*-*y* graph. The base objective (*x*-axis) constructs the width of the all *m-1* rectangles and each of the other objectives constructs the height of its corresponding rectangle (*y*-axis). To distinguish the divided rectangles of a particular hyper-rectangle in different *x*-*y* graphs, they are colored with the same color in all graphs. If two or more hyper-rectangles have overlap in some spaces, textures are used. Each hyper-rectangle has a specific texture. Therefore, the overlapping areas contain several textures. This overlapping is caused because some dimensions of hyper-rectangles are removed in *x*-*y* graphs.

Figure 4.14 shows an example of hypervolume visualization for a Pareto optimal set containing three solutions: $S_1 = \{0.1, 0.5, 0.2\}$, $S_2 = \{0.3, 0.3, 0.4\}$, $S_3 = \{0.5, 0.1, 0.1\}$. Small black circles in the graphs display the Pareto optimal solutions. The reference point (W) is $\{1, 1, 1\}$. As can be seen in this figure the hypervolume region consists of four hyper-rectangles, which are denoted by numbers from 1 to 4. For each hyper-rectangle, the values of its axis in each dimension and its volume are

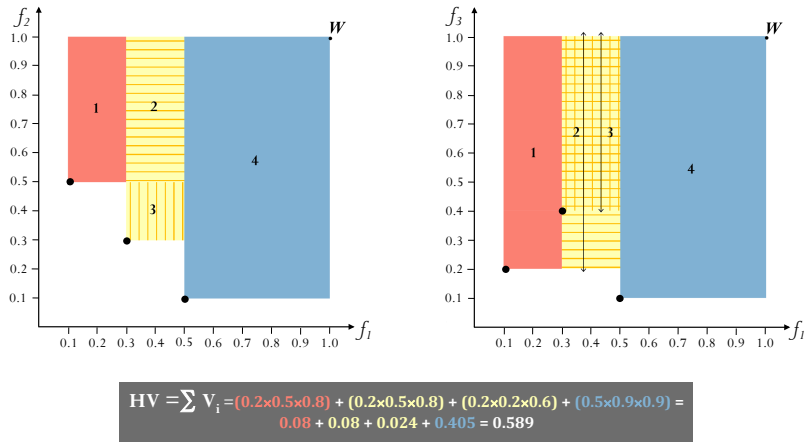


Figure 4.14: Visualization of the Hypervolume metric

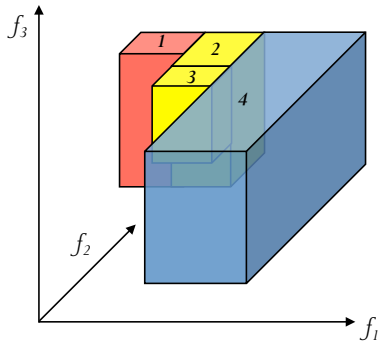


Figure 4.15: Visualization of the Hypervolume metric in *3D* space

written at the bottom of the visualization with the same color as its corresponding rectangles in *x*-*y* graphs. For instance, in Figure 4.14, the volume of the hyper-rectangle denoted as 1 is $0.2 \times 0.5 \times 0.8 = 0.08$

The second and third hyper-rectangles have overlap in the f_3 versus f_1 graph. Therefore, textures are used. The texture of the second hyper-rectangle is horizontal lines and the texture of the third hyper-rectangle is vertical lines. Thus, the overlapping area has both textures. To illustrate the reason of this overlap, Figure 4.15 shows these hyper-rectangles in *3D* space. As can be seen in this figure, the hyper-rectangles 2 and 3 have exactly the same width on the f_1 axis and completely different depth on the f_2 axis. However, their heights on the f_3 axis are common in the top part. Thus, by removing the f_2 axis in the $f_1 - f_3$ graph, hyper-rectangles 2 and 3 will have overlap in the common part in f_3. However, by utilizing textures, this overlap can be seen.

The hypervolume value is calculated by summing the volume of hyper-rectangles, as shown in Figure 4.14. This type of visualization enables us to see the dominating area of each objective surface separately. Since the *x*-axis is the same in all surfaces, the comparison between them can be made easily. For instance, in the example shown in Figure 4.14, we can see that the size of the dominating region in the $f_3 - f_1$ surface is bigger than the dominating area in the $f_2 - f_1$ surface.

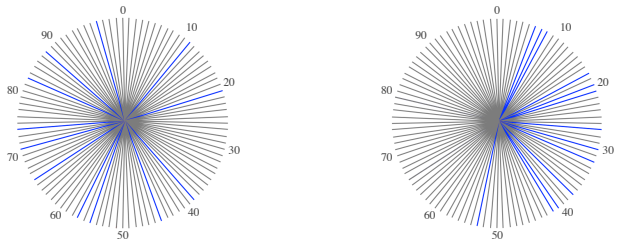
Dynamic Metrics

Previous metrics described in this section are used to assess the quality of the obtained Pareto optimal set at the end of the searching process. However, dynamic performance metrics show how an optimization algorithm achieves the final quality during its execution. They illustrate how the quality of solutions is varying during consecutive iterations. Such information provides insight in the working of the algorithm and allows detailed evaluation of the strengths and weaknesses of the algorithm. Moreover, these metrics are especially useful when the quality of the end results of compared algorithms do not differ significantly; however, the way that these results are achieved may provide valuable information such as how fast a certain searching algorithm converges to the Pareto optimal set. The output of dynamic metrics is not a single value. Instead, a set of values is computed that shows the dynamic behavior of the algorithm. In VMODEX, three dynamic metrics are provided: 1) dynamic hypervolume, 2) dynamic closeness, and 3) the dynamic process of finding final Pareto optimal solutions.

Dynamic Hypervolume (*DH*) This metric shows how the value of the hypervolume metric evolves over the iterations. If an algorithm can reach to a desirable hypervolume with fewer evaluations, this means its optimization speed is faster and therefore its performance is better. As we mentioned before, the hypervolume metric measures both the closeness and diversity of obtained solutions in an implicit manner. Therefore, this metric provides an overall evaluation of how an optimization algorithm finds better approximations of Pareto optimal solutions during its execution. For showing this metric, the hypervolume value is drawn versus the generation numbers. There is an example of this metric in the next section for evaluating our case study results.

Dynamic Closeness (*DC*) This metric is similar to the dynamic hypervolume, except that one of the closeness metrics is used. Therefore, it shows how fast the found Pareto front converges towards the true Pareto front.

Dynamic of Finding Pareto Optimal Solutions (*DFPOS*) We propose a new dynamic metric, called *DFPOS*, which shows the progress of the algorithm in finding the final Pareto optimal solutions. A final Pareto optimal solution is a solution that will not be dominated by any other solution during the searching process. This metric represents a set of generation numbers in which a final Pareto optimal solution is found for the first time. Thus, it is easy to understand in which generations the algorithm



(a) (b)

Figure 4.16: Visualization of the *DFPOS* metric

has found a new final Pareto optimal solution. Note that this metric can be calculated when the optimization process is terminated.

We also propose a visualization method for showing the results of the *DFPOS* metric, which is shown in Figure 4.16. For each generation a line is drawn. If within a generation a new final Pareto optimal solution is found, then the color of the line representing that generation is blue. Otherwise, a gray line is used. For better viewing, these lines are shown as radiuses of a circle in which the angle between any two lines is the same. In compare to the simple strip, the circular visualization can better represent the distribution of blue lines (generations in which a final Pareto optimal solution is found) in the entire generations. In the circular visualization, the fractions of generations can be seen much better.

The benefit of the *DFPOS* metric is illustrated by the example shown in Figure 4.16. In Figure 4.16(a), the optimization algorithm has found some new Pareto optimal solutions in the last generations. So, it is more likely to find other Pareto optimal solutions by running this algorithm for more generations and improve the quality of the obtained solutions. However, in Figure 4.16(b) there is no new Pareto optimal solution after the 54^{th} generation. Therefore, by evaluating more solutions, the algorithm will probably not find a new Pareto optimal solution and the quality of the results will not improve. Thus, we can reduce the number of generations in order to decrease the computational cost.

4.3.2 Performance Metrics in the Decision Space

In recent years, many new optimization methods and algorithms have been proposed to improve the performance of optimization process. Subsequently, various measures have been suggested to assess the quality of their results from different perspectives. However, most of the current performance measures concentrate on evaluating the quality of found Pareto optimal solutions in the objective space and measuring the behavior of the optimization algorithm in the decision space has mostly been disregarded. However, turning the focus of attention from exclusively evaluating optimization success in the objective space to also considering the decision space is essential

to be assured of: a) finding distinct optimal solutions (in the design space) which have similar quality in the objective space, and b) obtaining global optimum solutions in the case of existing multimodal objective functions. A multimodal function is a function with many local maxima. Furthermore, understanding the distribution of Pareto optimal solutions in the decision space can provide guidelines on selecting the most suitable solutions, which would eventually be implemented.

In this thesis, we define some new metrics to assess the performance of optimization algorithms in the decision space. In the following subsections, we categorize these metrics into three groups: 1) decision space coverage metrics to assess the diversity or distribution of the evaluated design points in the decision space, 2) unique ratio metric to evaluate the performance of the searching algorithm in finding unique design points in the decision space, and 3) dynamic metrics to show how an optimization algorithm explores and covers the design space during its consecutive iterations.

Decision Space Coverage Metrics

Decision space coverage metrics can be divided into two groups: 1) metrics that evaluate the variety of evaluated solutions with respect to different design parameters, and 2) the distribution of the found solutions in the decision space. In this thesis, we define one metric for each group.

Covered Sets Ratio (*CSR*) It is essential that a searching algorithm is able to find solutions in as many as possible different regions of the decision space, even if those solutions are not Pareto-optimal. In this case, the exposed solutions represent the variety of possible designs in the decision space and enable the designer to do a more comprehensive study on the relationship between design parameters and their effects on the design criteria. Furthermore, a higher diversity and coverage in the decision space means that there is a higher chance of finding the global optimal solutions rather than the local optimum ones (the algorithm does not get trapped in local optimum points in the design space). The spread of design points in the design space can help the algorithm to escape from such local optima. For measuring the decision space coverage, we define the Covered Sets Ratio (*CSR*) metric as follows:

$$CSR = \frac{|Searched\ Combinations|}{|Total\ Feasible\ Combinations|} \tag{4.7}$$

Where $||$ denotes the number of elements in a set. This metric simply calculates the proportion of the possible combinations of design parameters, which are visited by the searching algorithm. A bigger *CSR* value indicates that more distinct regions of the design space are searched and therefore is better. This metric is applicable for discrete decision spaces. In the case that design space parameters are continuous variables, we can divide the design space into cells. Each of k parameters can be divided into m bins of equal size, yielding m^k cells or hypercubes. Each cell that contains at least one design point is considered as a searched cell. A variant of the *CSR* metric, which

is applicable for continuous design spaces, can be expressed as follows:

$$CSR = \frac{|Searched\ Cells|}{|Total\ Cells|} \tag{4.8}$$

Visualization of the CSR For more detailed information about the parts of the design space covered/not covered by the optimization algorithm, the designers can use our visualization tool. Our DSE tree visualization clearly shows which parts of the design space are not searched at all (no design point is evaluated there) and which parts contain evaluated solutions.

Figure 4.17 shows an example design space, which is visualized by VMODEX. In VMODEX, each parameter of the design space in shown as one level in the DSE tree. In this example, the design space has four parameters, which (from top to bottom) are: number of processors, processor type, number of memories and memory type. In this figure, only the parameters segment of the DSE tree is shown. Each architecture platform instance is indicated by a unique combination of these parameters. Those combinations that lead to the platform instances, which are capable of executing the application are denoted as *"Feasible Combinations"*. In the DSE tree only the feasible combinations are shown. In the tree shown in Figure 4.17, there are 11 feasible combinations. The number of nodes at the last level of the parameters segment (memory type level in Figure 4.17) indicates the number of feasible combinations. In the DSE tree, those parameter combinations that are not visited by the searching algorithm are represented by a white color and dashed lines. In Figure 4.17, only 4 combinations are searched while 7 architecture instances are not visited and therefore there is no evaluated design point there. Thus, the *CSR* value for this example is $4/11 = 0.36$.

Since each level in the DSE tree (in the parameters segment) indicates one design parameter, the *CSR* metric can be applied for each parameter of the design space separately. The coverage issue we discussed above considers the entire design space and therefore is assigned to the root node. However, we are also able to see the coverage of internal nodes in the parameter segment of the DSE tree. For instance, in Figure

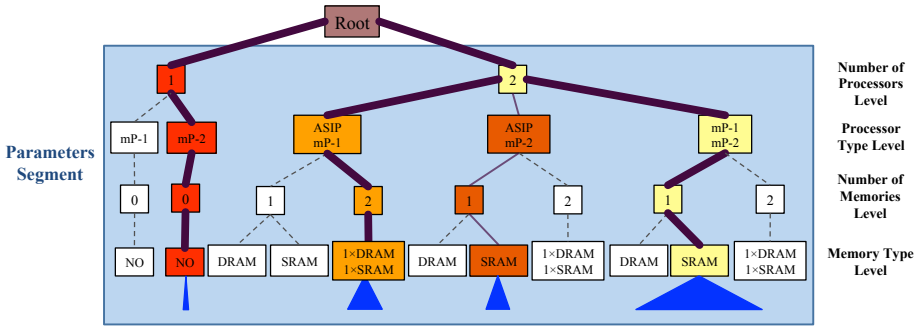


Figure 4.17: an example design space visualized by VMODEX

4.17, the coverage of two-processors architecture platforms is $3/9 = 0.33$ (in total there are 9 different architecture instances containing two processors and only three of them are searched) while the coverage of single processor platforms is $1/2 = 0.5$.

Distribution of Design Points in the Decision Space For analyzing the distribution of design points in the design space, we divide the design space into exclusive subspaces. Each subspace represents a partial instantiation of the design variables and includes a set of design points with similar characteristics. A uniform distribution is desirable and obtained by having almost the same number of design points in all subspaces. For instance, it would not be appropriate to have 80% of the evaluated design points in only one subspace and 20% in all the other subspaces.

We utilize the *box-percentile plot* [90] to summarize the distribution information and provide easy comparison between the distributions of the results of different searching algorithms. The box-percentile plot is a modified version of the well-known box plot. A box-percentile plot gives the same graphical impression as a box-plot, but also provides a far more informative view of the data distribution. The box plot summarizes the distribution using five statistical measures, which consists of the minimum data value, 25^{th} percentile, 50^{th} percentile (median), 75^{th} percentile, and maximum data value. Although this information is helpful and quickly expresses the general characteristics of the data set, it is difficult to assess the full-range frequency distributions from box plots. Some anomalies may not be detected in box plots. For instance, box plots hide the modality of a distribution, and different distributions with varying modality may be encoded in similar box plots. Modality measures the number of major peaks in a distribution. One solution to overcome these kinds of problems is to add the density information of distribution into the box plot. The Box-percentile plot is a simple approach that adds the empirical cumulative distribution of the dataset into the box plot.

Unlike the box plot, which has a fixed width, the box percentile plot uses width to encode information about the shape of the distribution. For each position in the plot, up to the 50^{th} percentile, the width of the irregular box is proportional to the percentile of that data value. For positions above the 50^{th} percentile, the width is proportional to 100 minus the percentile. Thus, the width at any given position is proportional to the percent of observations that are more extreme in the direction leading away from the median. Therefore, a box-percentile plot is wide in the middle, narrow away from the middle and very narrow at the extremes. As in box plots, the median, 25^{th} and 75^{th} percentiles are marked with lines across the box. Actually, the box-percentile plot combines the benefits of box plots, which are easy to interpret for non-expert users and several data sets can be compared simultaneously, with percentile plots which display all the data. Thus, there is no loss of information due to the grouping. The typical constructions of the box plot and box-percentile plot are shown in Figure 4.18.

To illustrate the effectiveness of the extra information added to the box-percentile plot in comparison to the box plot, consider the two different distributions shown as histograms in Figure 4.19. Figure 4.19(a) shows a trimodal distribution. In this figure

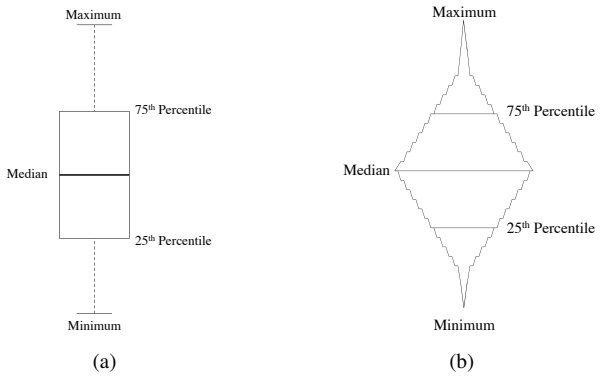
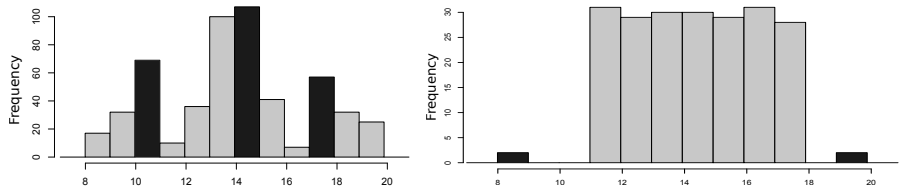


Figure 4.18: The anatomy of (a) box plot (b) box percentile plot

black bars demonstrate the three major peaks. The distribution in Figure 4.19(b) is a uniform distribution with two extreme outliers, one on each side. Values that are sufficiently far from the central part of the data are defined as outliers. In Figure 4.19(b) black bars represent outliers. As shown in Figure 4.20, the box plots of these two different distributions are similar. Just by looking at the box plots, it is not possible to distinguish any difference in their distributions. However, their box-percentile plots, shown in Figure 4.21, are completely different and allow the observer to recognize the predominant pattern of distribution.

In Figure 4.21, the two outliers in the uniform distribution can easily be identified. Outliers cause a long, thin line leading from the main body of the box-percentile plot to the outliers. The main body of the plot indicates a uniform distribution since the sides are straight and has the characteristic of diamond-like shape (the percentile plot of a uniform distribution is linear). In the trimodal box-percentile plot in Figure 4.21, the vertical lines in the outline of the box illustrate the typical feature of a multi-modal distribution. The valleys between modes have fewer observations relative to the peaks, so there is little change in the percentiles in those regions, which are translated into flat, near vertical lines. The typical pattern of a box-percentile plot for skewed data is shown in Figure 4.22. This figure shows a right (positive) skew distribution. As we illustrated in this section, the box-percentile plot provides a means to easily and quickly characterize distribution patterns and is straightforward enough to be understandable by non-expert readers.

Visualization of the Distribution The DSE tree visualization can be used to do more detailed analysis of the distribution of solutions in the decision space. It clearly shows which parts of the design space contain more design points and in which parts only a few number of solutions are evaluated. By modeling the design space as a tree, it is divided in several subspaces. Each subspace represents a unique combination of design parameters (in our case, a unique instance of the architecture platform).



(a) Trimodal distribution (b) Uniform distribution with two outliers

Figure 4.19: Histograms of two different distributions

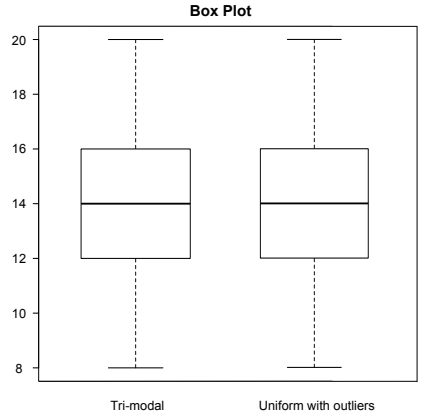


Figure 4.20: Box plots for the distributions shown in Figure 4.19

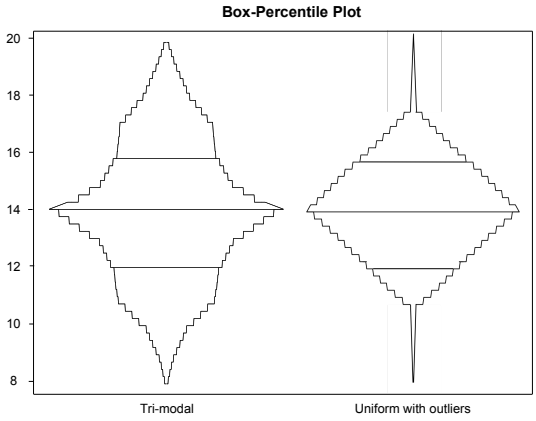


Figure 4.21: Box percentile plots for the distributions shown in Figure 4.19

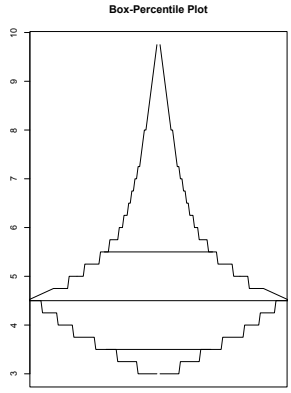


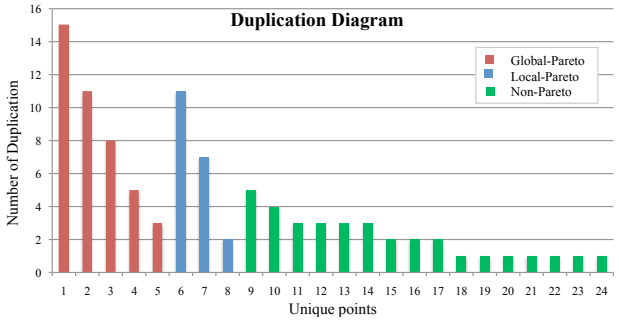
Figure 4.22: A box percentile plot showing the typical pattern for skewed distributions

On the other hand, solutions inside a subspace have exactly the same architecture components but the way that the application is mapped onto those components is different. The distribution of the design points can easily be seen in the DSE tree. It clearly shows how many design points each subspace contains. For a subspace with more design points, the DSE tree creates more nodes and therefore its corresponding sub tree becomes bigger. For better understanding the distribution, the designer can hide the design point nodes in each subspace (collapse its subtree) to make the tree smaller and focuses only on those levels of the tree that show the design parameters (i.e. parameters segment). In this case, for each subspace, the size of the blue triangle represents the number of evaluated design points (see Figure 4.17). The searching algorithm does not visit subspaces without a blue triangle at all. The bigger the triangle means that there are more nodes in the subtree. For example, in Figure 4.17, we can see that the subspace that consist of two microProcessors (mPs) and one SRAM memory includes the highest number of design points since its blue triangle is the biggest. Thus, using the DSE tree, we can evidently see the distribution of design points in the decision space.

Unique Ratio Metric

During the process of design space exploration using a heuristic searching technique, some design points that have a good performance may be regenerated in different generations. Therefore, there might be some duplicate design points in different generations. An algorithm with less duplication has better performance because a greater number of design points can be investigated with the same computational effort. For measuring this performance aspect, we propose the Unique Ratio (*UR*) metric as follows:

$$UR = \frac{|Unique\ evaluated\ design\ points|}{|Total\ evaluated\ design\ points|} \tag{4.9}$$



$UR = 24 / 100 = 0.24$

Figure 4.23: An example of duplication diagram

A greater *UR* value denotes that more distinct design points are investigated and therefore is better.

Visualization of the *UR* To understand the general trend of design point duplications, we provide a bar chart plot, called *Duplication Diagram*. In this diagram, there is a bar for each unique design point and the lengths of the bars show the number of duplications. Figure 4.23 shows an example of a duplication diagram. In the DSE tree, design points are categorized in three groups (global Pareto, local Pareto and non-Pareto). Therefore, in the duplication diagram, each group is differentiated by a separate color. Thus, the designer is able to compare the duplication ratio among these groups. In each group, design points are sorted by their duplication number in descending order.

If the designer is interested to know more about some design points with a particular duplication number (e.g. highest duplication), he can select them in the duplication diagram and then the corresponding solutions in the DSE tree are highlighted. Therefore, the designer can easily see the characteristics of those solutions such as their design parameters, objective values, generation numbers, etc.

Dynamic Metrics

Since exploring the design space is an iterative process, it is important to know how a searching algorithm walks through the design space and trace its progression in finding new design points and covering the design space during successive iterations. As we explained in Chapter 3, Section 3.2.5, in VMODEX, we provide an interactive user interface, called step-by-step animation, which allows designers to follow the evolutionary exploration process during numerous generations. However, to summarize the dynamic exploration process of the design space and also for comparing the dynamic behavior of different searching algorithms, we have proposed two dynamic metrics: 1) the dynamic access to the new parts of design space and 2) dynamic unique ratio.

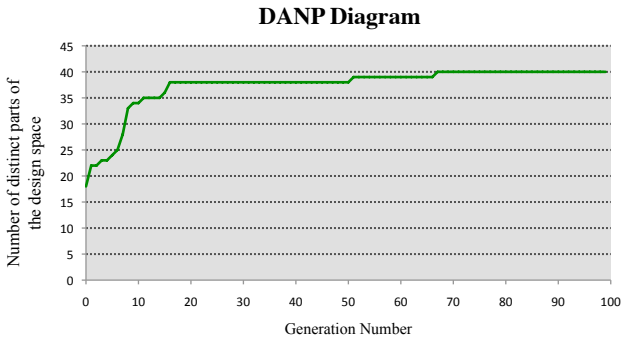


Figure 4.24: An example of *DANP* diagram

The output of the dynamic metrics is not a single value. Instead, a set of values is computed that shows the dynamic behavior of the algorithm during its execution.

Dynamic Access to the New Parts (*DANP*) This metric shows how the searching algorithm accesses to the new parts of the design space over the generations. For measuring this metric, the design space is divided into exclusive parts (subspaces). Each subspace represents a partial instantiation of the design variables. Each part of the design space that contains at least one design point is considered as a visited part. To make performance metrics consistent, the same approach should be used for partitioning the design space in all metrics. Thus, the same subspaces are considered for assessment of all metrics. Figure 4.24 represents an example of this metric. The *x*-axis shows the generation numbers and the *y*-axis denotes the number of distinct parts of the design space that have been visited by the algorithm until the corresponding generation number. Generation zero indicates the initial population, which has been generated randomly. In Figure 4.24, until the 20^{th} generation, the algorithm has a good performance in discovering new parts in the design space. However, after the 20^{th} generation, only two new parts are found.

Dynamic Unique Ratio (*DUR*) This metric represents the varying of the unique ratio during the search generations. Figure 4.25 represents an example of this metric. The *x*-axis shows the generation numbers and the *y*-axis denotes the ratio of the evaluated design points (in corresponding generation number) that are new and have not been found by the algorithm in the previous generations. Although generally we expect a smaller unique ratio in the later generations, the slope of the changes in *y*-values is quite important as it implicitly indicates the performance of the searching algorithm. A gentle slope is better since it means that there is a more gradual deterioration in the rate of finding new design points during the generations. In Figure 4.25, the unique ratio for the initial population (generation zero) is 0.92 (there are only a few duplicate points in the same generation). However, there is a big degradation in unique ratio in

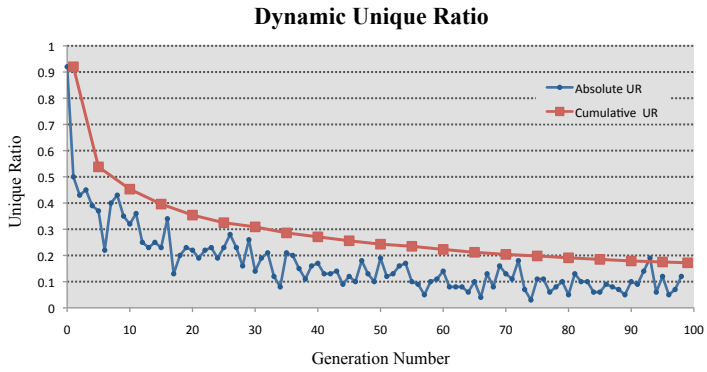


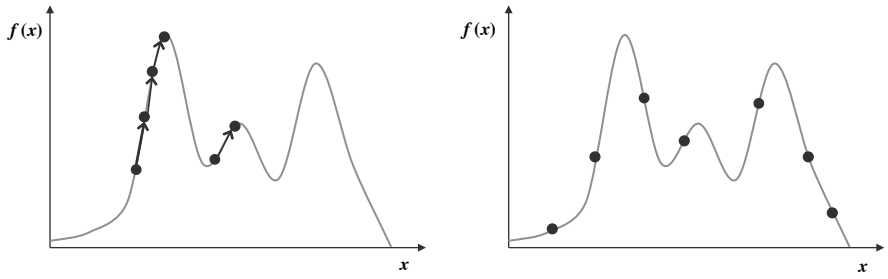
Figure 4.25: An example of *DUR* diagram

the next generation (0.5). But after that it is changed more smoothly. In Figure 4.25, the red plot represents the cumulative unique ratio.

4.4 A Case Study

Because in real problems the design space often is extremely large, it is not possible to exhaustively explore the design space, and evaluate and compare every single point (possible combination of design parameters) in this space. Therefore, several exploring strategies have been suggested, which iteratively walk through the design space and try to obtain a proper coverage of the design space during their explorations. These methods can be generally divided into two types, based on their progress from iteration to iteration: guided search and unguided search. The guided search methods, such as hill climbing [28, 29], evolutionary algorithms [30, 31], ant colony optimization [32–34] and simulated annealing [35, 36], use information learned so far to guide the search process. Thus, they try to gradually improve the convergence towards the optimal solutions. However, because of their restrictions, they may not reach to certain regions of the design space and also may be trapped in local optima. The unguided search methods such as random walk aim to provide an unbiased view of the design space. Each design point is chosen randomly and entirely by chance. They attempt to ensure that the searched space is random and thus representative enough to make generalization about the whole design space. However, they do not use any training techniques to guide them towards the optimal solutions. Figure 4.26 graphically describes two different search strategies for exploring the design space. This figure shows a one-dimensional design space (x) with one objective function ($f(x)$), in which the higher objective value is better (maximization problem).

For solving a multi-objective optimization problem, choosing the best optimization algorithm, which efficiently and effectively explores the design space and finds high



(a) Guided search (b) Unguided search

Figure 4.26: Illustration of search strategies

quality solutions is a big challenge. This because different strategies may have different performance on different problems, and there is no conclusive answer regarding to which algorithm is the best for a specific problem. Performance metrics can be used to compare the outcomes of different multi-objective optimizers in a quantitative manner. The purpose of these metrics is to reveal the strengths and weaknesses of each optimization approach and to identify the most promising technique.

To illustrate the metrics and visualizations presented in this chapter, we revisit the case study form Chapter 3. In this case study, we map a Motion-JPEG (M-JPEG) encoder to a heterogeneous multi-processor system-on-chip platform architecture consisting of a general-purpose micro Processor (mP), a micro Controller (mC), an Application Specific Instruction Processor (ASIP) and two Application Specific Integrated Circuits (ASIC-DCT and ASIC-VLE). For communication, the platform architecture contains two dedicated point-to-point FIFOs (between mP and ASIP) and two shared memories; one Static RAM (SRAM) and one Dynamic RAM (DRAM), each one is accessible through a common bus. Note that the evaluated design instances only use a subset of the platform resources, based on the mapping of application tasks and communication channels onto the platform resources. The MP-SoC architecture model and the M-JPEG encoder application are shown in Chapter 3, Figure 3.12 and Figure 3.13 respectively. Our design space in this study has four parameters: number of processors, processor type, number of memories and memory type. Each architecture platform instance is indicated by a unique combination of these parameters and is considered as a subspace of the design space. Thus, solutions in each subspace have exactly the same architectural components but the way that application tasks and channels are mapped to those components is different. We define three design criteria to evaluate optimal solutions, which are: the processing time, energy consumption and cost of the architecture. For solving the mapping decision problem, we model it as a multi-objective optimization problem and use three different optimization mechanisms to achieve a set of optimal design points (in terms of alternative architectural solutions and mappings) under the aforementioned criteria. In this case study, we are

going to compare the exploration results of these three optimization approaches for our mapping problem.

From the guided search methods, we use NSGA-II evolutionary algorithm for exploring the design space and random walk is used as an unguided method. Furthermore, in order to handle design point duplication in NSGA-II, we add an extra step to the original algorithm to check the duplication of solutions at each generation. After generating a new population and before evaluating solutions in it, we check whether there are some solutions that are repeated in this population. If there are, we first use mutation to modify the duplicated solutions. If the mutated solution also exists in the current population, then we just replace it with a random solution and do not check it again for duplication. Note that our duplication-handler mechanism only prevents the duplication of design points within the same generation and it does not check the repetition of solutions between different generations. Thus, there might be some duplicated points during different generations. In order to examine the effect of our duplication-handler mechanism on the NSGA-II performance, we compare the results of this variant of NSGA-II with the original one. Therefore, in our case study we compare the outcomes of three optimization algorithms: random walk (Random), original NSGA-II and NSGA-II with duplication handler (NSGA-II-dh). For each optimization algorithm, we performed 12 runs with different random generator seeds. In each run, we generated 100 generations, each with a population size of 50.

Table. 4.1 presents averages and standard deviations of the performance metrics for each optimization algorithm with respect to 12 runs. The best value obtained for each metric is highlighted in bold. From this table we can see that on average, NSGA-II-dh has the best performance in terms of all closeness metrics. However, in terms of uniform distribution of solutions in the objective space, its performance is the worst among the three optimization methods (σ_{mst}-metric). The Random strategy has the best spread of Pareto optimal solutions in the objective space (∇-metric). With respect to the Hypervolume metric, the size of the dominated region is almost the same for all approaches. However, the hypervolume value in NSGA-II-dh is the largest. The performance of NSGA-II in the decision space is the worst for the compared algorithms. Applying the duplication handler mechanism to the original NSGA-II significantly improves the ratio of generating unique design points (from 0.127 in NSGA-II to 0.573 in NSGA-II-dh). However, the Random strategy performs strictly better than the other two approaches in terms of unique ratio metric. On average, 95% of evaluated design points in the Random strategy are unique.

In the following, we are going to perform a more detailed analysis on the performance of three optimization methods. For each optimization approach, the result of the run, which produces the best approximation of the Pareto optimal solutions with respect to the closeness metrics, is chosen to undergo further investigation. The performance of these three selected runs (one run for each optimization algorithm) in both objective and decision spaces is measured and visualized by VMODEX with a variety of metrics described in the previous sections.

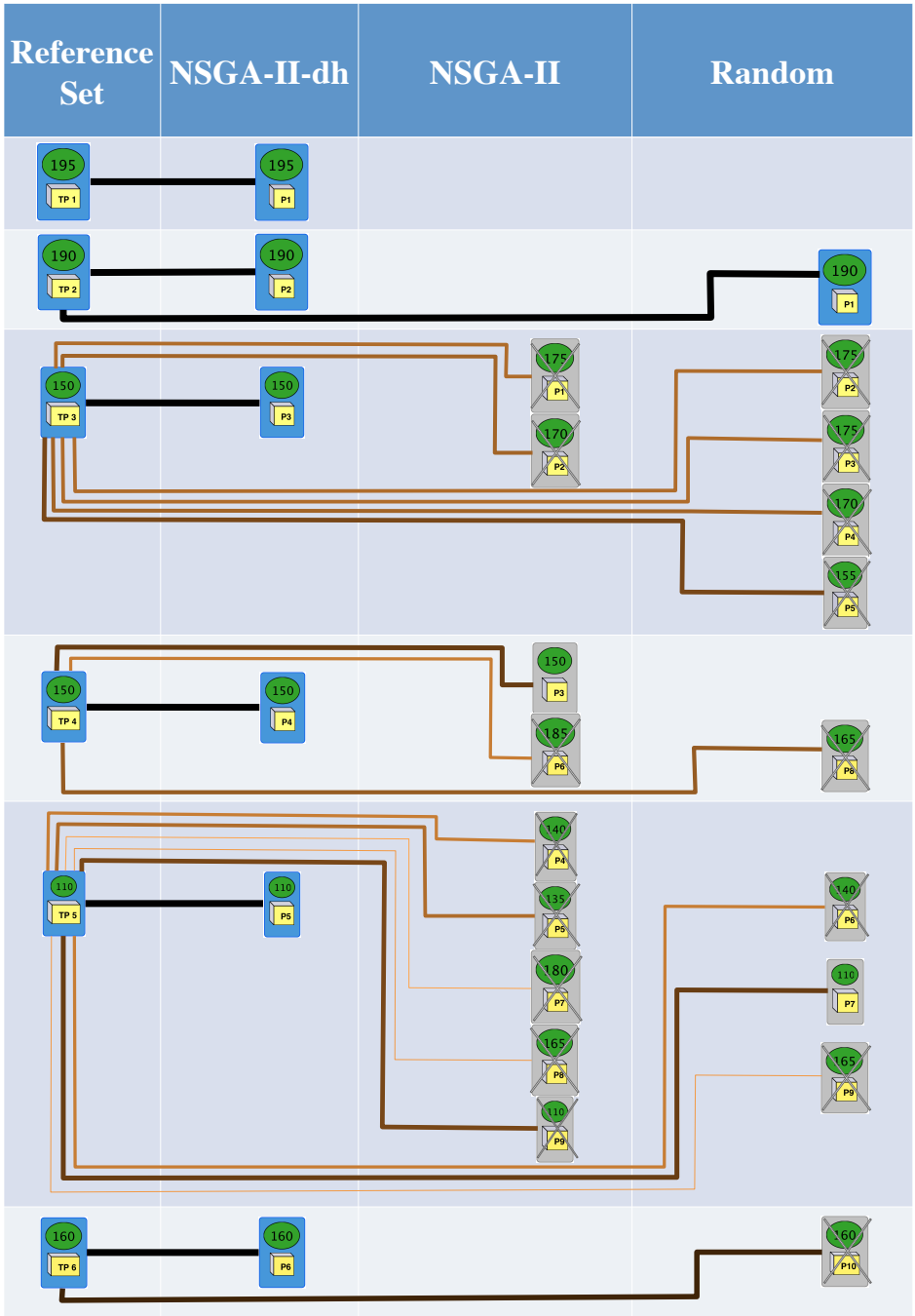
Table 4.1: Performance comparison of the three different optimization algorithms for our case study

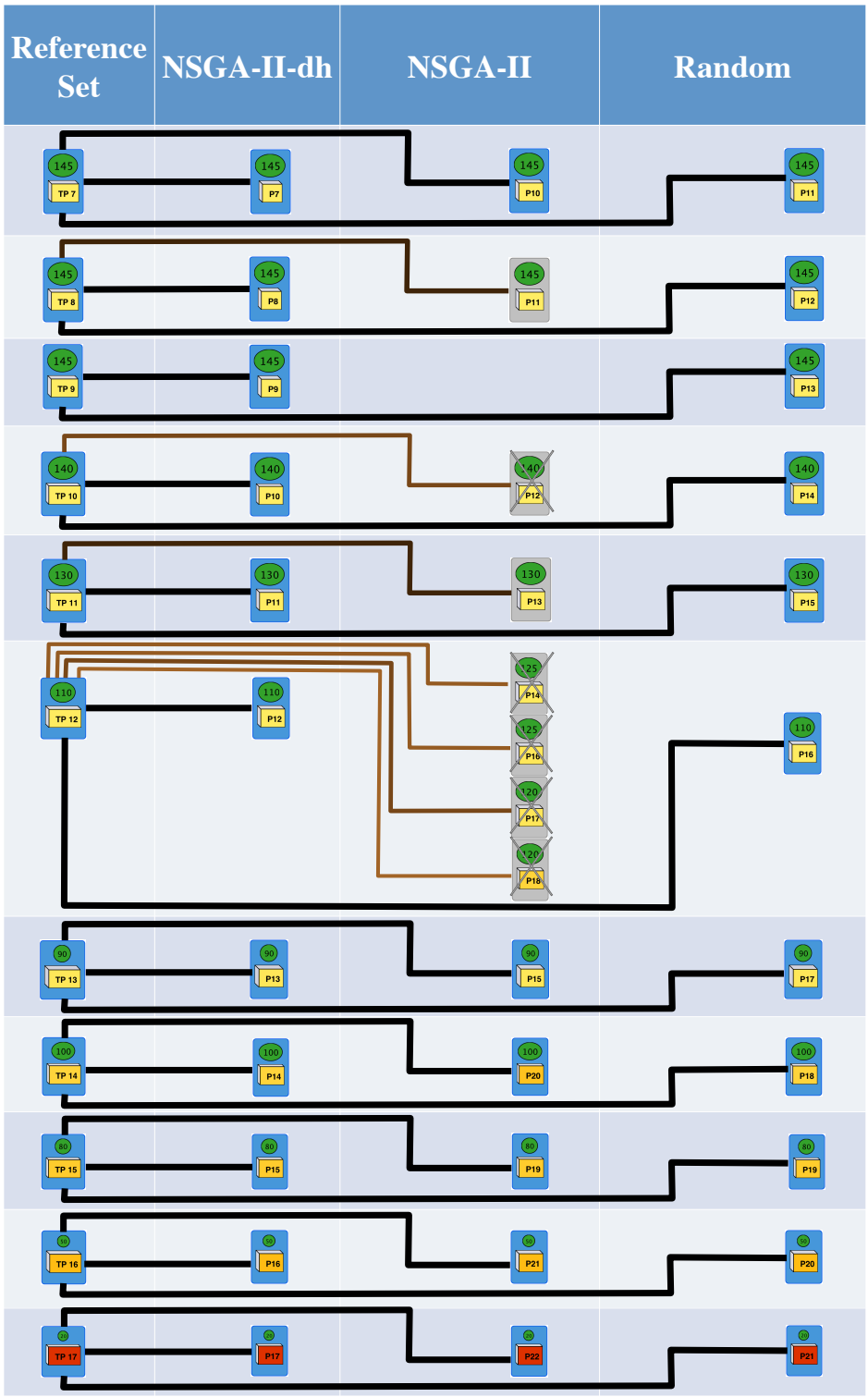
Performance Metrics				NSGA-II	NSGA-II-dh	Random
Objective Space	Closeness	*ER*	*avg.*	0.803	**0.151**	0.526
			std.dev.	0.066	0.064	0.071
		GD	*avg.*	0.0111	**0.0017**	0.0094
			std.dev.	0.0013	0.0012	0.0011
		WSGR (w_1=0.6, w_2=0.4)	*avg.*	0.302	**0.848**	0.653
			std.dev.	0.050	0.071	0.058
	Diversity	*V*	*avg.*	0.066	0.076	**0.084**
			std.dev.	0.0027	0.0043	0.003
		σ_{mst}	*avg.*	**0.1006**	0.1145	0.1012
			std.dev.	0.0081	0.0058	0.0073
		# of Clusters	*avg.*	**3.25**	4.083	3.33
			std.dev.	0.452	0.515	0.492
	Combined	*HV*	*avg.*	0.837	**0.843**	0.839
			std.dev.	0.0078	0.005	0.0057
Decision Space	Coverage	*CSR*	*avg.*	0.783	0.915	**0.918**
			std.dev.	0.028	0.0052	0.003
	Unique Ratio	*UR*	*avg.*	0.127	0.573	**0.952**
			std.dev.	0.0043	0.0131	0.0027

4.4.1 Comparing Performance in Objective Space

Closeness Metrics

Since in our case study the true Pareto front is not known, we use a reference set to evaluate the closeness of discovered Pareto optimal solutions. For estimating the reference Pareto front, we first combined the outcomes of 12 runs of the three optimization algorithms. Then we removed the dominated solutions from the combined set. As a result, we found 17 Pareto optimal solutions that are used as an estimation of the true Pareto optimal solutions. The visualization of closeness metrics for the three optimization approaches is shown in Figure 4.27. In this figure, solutions in each Pareto optimal set are labeled by an index, in the order of increasing processing time. In the reference set, the solutions are sorted by the processing time in ascending order. Thus, from top to down the processing time is increased. As can be seen in this figure, the NSGA-II-dh has found all the Pareto optimal solutions in the reference set.


Reference Set
NSGA-II-dh
NSGA-II
Random
195
TP 1
195
P1
190
TP 2
190
P2
190
P1
150
TP 3
150
P3
175
P1
170
P2
175
P2
175
P3
170
P4
155
P5
150
TP 4
150
P4
150
P3
185
P6
165
P8
110
TP 5
110
P5
140
P4
135
P5
180
P7
165
P8
110
P9
140
P6
110
P7
165
P9
160
TP 6
160
P6
160
P10


Reference Set
NSGA-II-dh
NSGA-II
Random
145 TP 7
145 P7
145 P10
145 P11
145 TP 8
145 P8
145 P11
145 P12
145 TP 9
145 P9
145 P13
140 TP 10
140 P10
140 P12
140 P14
130 TP 11
130 P11
130 P13
130 P15
110 TP 12
110 P12
125 P14
125 P16
120 P17
120 P18
110 P16
90 TP 13
90 P13
90 P15
90 P17
100 TP 14
100 P14
100 P20
100 P18
80 TP 15
80 P15
80 P19
80 P19
60 TP 16
60 P16
60 P21
60 P20
20 TP 17
20 P17
20 P22
20 P21

	WSGR ($w_1 = 0.6, w_2 = 0.4$)	*ER*	*GD*
NSGA-II	*0.6 (6 / 17) + 0.4 (9 / 22) = 0.375*	*16 / 22 = 0.727*	*0.0096*
NSGA-II-dh	*0.6 (17 / 17) + 0.4 (17 / 17) = 1.0*	*0 / 17 = 0.0*	*0.0*
Random	*0.6 (12 / 17) + 0.4 (13 / 21) = 0.671*	*9 / 21 = 0.428*	*0.0083*

Figure 4.27: Comparing performance with closeness metrics

The Random strategy has found 70% (12 out of 17) and NSGA-II only 35% (6 out of 17) of the reference Pareto optimal solutions. Thus, the performance of NSGA-II-dh in finding true Pareto optimal solutions is the best.

From the visualization we can see that all the three optimization algorithms have found the cheapest solutions (less than 100) in the reference set. Furthermore, we can see that, except for $TP\,1$, the Random strategy has found those Pareto optimal solutions in the reference set that have higher processing time (solutions at the bottom of the reference set). With respect to the *ER* metric, all the solutions in NSGA-II-dh are in the reference set and therefore its *ER* is zero (the background color of all of them is blue). On the other hand, more than 70% of the solutions found by NSGA-II are dominated by the solutions in the reference set. This indicates that it was not effective in reaching the true Pareto points. According to the *WSGR* metric, 62% of the solutions in the Random set and 41% of solutions in the NSGA-II set are close enough to the reference set, with respect to the distance threshold. In the closeness visualization, solutions that are considered as sufficiently close solutions are shown either by the blue background (if they are in the true Pareto) or grey background without a cross sign. Furthermore, using the edge visualization technique in showing the distance values we can see that in the Pareto optimal set found by the NSGA-II there are two outliers: $P\,7$ and $P\,8$. Since the thickness and color of their edges are very different from the other edges in this set. The $P\,9$ solution in the Random Pareto set is also an outlier. As a result, the NSGA-II-dh performs better than the other two approaches with respect to all closeness metrics.

Dynamic Closeness

Figure 4.28 shows how fast the Pareto front found by different optimization algorithms converges towards the true Pareto front. In this figure the generational distance is used for measuring the closeness. Since the true Pareto front is not known in our case study, a reference set is used. We can see that NSGA-II-dh finds the closest solutions to the true Pareto front and its convergence speed is higher than the other two methods. In the NSGA-II-dh, during the later generations the distance becomes zero. This indicates that all the found Pareto optimal solutions by NSGA-II-dh are in the true Pareto front. Furthermore, it can be seen that during all the generations, the solutions found by NSGA-II-dh are closer to the reference set than solutions in NSGA-II. This indicates

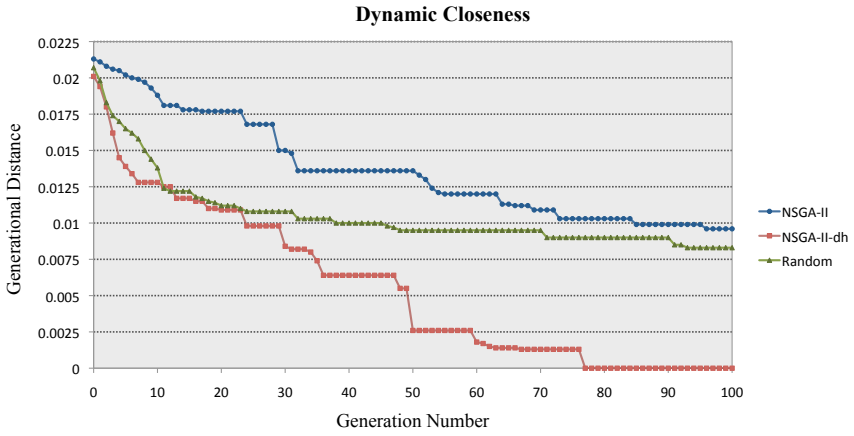


Figure 4.28: Comparing performance with *DC*-metric

that our method for handling the duplication can improve the performance of NSGA-II to find solutions that are closer to the true Pareto front.

∇-Metric

The visualization of the ∇-metric for each optimization strategy is shown in Figure 4.29. As can be seen in this figure, all of them have almost the same extent in all the three criteria. They have nearly perfect extent in cost and their spread in terms of processing time is fairly good. However, only a small portion of the energy consumption is covered by the Pareto optimal solutions found by each optimization approach. In overall, the Random strategy has the best performance in terms of the ∇-metric.

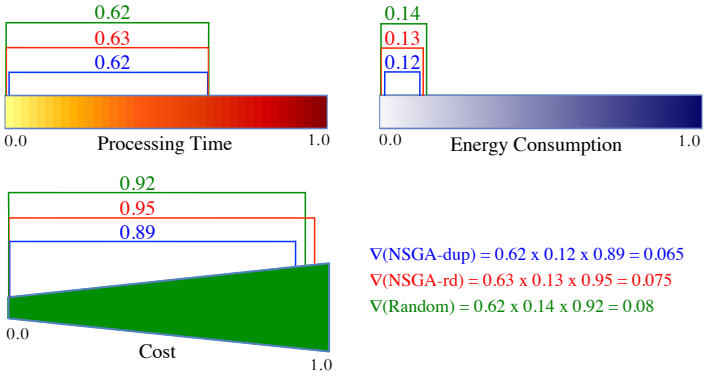
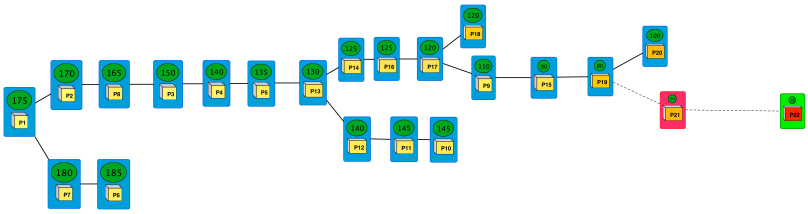


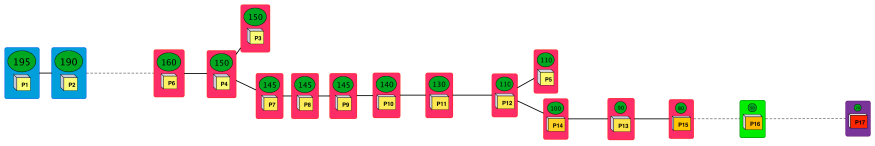
Figure 4.29: Comparing performance with ∇-metric

σ_{mst}-Metric

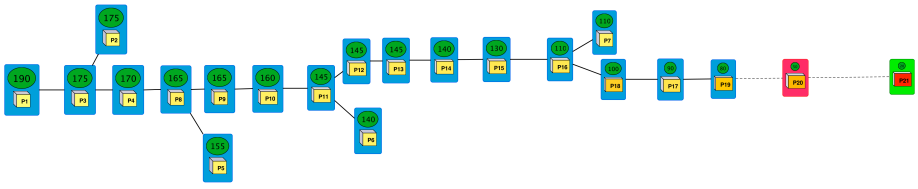
Figure 4.30 shows the visualization of the σ_{mst}-metric for the three optimization algorithms. In this figure, $\bar{w} + \sigma_{mst}$ of the reference set is chosen as a threshold for clustering the solutions. NSGA-II and Random have found respectively 22 and 21 Pareto optimal solutions that are grouped into three clusters. This can be seen in Figure 4.30 as clusters are differentiated by separate colors and there are three different background colors in their σ_{mst} visualizations. Within each cluster, the distances between connecting solutions are less than the threshold and therefore the distribution is nearly uniform. NSGA-II and Random also have approximately the same value for the σ_{mst} metric. However, solutions in NSGA-II-dh are distributed into four clusters and its σ_{mst} value is higher than the two other approaches. Thus, the performance of NSGA-II-dh with respect to the distribution of Pareto optimal solutions in the objective space is worse than the other two optimization methods. From Figure 4.30 we can see that in all three approaches, the two cheapest solutions are grouped in separate clusters, which means that they are far away from the other solutions. Thus, we can conclude that in the part of the objective space containing low cost solutions the distribution is poor.



$\bar{w}$ (NSGA-II) = 0.080 σ_{mst} (NSGA-II) = 0.102



$\bar{w}$ (NSGA-II-dh) = 0.097 σ_{mst} (NSGA-II-dh) = 0.117



$\bar{w}$ (Random) = 0.083 σ_{mst} (Random) = 0.104

Figure 4.30: Comparing performance with σ_{mst} metric

Hypervolume Metric

Figure 4.31 represents the visual form of the hypervolume metric for all the three optimization approaches. As can be seen in this figure, in all of them, a large portion of objective space is dominated by Pareto optimal solutions. However, the size of the dominating area in the processing time-energy consumption surface is bigger than the processing time-cost surface.

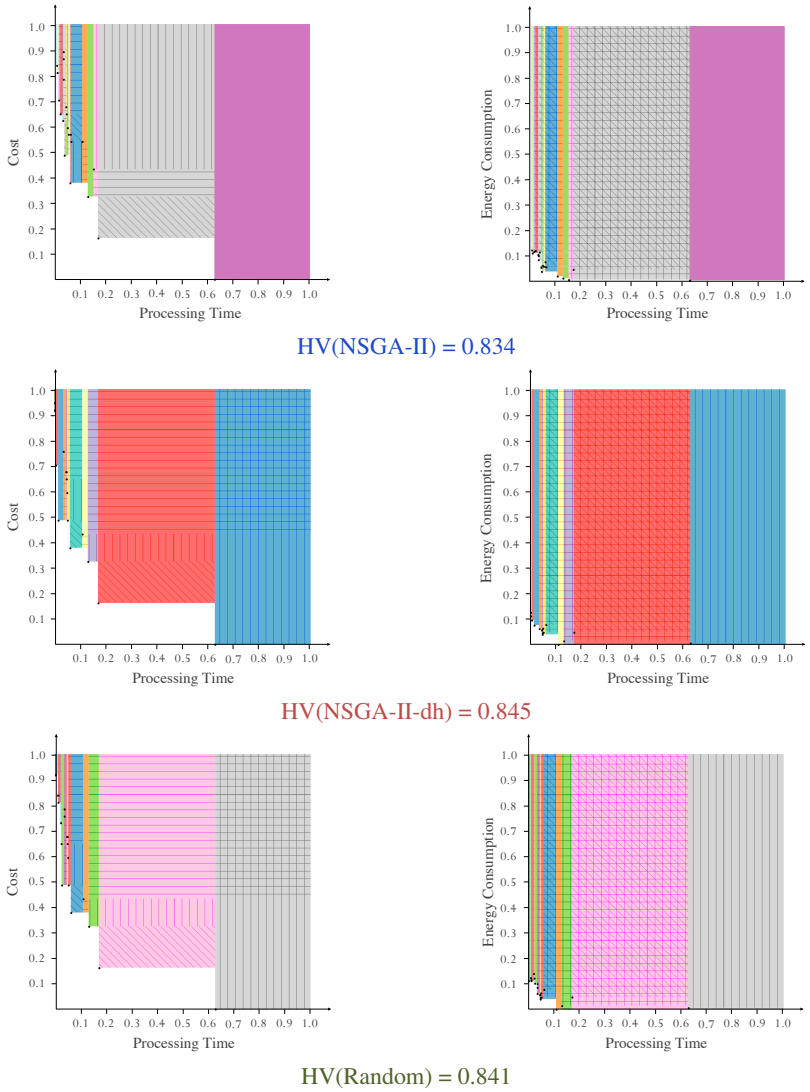


Figure 4.31: Comparing performance with *HV* metric

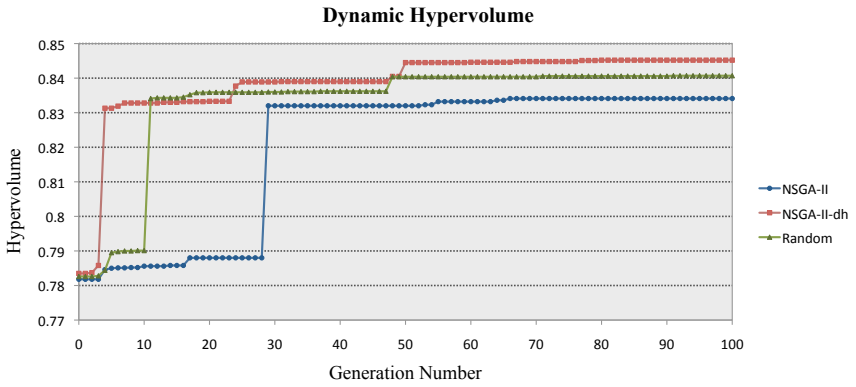


Figure 4.32: Comparing performance with *DH*-metric

Dynamic Hypervolume

Figure 4.32 represents how the hypervolume value is improved during the exploration process. The hypervolume value for generation zero indicates the hypervolume for the initial population, which has been generated randomly. As can be seen in this figure, all the three optimization approaches reach to almost the same hypervolume at the end of the execution. However, their progresses toward this value are different. The hypervolume value that NSGA-II attains after 30 generations can be reached by the NSGA-II-dh after only 4 generations. This indicates that our duplication handler mechanism could increase the optimization speed in a remarkable way. For all of the comparing optimization methods, there is a significant improvement on a certain generation. While before and after that point, the hypervolume only slightly increases during generations. Actually, when the cheapest design point (single-processor solution) is found, there is a major increment in the hypervolume because it can cover a large portion of the objective space.

DFPOS Metric

The visualization of the DFPOS metric for the three comparing optimization methods is shown in Figure 4.33. As can be seen in this figure, the Random strategy has a somewhat uniform process in finding final Pareto optimal solutions, while in the other two approaches there are some gaps in which no Pareto optimal solution is found. These gaps could indicate those generations during which the algorithm is gradually converging towards the optimum solutions by utilizing the information learned from previous generations. However, all approaches have found some new Pareto optimal solutions in the last generations. Thus, by continuing the exploration process for more generations, it is more likely to find other Pareto optimal solutions.

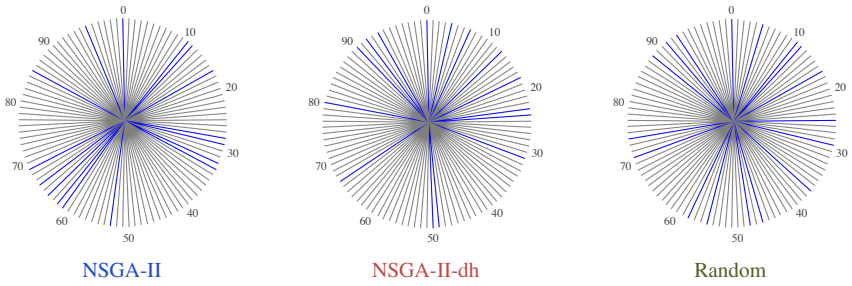


Figure 4.33: Comparing performance with *DFPOS* metric

4.4.2 Comparing Performance in the Decision Space

Decision Space Coverage

In our case study, there are 99 feasible combinations of design parameters (architecture instances), which are capable of executing the M-JPEG application. Note that these feasible combinations are only in the architectural space and does not include the mapping. Each architecture instance is considered as a subspace of the design space. Figure 4.34 represents the dynamic behavior of the three optimization algorithms in finding new architecture instances and accessing the different subspaces of the design space. The value for generation zero indicates the number of searched combinations in the initial population, which is generated randomly. As can be seen in this figure, in our initial population, there exist 58 architecture instances that indicate that the diversity of solutions in the design space for the initial population is really good. More than half of the feasible combinations are searched in the initial population. From Figure 4.34 we can further see that, at the end of the execution, both the NSGA-II-dh and Random mechanisms have visited 91 feasible combinations, which implies

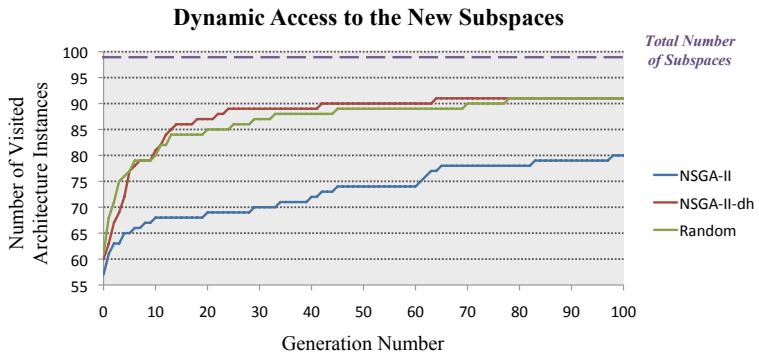


Figure 4.34: Comparing performance with *DANP*-metric

that they have achieved 92% coverage. However, NSGA-II-dh could find these 91 combinations within 64 generations whereas the Random mechanism discovered these number of combinations after 78 generations. NSGA-II has the worst design space coverage among the optimization algorithms. It found 80 architecture instances during its execution. The progress of NSGA-II-dh and Random in finding new instances is relatively good during the early generations (until generation 30) and after that only a few new instances are found. However, the behavior of NSGA-II in finding new instances is somewhat different as it has a much smoother improvement.

Distribution of Design Points in the Decision Space

As we described before, we use a box-percentile plot for showing the distribution information and providing a way to easily and quickly compare the distributions of the results of different optimization algorithms. In our case study, the number of solutions in each subspace is used as the distribution information. We are interested to have almost the same number of evaluated solutions in each subspace (uniform distribution). Figure 4.35 represents the box-percentile plot for the three optimization algorithms. The y-axis indicates the number of solutions in subspaces. For instance, the 50^{th} percentile (median) in the Random strategy is 8, which means that half of the subspaces of the design space contain equal or less than 8 design points. As can be seen in this figure, the distribution of solutions in all optimization algorithms is skewed towards the higher values and there are a few outliers in all of them (long-thin lines in box-percentile plot). However, the positions of these outliers for each algorithm are different. In NSGA-II, there is a single outlier at 138. All the subspaces contain less than 50 solutions (main body of the box-percentile plot) except one of them that have 138 solutions. In NSGA-II-dh, there are two outliers: one at 480 and another one at 260 (there are two thin lines with different widths). In Figure 4.35, these two outliers are filled with different colors. In Random, there are some outliers at around 210 (the width of the line is not too thin, which indicates that there are more than one outlier at this point) and one outlier at 380.

Another noticeable feature in Figure 4.35 is that the shape of the main body of plots is quite different that implies the different distributions for each optimization algorithm. The shape of the plot for NSGA-II is very narrow. This means that most of the subspaces contain a small number of design points. Whereas the plots in NSGA-II-dh and Random are wider. This indicates that there are some subspaces, which involve a larger number of design points.

Unique Ratio

Figure 4.36 shows the duplication diagrams for the optimization algorithms. As expected, the Random strategy has the best performance in finding unique design points (non-duplicated points). In the Random approach, the maximum number of duplication is four and more than 94% of all evaluated solutions are unique. In NSGA-II-dh,

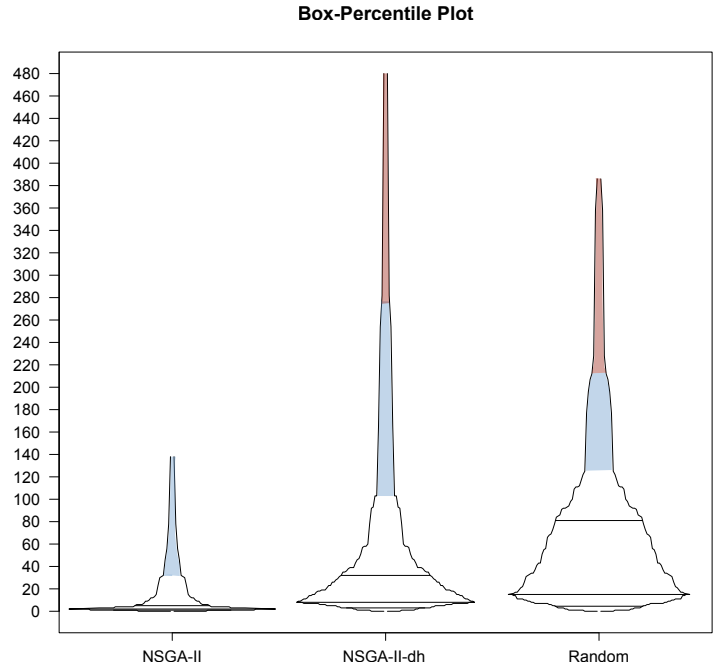


Figure 4.35: Comparing Distribution of Design Points in the Decision Space using box-percentile plot

the maximum number of duplication is 41 and more than half of the searched design points are unique. In NSGA-II, the situation is the worst. Among the total 5100 evaluated design points, it could find only 672 unique solutions (less than 15%). The maximum number of duplication is 500 and there are 15 design points, which are regenerated more than 100 times. From Figure 4.36 we can see that our duplication handler mechanism in NSGA-II-dh performs quite well. It could improve the unique ratio from 0.132 in NSGA-II to 0.586 in NSGA-II-dh. This means that with the same computational effort the number of unique evaluated solutions is increased more than 4 times.

We used the DSE tree to identify the characteristics of design points with the highest duplication. In both NSGA-II and NSGA-II-dh, the three most duplicated solutions in global-Pareto and local-Pareto groups were exactly the same. These solutions were the cheapest solutions (in terms of architecture cost) in our case study. Therefore, these solutions are considered as optimum design points and are selected many times as parents to generate the next population. On the other hands, for all of them there is only one possible mapping of the application onto their underlying architectures.

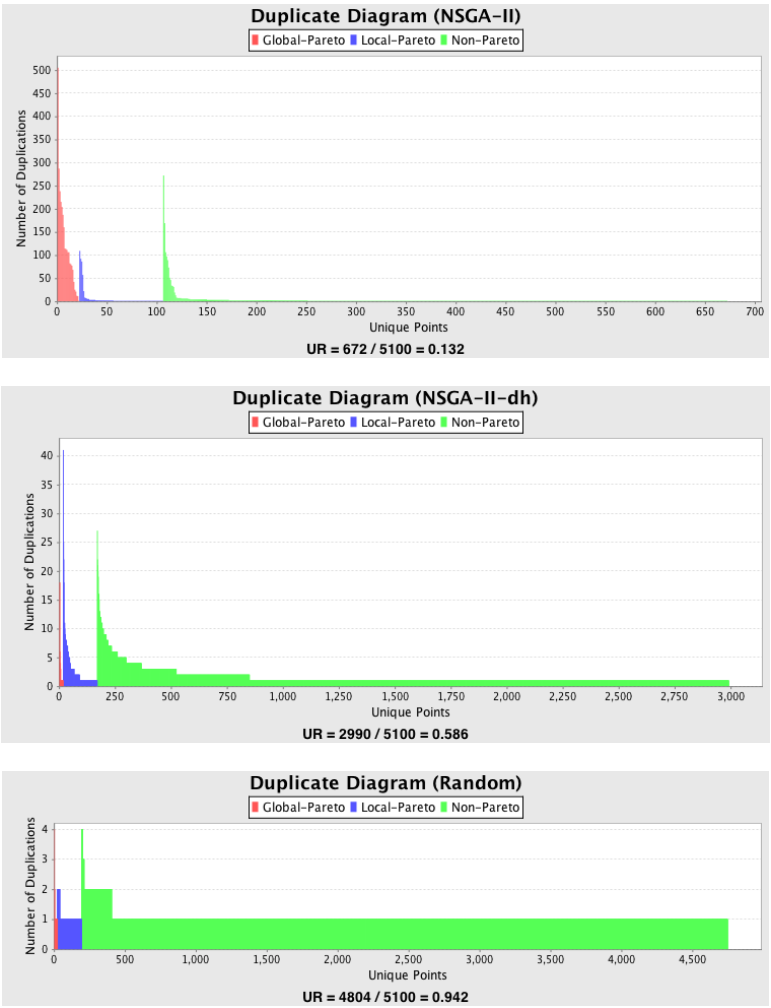


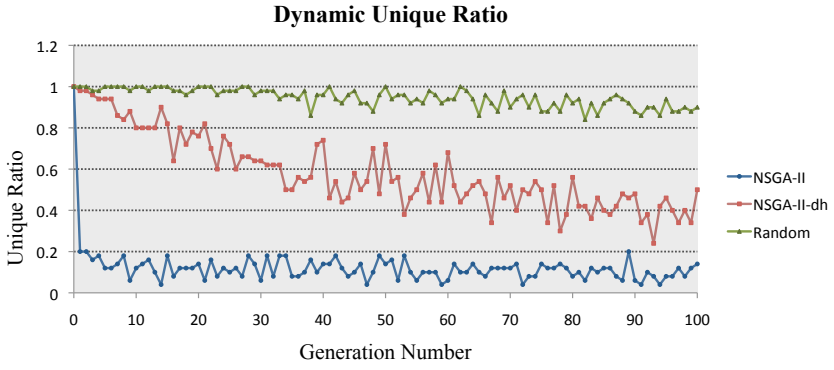
Figure 4.36: Comparing performance with *UR* metric

Thus, they cannot be mutated to create a new design point. As a result, they will be repeated in the next population without any changes.

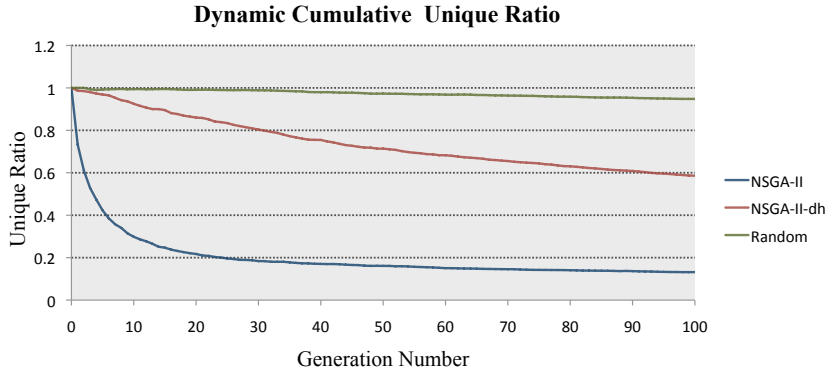
It should be reminded that the duplication handler in NSGA-II-dh only prevents the duplication of design points within the same generation and it does not check the repetition of solutions between different generations. Thus, there might be some duplicated points during different generations. As can be seen in Figure 4.36, even though duplication handler allows repetition between different generations, it still could significantly improve the performance of original NSGA-II with respect to the unique ratio metric. The solution that is repeated 500 times in NSGA-II (highest duplicated

solution in the Global-Pareto category) is only regenerated 18 times in NSGA-II-dh.

Figure 4.37 shows the dynamic performance of the optimization algorithms in finding unique design points. Figure 4.37(a) represents absolute unique ratio and Figure 4.37(b) shows the cumulative unique ratio. As can be seen in Figure 4.37(a), in NSGA-II, except the initial population that is generated randomly, for all the other generations the unique ratio is less than 0.2. In NSGA-II-dh, however, even in later generations, the unique ratio is relatively good. In the Random approach, for all the generations the unique ratio is more than 0.85. From Figure 4.37(b) we can see that in Random strategy the dynamic cumulative unique ration plot is approximately a straight line close to 1.0. In NSGA-II-dh, the slope of plot is relatively gentle that indicates that there is a gradual deterioration in the rate of finding unique design points during the generations. In NSGA-II, the slope of plot is very steep in the first gener-



(a)



(b)

Figure 4.37: Comparing performance with *DUR* metric

ations, while after that it becomes almost a straight line. This means that there is a big degradation in the unique ratio at the first generations. However, during the next generations it is changed only a little bit.

4.4.3 Overall Comparison

The comparison of the performance between the three optimization algorithms for our case study was discussed in the previous two subsections. We compared these algorithms with respect to a verity performance metrics in both objective and design spaces. As a result, the NSGA-II performs better in only finding well-distributed solutions in the objective space (σ_{mst} metric) while for all the other quality aspects, NSGA-II-dh and Random obtain more preferable Pareto optimal solutions. Thus, we can conclude that our duplication handler mechanism could significantly improve the performance of original NSGA-II in terms of most performance aspects. Furthermore, we can see that, in general, the performance of Random strategy is fairly satisfactory. For our specific problem, the random search strategy performs quite well. Thus, as an algorithm developer, one should make the heuristic searching algorithm smart enough to be able to considerably outperform the normal random search.

4.5 Conclusion

In this chapter, we explained various performance metrics (from literature and new ones) to evaluate and compare the outcomes of different optimization algorithms from several view points. In multi-objective optimization problems, several distinct goals need to be achieved and therefore there cannot be a single quality measure that indicates the performance of an optimization algorithm in an absolute sense. Thus, various metrics need to be used to gain a comprehensive analysis of the performance of an optimization approach. In VMODEX, a variety of performance metrics are provided to enable algorithm developers to assess the quality of the discovered Pareto optimal solutions from different angles. Furthermore, several visualization approaches are proposed, which enable researchers to do detailed and more accurate analysis of the different performance aspects. The visualization techniques can reveal some useful and interesting information, which has been hidden in the quantitative representation of a quality aspect and provide insight on the reasons behind the strength/weakness of a Pareto optimal set with respect to a particular metric. Therefore, by using VMODEX, algorithm developers can easily examine different optimization algorithms for exploring the design space and find the best one for their specific problems. Then, the results of the best algorithm are delivered to designers for further analysis. As we explained in Chapter 3, designers can use VMODEX to gain insight into the landscape of the design space and understand how the design space is searched by a heuristic search algorithm.

To illustrate the benefits of our tool, which integrates and visualizes different per-

formance metrics in a single environment, we presented a case study. We studied a multi-objective design problem from the multiprocessor system-on-chip domain: mapping process networks onto heterogeneous multi processor architectures. Here, we have taken three objectives into account, namely the maximum processing time, power consumption, and cost of the architecture. We used three optimization approaches to search the design space and find the Pareto optimal solutions. Then, we utilized VMODEX to compare the performance of these three methods on our specific problem.

It should be mentioned that in the work described in this chapter, the goal is proposing visualization techniques allowing the analyzing and interpreting the performance metric results and enabling users to understand how and why a particular performance aspect is achieved. Although using only scalar values is a straightforward and easy approach for comparing the performance of different algorithms, it cannot provide insight into why one algorithm performs better than the others. So, we define visualization methods to help users to do more rigorous and in-depth analysis on the performance of optimization algorithms. In the visualizations proposed in this chapter, we did not take into account the average behavior of the optimization algorithms in which each algorithm is executed more than once. However, these visualizations address the analysis of results from a one-shot algorithm execution. Nevertheless, the same as comparing different algorithms, several runs of the same algorithm can be evaluated and compared by our tool as well. Therefore, it is possible to understand the average behavior of an algorithm during numerous runs. However, in our future work, we are going to extend our visualization approaches in order to show the average performance of an algorithm (with respect to a specific aspect) in a single view.

5

Multi-Objective Decision Making: Methods and Visualizations

One significant aspect of multi-objective optimization is the fact that finding the Pareto optimal solutions does not completely solve a Multi-Objective Optimization Problem (MOOP). Actually a MOOP involves two stages:

1. Searching the decision space and finding the Pareto optimal solutions
2. Selecting the most suitable solution from the Pareto optimal set

Finding only a set of Pareto optimal solutions is not sufficient. The decision maker still has to choose one solution from this set to be implemented. However, the process of choosing the most preferred solution among the several non-dominated solutions is not trivial. Therefore, many Multi-Objective Decision Making (MODM) methodologies have been suggested to guide decision makers towards the most appropriate solution among the Pareto optimal solutions.

In this chapter, we explain some MODM methods that are provided in VMODEX and can help decision makers to understand the trade-offs between different criteria and select the final solution for the implementation. Furthermore, new visualization approaches are proposed, which provide the visual interpretation and detailed analysis of the results of MODM methods.

5.1 Introduction

As we mentioned in Chapter 2, in multi-objective optimization with conflicting objectives, there is no single optimum solution that simultaneously optimizes all objectives. Instead, a set of Pareto optimal solutions has to be found in which no improvement can be obtained in any of the objectives without causing a simultaneous degradation in at least one other objective. However, after finding the Pareto optimal solutions of the multi-objective optimization problem, we are encountering some difficulties:

Although many efficient solutions has been found, only one or a reduced number of optimal solutions need to be considered for the final decision. Therefore, there is an issue regarding how a decision maker chooses the best solution from a set of Pareto optimal solutions.

In the mathematical sense, all Pareto optimal solutions are regarded as equally desirable. Thus, systematic approaches are needed to express preference information related to the multiple objectives and aid the decision maker to identify the most preferred solution. Without a systematic approach one cannot be sure that the proper decision has been made. Therefore, some decision making methods are provided in VMODEX to assist the decision makers to make better decisions.

Usually, the results of MODM methods are shown in a table or displayed in a 2D graph. Although these kinds of representations are useful and appropriate for gaining a general understanding of the results, they do not provide insight on how these results are achieved. In this chapter, we propose several visualization techniques that enable decision makers a deeper understanding and detailed analysis of the relations between Pareto optimal solutions and allow them to find out why one particular solution is considered as a more preferred solution with respect to a specific MODM method.

For describing the visualization techniques in VMODEX that are developed for visualizing the outcomes of MODM methods, we use a case study of which the results are used as input data for all the visualization approaches explained in this chapter. In our case study, we map a parallel multi-media application to an MPSoC platform architecture consisting of: two Application Specific Integrated Circuits (ASICs), an Application Specific Instruction Processor (ASIP), a general-purpose microprocessor (mP), a microcontroller (mC), two Dynamic RAMs (DRAMs) and one Static RAM (SRAM). The mapping decision problem (i.e. mapping application tasks and communication channels onto the architecture components) is formulated as a multi-objective optimization problem in which three criteria are considered: the processing time, energy consumption and cost of the architecture. To solve this problem, we utilized the SPEA2 [18] evolutionary algorithm to achieve a set of Pareto optimal solutions under the aforementioned criteria. The outcome of running the SPEA2 on our case study has resulted in 14 Pareto optimal points which are shown in Figure 5.1. Note that in this chapter, the considered objectives and their visual representations are the same as in Chapters 3 and 4. Processing time is shown by the color of the node representing a solution. Colors are varied from yellow to red with all color grades in between. Nodes with the lowest processing time are yellow and nodes with the highest processing time are red. The size and color of the third dimension of a solution shows the energy consumption. As the energy consumption increases, the size of the third dimension becomes bigger and its color becomes darker. The architecture cost is shown as separate nodes at the cost level. Cost nodes are represented with circle symbol. The size of the circle becomes bigger as the cost increases.

The rest of this chapter is organized as follows. In Section 5.2 we give some preliminary definitions that are used in multi-objective decision making methods. Section 5.3 describes the four basic problem formulations in MODM, which are: choice, clas-

Figure 5.1: Pareto optimal solutions found through our case study and visualized by VMODEX

sification/sorting, clustering and ranking problems. For each problem formulation, some methodologies (either existing ones or new ones) are explained. Furthermore, different visualization approaches are proposed for visualizing the results of MODM techniques. Finally, Section 5.4 concludes this chapter.

5.2 Preliminary Definitions

In this section, we explain some basic notions that are used in the Multi-Objective Decision Making (MODM) methods. Before introducing the preliminary definitions, the decision problem should be formally stated. In particular, we consider a set of n solutions $S = \{s_1, ..., s_i, ..., s_n\}$. The solutions are evaluated in terms of m objective functions $F = \{f_1, ..., f_j, ..., f_m\}$. The performance of solution s_i on the objective function f_j is denoted by $f_j(s_i)$. Note that in this chapter, index i denotes the counter for the solutions and index j indicates the counter for the objective functions.

5.2.1 Weights

Typically, multiple criteria have different importance. For instance, performance may be more important than cost to the decision-maker. Thus, weighting factors must be used to indicate the relative importance of multiple objectives. The decision maker expresses his preferences among different criteria by assigning a weight to each criterion. The weight of each objective represents its priority relative to the other objectives under consideration. The higher the assigned weight, the more important the objective is. Weights are usually normalized to sum up to 1, so that in a set of weights following condition should be satisfied:

$$\sum_{j=1}^{m} w_j = 1$$

Where m is the number of objectives. There are several ways for deriving weights, however, in this chapter we only explain the two most common methods, which are: ranking and analytical hierarchy processes.

The ranking method is the simplest approach for calculating the weights. First, the objectives are ranked from the most important to the least important (the rank of the most important one is m, second most important is $m - 1$, etc). Then the ranks are normalized into the range [0, 1] by dividing by the sum of the ranks. Let r_j be the

rank given to the j^{th} objective. The weight of the j^{th} objective (w_j) is calculated as:

$$w_j = \frac{r_j}{\sum_{k=1}^{m} r_k} \tag{5.1}$$

In the case that some objectives are equally important, an average rank is assigned to all of them. This average is calculated based on the ranks that they would have when they differ in importance. For example, consider a 4-objective case in which objective 1 is assessed to be the most important, objective 2 and 3 are equally important, and objective 4 is the least important. The ranks will be $r_1 = 4$, $r_2 = 2.5$, $r_3 = 2.5$, $r_4 = 1$.

The Analytic Hierarchy Process (AHP) has been developed by T. Saaty [91, 92] and is based on pairwise comparison. The objectives are compared against one another. The result is a comparison matrix (C) in which the value of the entry c_{jk} indicates the relative importance of the j^{th} objective as compared with the k^{th} objective. For each pair of objectives, the relative importance is determined by assigning a weight between 1 (equal importance) and 9 (extreme importance) to the more important objective, whereas the reciprocal of this value is assigned to the other objective in the pair. Thus, in the comparison matrix C, we have $c_{jk} = c_{kj}^{-1}$ and $c_{jj} = 1$. This means that only $\frac{1}{2}m(m-1)$ comparisons are needed to establish the objective pairwise comparison matrix for m objectives.

The next step is to calculate a set of weights $W = (w_1, w_2, ..., w_m)$ that are most consistent with the relativities determined in the comparison matrix. This means that for each element c_{jk} of matrix C, the corresponding ratio of objective weights (w_j/w_k) should be as close as possible to the value of c_{jk}. There are several techniques for estimating the weighting factors from the comparison matrix. Here we describe the Eigenvector approach, which obtains the weighting vector in a short computational time. The procedure of calculating weights is as follows:

1. Compute the square of comparison matrix.
2. Sum the rows and then normalize the sum values by dividing by the total sum of the rows. The result is the Eigenvector that is considered as the weighting vector.
3. Repeat step 1 and 2 until the difference between two consecutive vectors is smaller than a predefined value.

For better understanding the AHP, we present a numerical example. In this example, we consider a 3-objective problem and we use the AHP approach to calculate the weights of these three criteria. Assume that the following matrix is our comparison matrix (Note that we used four decimal places in our calculations):

$$C = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{3}{1} \\ \frac{2}{1} & \frac{1}{1} & \frac{4}{1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{1} \end{bmatrix} = \begin{bmatrix} 1.0000 & 0.5000 & 3.0000 \\ 2.0000 & 1.0000 & 4.0000 \\ 0.3333 & 0.2500 & 1.0000 \end{bmatrix}$$

$$C^2 = \begin{bmatrix} 3.0000 & 1.7500 & 8.0000 \\ 5.3332 & 3.0000 & 14.0000 \\ 1.1666 & 0.6667 & 3.0000 \end{bmatrix}$$

$$W = \begin{bmatrix} 3.0000 + 1.7500 + 8.0000 \\ 5.3332 + 3.0000 + 14.0000 \\ 1.1666 + 0.6667 + 3.0000 \end{bmatrix} = \underset{sum=39.9165}{\begin{bmatrix} 12.7500 \\ 22.3332 \\ 4.8333 \end{bmatrix}} = \begin{bmatrix} \frac{12.7500}{39.9165} \\ \frac{22.3332}{39.9165} \\ \frac{4.8333}{39.9165} \end{bmatrix} = \underset{sum=1.0000}{\begin{bmatrix} 0.3194 \\ 0.5595 \\ 0.1211 \end{bmatrix}}$$

By repeating the above process for one more time, the following weighting vector is obtained:

$$W = \begin{bmatrix} 0.3196 \\ 0.5584 \\ 0.1220 \end{bmatrix}$$

The difference between this vector and the previous one is not a lot with respect to four decimal places. Therefore, the iterative process can be terminated. As a result, we obtained the following weights for our three objectives: $w_1 = 0.3196$, $w_2 = 0.5584$, $w_3 = 0.1220$.

5.2.2 Utility Functions

Utility functions are widely used in MODM for describing the preferential modeling. A utility function U is a mathematical representation of the decision maker's preferences such that: $U(x) > U(y)$ means that alternative x is preferred over alternative y and $U(x) = U(y)$ indicates that both alternatives are equally preferred (x is indifferent from y).

For each objective function f_j, a marginal utility function u_j needs to be defined. A marginal utility function is a monotone function (linear or nonlinear) defined on the criterion scale and provides a mechanism for transforming the scale of the criterion into a new scale ranging in the interval [0, 1], by taking into account the following two conditions:

$$u_j(f_j^{best}) = 1 \quad and \quad u_j(f_j^{worst}) = 0$$

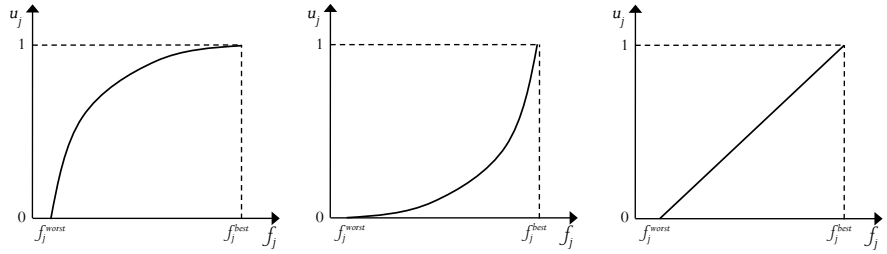


Figure 5.2: Examples of utility functions for typical preference models

Where f_j^{best} denotes the most preferred and f_j^{worst} worst indicates the least preferred value of criterion f_j. These values are calculated according to all solutions in the set S, as follows:

- For objectives in which higher values indicate higher preference

$$f_j^{best} = \max_{i=1}^{n} f_j(s_i) \quad and \quad f_j^{worst} = \min_{i=1}^{n} f_j(s_i)$$

- For objectives in which higher values indicate lower preference

$$f_j^{best} = \min_{i=1}^{n} f_j(s_i) \quad and \quad f_j^{worst} = \max_{i=1}^{n} f_j(s_i)$$

The form of the marginal utility functions depends upon the decision makers preferential system. Figure 5.2 shows some examples of utility functions for three typical preference models. Figure 5.2(a) represents a concave form of the utility function. In this case, a small deviations from the worst value leads to a significant improvement in the preference score. This describes a situation in which the decision maker is satisfied with acceptable solutions and does not necessarily look for solutions with high performance. Figure 5.2(b) shows an inverse situation. In this case, the decision maker is mainly interested in solutions with high performance. The linear marginal utility function shown in Figure 5.2(c) indicates that the preference increases linearly by moving from the worst to the best criterion value.

The global utility provides an overall evaluation of the performance of the solution s_i when all criteria are taken into account. The simplest way for obtaining global utility is the weighted sum of the utility values of all criteria, as follows:

$$U(s_i) = \sum_{j=1}^{m} w_j \, u_j(f_j(s_i)) \tag{5.2}$$

Where $U(s_i)$ is the global utility of solution s_i and w_j is the weight of the j^{th} objective. The global utilities are in the range [0, 1].

5.2.3 Preference Functions

Unlike the utility functions that assign an absolute utility to each solution (globally or on each criteria), the preference functions as introduced by Brans et al. [93, 94], are based on pairwise comparisons between solutions. The preference function PF is used as a measure of intensity of preference of solution x over solution y. This function should be defined separately for each criterion and its value is in the range [0,1]. In this case, the difference of performance between two solutions on a particular criterion ($f_j(x) - f_j(y)$) is considered as a degree of preference, $P_j(x, y)$, as follows:

$$\begin{aligned} P_j(x,y) &= PF_j(d_j(x,y)) \\ where \quad d_j(x,y) &= f_j(x) - f_j(y) \\ for\, which \quad & 0 \leq P_j(x,y) \leq 1 \end{aligned}$$

A smaller deviation indicates the greater indifference and therefore a smaller preference value is allocated to the better solution and even possibly zero if the deviation is negligible. The larger deviation implies greater preference and the value is closer to one. Thus, the degree of preference $P_j(x, y)$ represents the decision makers preference intensity of x over y on the j^{th} criterion as follows:

- $P_j(x, y) = 0$ represents indifference or no preference, $f_j(x) \simeq f_j(y)$
- $P_j(x, y) \simeq 0$ represents weak preference, $f_j(x) > f_j(y)$
- $P_j(x, y) \simeq 1$ represents strong preference, $f_j(x) \gg f_j(y)$
- $P_j(x, y) = 1$ represents strict preference, $f_j(x) \ggg f_j(y)$

For each pair of solutions x and y, the following condition is true:

$$P_j(x, y) > 0 \Rightarrow P_j(y, x) = 0$$

In [94], six types of particular preference functions are defined that can cover most of the practical applications. Figure 5.3 describes the three most commonly used of these functions. The parameters p and q have to be identified by the decision maker. The meaning of these parameters is as follows:

- q is an indifference threshold. When the difference is smaller than q it is considered as negligible by the decision maker. Therefore, the corresponding preference degree is zero.
- p is the strict preference threshold. If the difference is greater than p it is considered to be significant enough to generate a strict preference. Therefore the preference degree is one.

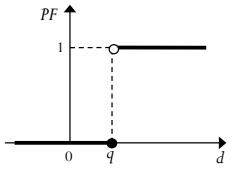
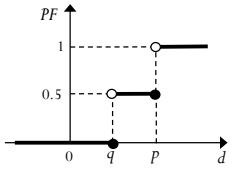
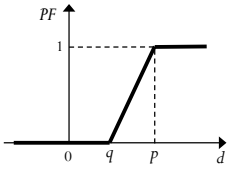
Preference Function	Definition
	*U-shape criterion* $PF(d) = \begin{cases} 0 & d \le q \\ 1 & d > q \end{cases}$
	*Level criterion* $PF(d) = \begin{cases} 0 & d \le q \\ 0.5 & q < d \le p \\ 1 & d > p \end{cases}$
	*V-shape criterion with indifference criterion* $PF(d) = \begin{cases} 0 & d \le q \\ \dfrac{d-q}{p-q} & q < d \le p \\ 1 & d > p \end{cases}$

Figure 5.3: Types of Preference functions

The preference functions shown in Figure 5.3 are applicable for the criteria that need to be maximized. In the case of criteria that need to be minimized, the preference function should be reversed or alternatively given by:

$$P_j(x, y) = PF_j(-d_j(x, y))$$

The global preference index, $\pi(x, y)$, is defined as the weighted sum of the preference functions $P_j(x, y)$ for all the criteria:

$$\pi(x, y) = \sum_{j=1}^{m} w_j \, P_j(x, y) \tag{5.3}$$

$$for\ which \quad 0 \le \pi(x, y) \le 1$$

Where w_j is the weight given to the criterion j. $\pi(x, y)$ indicates the degree in which solution x is preferred over y for all the criteria. It is clear that $\pi(x, y) \simeq 0$ implies a weak global preference of x over y and $\pi(x, y) \simeq 1$ implies a strong global preference of x over y.

5.3 Problem Formulations in Multiple-Objective Decision Making

The aim of Multi-Objective Decision Making (MODM) is to help decision makers to comprehensively understand the trade-offs between different criteria and guide them towards the most preferred decision. The decision making methods allow decision makers to apply their preferences to a decision problem in a logical and mathematical approach. The final recommendation in MODM may take different forms, according to the manner in which a problem is formulated. Four basic problem formulations are as follows:

- The *choice* problem that aims to select a subset of Pareto optimal set, as small as possible, which contains the most satisfying solutions.
- The *classification/sorting* problem that intends to assign each solution to one of the predefined groups.
- The *clustering* problem that aims to assign Pareto optimal solutions to different clusters according to some similarity measure.
- The *ranking* problem that aids to determine the relative goodness of each solution in the Pareto optimal set as compared with the other solutions. The result is usually presented as a ranking of the Pareto optimal points from the best to the worst, with respect to the decision maker preferences.

While the aim of both classification and sorting is the same, they differ depending on whether the groups are preferentially ordered or not. Classification refers to the case where the classes are defined in a nominal way (i.e. there is no order on the classes) whereas in sorting the classes are defined in an ordinal way starting from those containing the most preferred solutions to those containing the least preferred solutions.

It is necessary to emphasize the difference between classification and clustering. They have two distinctive differences: 1) in the classification problems, the aim is to assign objects to the groups that are defined a priori and these groups are so-called ”classes”. Whereas in the clustering, there is no knowledge about the groups in advance and the groups are so-called ”clusters”. Thus, the purpose is to elicit clusters of objects sharing similar characteristics. 2) Classification is based on absolute judgments, while clustering is based on relative judgments. In classification, the assignment of an object to an appropriate class relies on the intrinsic evaluation of that object, on all criteria, with respect to the profiles defining the classes (not on the comparison of that object to the other objects in the set). On the contrary, clustering is based on comparing the objects inside a set with each other, according to some similarity measure, and organizing them into clusters including objects with similar characteristics.

In the following subsections, we explain some decision making methods for each problem formulation.

5.3.1 Choice Problem

In VMODEX, we provide two approaches for selecting a set of best Pareto optimal solutions: filtering and a fuzzy scheme.

Filtering

By applying filtering operations, any solution that does not satisfy the decision maker's preference constraints will be left out for further consideration, and the decision making process proceeds with a set of most preferred design points.

In VMODEX, there are several ways in which the preferences of decision makers can be defined. The decision maker can determine his preference objective values by setting the upper and lower limits on each objective. Then, those solutions with objective values inside the selected ranges are chosen as most preferred solutions and all the other solutions that exceed the (objective values) limits are eliminated from the Pareto optimal set.

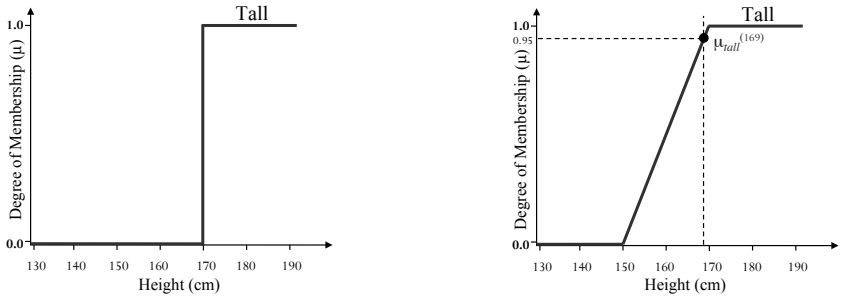
The decision maker is also able to determine his preferences on the design parameters and consider only the solutions with preferred parameters. For instance, in our case study, the designer may prefer the points that have an ASIC in their underlying architecture. Therefore, by applying the filtering option, the points without an ASIC will be eliminated from the Pareto optimal set.

Another possibility for filtering the solutions, which is specific to our domain, is based on the mapping decision (mapping of the application tasks and channels onto the architecture components). The decision maker can indicate his preferences on the mapping (e.g. a specific task should be mapped to a specific processor or some specific tasks should be mapped on the same processor) and then filter the design points to see only those solutions that do not violate the determined mapping constraints. A more detailed description about the mapping filtering is given in Chapter 3, Section 3.2.5. Establishing the preferences on the combination of objective values, design parameters and mapping decisions is provided as well.

Fuzzy Scheme

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth, being values that lie between "absolutely true" and "absolutely false". Boolean logic deals with situations that can be true or false. However, fuzzy logic allows degrees of truth (expressed as a membership function) in the range of zero to one. A degree of zero means absolutely false and a degree of one means absolutely true.

In a simple example, consider the set of "tall" men. The conventional method requires establishing a height threshold (e.g. 170 cm) and any person with the height more than the threshold is declared as a "tall" man and with the height less than the threshold as "not tall". So, a man with 171 cm would consider as a "tall" man (is a member of tall



(a) Boolean logic (b) Fuzzy logic

Figure 5.4: Tall membership function

set) and a man with 169 cm would not consider as a "tall" man (is not a member of tall set). Therefore, in Boolean logic, the degree of membership can be zero or one. Figure 5.4(a) shows the "tall" function in Boolean logic. However, using the fuzzy logic approach, membership in the set of "tall" men is a number between zero and one. An example of a fuzzy membership function for "tall" set is shown in Figure 5.4(b). The degree of membership for men with 171 cm height is one and for men with 169 cm is 0.95. Therefore, if person x has a higher degree of membership in "tall" set than person y this means that x is taller than y.

We can use fuzzy logic to extend the notion of the dominance relation between two solutions. Therefore, instead of saying "x dominates y" or "x does not dominate y", we can say that "x dominates y by degree μ". This fuzzy approach enables us to compare non-dominated solutions with each other and quantify the goodness of each solution within a Pareto optimal set.

Fuzzy Dominance For computing the dominance degree of solution x over solution y, we should consider for each objective, in which degree solution x is better than or equal to solution y. An example of definition of fuzzy sets for "better" and "equal" functions is shown in Figure 5.5. Here we consider a minimization problem, which means that all objectives are to be minimized and therefore smaller objective values are better. In Figure 5.5, two parameters ε and γ should be determined by the decision maker who is familiar with the problem. ε indicates the interval within which an improvement or degradation on the objective is meaningless and γ defines the relevant but not a significant improvement. The value of these parameters may be different for each objective. Although these parameters should be identified carefully, their intended meaning is evident and a decision maker with prior knowledge on the problem can easily specify them.

A straightforward definition for the fuzzy dominance relation can be expressed as

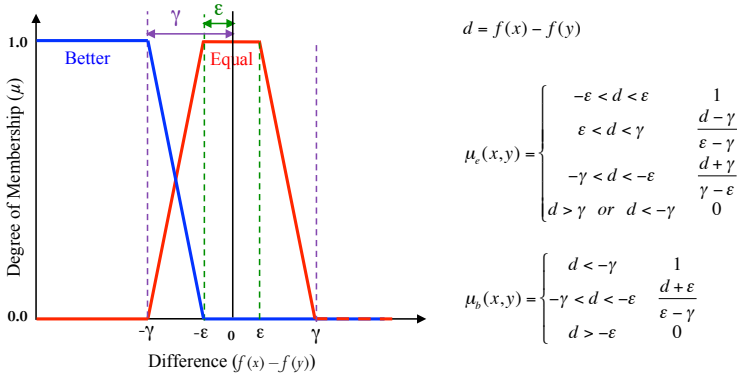


Figure 5.5: A fuzzy definition of "better" and "equal"

Equation 5.4. It is said that solution x dominates solution y by degree $\mu_d(x, y)$ with:

$$\mu_d(x, y) = \frac{1}{m}\mu_b(x, y) + \frac{1}{2m}\mu_e(x, y) \tag{5.4}$$

$$Where \quad \mu_b(x, y) = \sum_{j=1}^{m} \mu_b^j(f_j(x) - f_j(y))$$

$$\mu_e(x, y) = \sum_{j=1}^{m} \mu_e^j(f_j(x) - f_j(y))$$

Where m is the number of objectives and f_j is the j^{th} objective function. μ_b^j indicates the degree of "betterness" and μ_e^j denotes the degree of "equality" on the j^{th} objective. $\mu_d(y, x)$ shows the dominance degree of y over x or, on the other hand, indicating in which degree solution x is dominated by solution y. Note that the fuzzy dominance relation is not symmetric and therefore "dominating by degree μ" and "being dominated by degree μ'" have different fuzzy values.

For clarifying the fuzzy dominance relation described above, we give a numerical example. In this example, we calculate the dominance degrees for two non-dominated solutions $x = (0.1, 0.5, 0.3)$ and $y = (0.6, 0.4, 0.57)$. The fuzzy membership functions for "better" and "equal" are the same as ones shown in Figure 5.5. For all objectives $\varepsilon = 0.1$ and $\gamma = 0.3$. Figure 5.6 shows the values of μ_b and μ_e for all

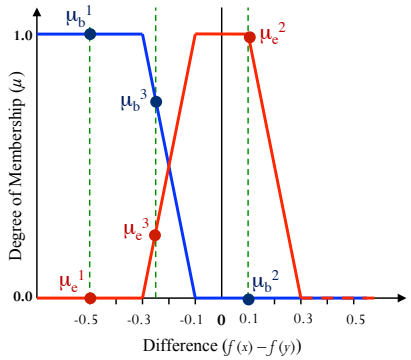


Figure 5.6: Fuzzy sets and membership values for the numerical example

objectives. The dominance degree of x over y is computed as follows:

$$\mu_b(x,y) = \mu_b^1(0.1-0.6) + \mu_b^2(0.5-0.4) + \mu_b^3(0.3-0.57)$$
$$= \mu_b^1(-0.5) + \mu_b^2(0.1) + \mu_b^3(-0.27) = 1+0+0.85 = 1.85$$

$$\mu_e(x,y) = \mu_e^1(-0.5) + \mu_e^2(0.1) + \mu_e^3(-0.27) = 0+1+0.15 = 1.15$$

$$\mu_d(x,y) = \tfrac{1}{3}\mu_b(x,y) + \tfrac{1}{6}\mu_e(x,y) = \tfrac{1}{3}(1.85) + \tfrac{1}{6}(1.15) = 0.81$$

In the same way we can calculate the dominance degree of y over x, $\mu_d(y,x)$, which is 0.19. Therefore x dominates y by degree 0.81 and is dominated by y by degree 0.19. Thus solution x is more preferred than y.

In this thesis, we propose a choice approach based on the fuzzy dominance relations between the solutions in the Pareto optimal set S. The decision maker should determine a dominance threshold (θ). Then, each solution $s_i \in S$ can be defined as a 2-tuple $< D^+, D^- >$ where:

$$D^+(s_i) = \{s_k \in S \mid \mu_d(s_i, s_k) > \theta\}$$
$$D^-(s_i) = \{s_k \in S \mid \mu_d(s_k, s_i) > \theta\}$$

$D^+(s_i)$ is a set of solutions such that s_i dominates them with degree higher than the threshold and $D^-(s_i)$ contains those solution for which s_i is dominated by them with degree higher than the threshold. Considering the 2-tuple representations of solutions, we define two ways for selecting the most satisfying solutions:

1. Select those solutions for which $|D^+(s_i)| > |D^-(s_i)|$. Where $||$ indicates the number of solutions in the set. So, each solution that more dominates rather than is dominated by a degree higher than θ is chosen as a preferred solution.

2. Specify a difference threshold θ', and then select all solutions with $|D^+(s_i)| - |D^-(s_i)| > \theta'$. The first method is a specific instance of this approach for which θ' is equal to zero.

For better analysis of the fuzzy dominance relations between non-dominated solutions in a Pareto optimal set, we visualize their relations in a directed graph called Fuzzy Dominance Graph (*FDG*). This graph shows more detailed information about the dominance degrees and can be used to compare the solutions. Figure 5.7 represents the fuzzy dominance graph for our case study. Each solution is shown as a node in the *FDG*. The dominance threshold θ is used for connecting the nodes. For each two solutions s_i and s_k, if the dominance degree of s_i over s_k is greater than the threshold ($\mu_d(s_i, s_k) > \theta$) then a directed edge is established from s_i to s_k. Thus, for each node (solution) in the *FDG*, the outgoing edges indicate D^+ and the incoming edges denote D^-. A node with more outgoing edges means that it dominates more solutions with the degree higher than θ and a node with more incoming edges means it is dominated by more solutions with the degree higher than θ. The size of each solution s_i indicates the value of $|D^+(s_i)| - |D^-(s_i)|$. The bigger the node means the greater difference between the number of solutions in D^+ and D^- and therefore the solution is more preferred. In Figure 5.7, the threshold is 0.6 and, as can be seen in this figure, solution $P5$ is the biggest node in the graph. It has 10 outgoing edges ($|D^+(P5)| = 10$) which shows that it dominates 10 (out of 14) solutions with a degree higher than θ while there is no incoming edge ($|D^-(P5)| = 0$). There is no solution that dominates $P5$ with a degree higher than θ. The solution $P3$ is the smallest node in the graph. It has no outgoing edge, whereas it is dominated by more than half of the solutions (8 out of 14) with a degree higher than θ ($|D^+(P3)| - |D^-(P3)| = 0 - 8 = -8$). In Figure 5.7, solutions with $|D^+| - |D^-| > 0$ are highlighted by a blue background color, which are $P5$, $P2$, $P11$, $P7$, and $P13$ in the order of decreasing the value of $|D^+| - |D^-|$.

If the decision maker wants to know more about the dominance degrees of one specific solution with respect to the other solutions, it is possible to select that particular solution and see more detailed information. In Figure 5.8, the dominance relations between $P5$ and all the other solutions are shown. For each solution, the length of the blue bar represents the dominating degree of $P5$ over that solution and the length of the red bar shows the degree in which $P5$ is dominated by that specific solution. As can be seen in this figure, except $P2$, the dominance degree of $P5$ over each solution (blue bar) is higher than the degree in which $P5$ is dominated by that solution (red bar).

5.3.2 Classification/Sorting Problem

The assignment of objects into predefined groups is referred to as the classification/sorting problem, depending on whether the groups are specified in nominal or ordinal terms. In this type of problem, groups are usually defined independently of the objects under consideration and assigning an object to the appropriate group requires the absolute comparison of that object to some reference profiles that distinguish the

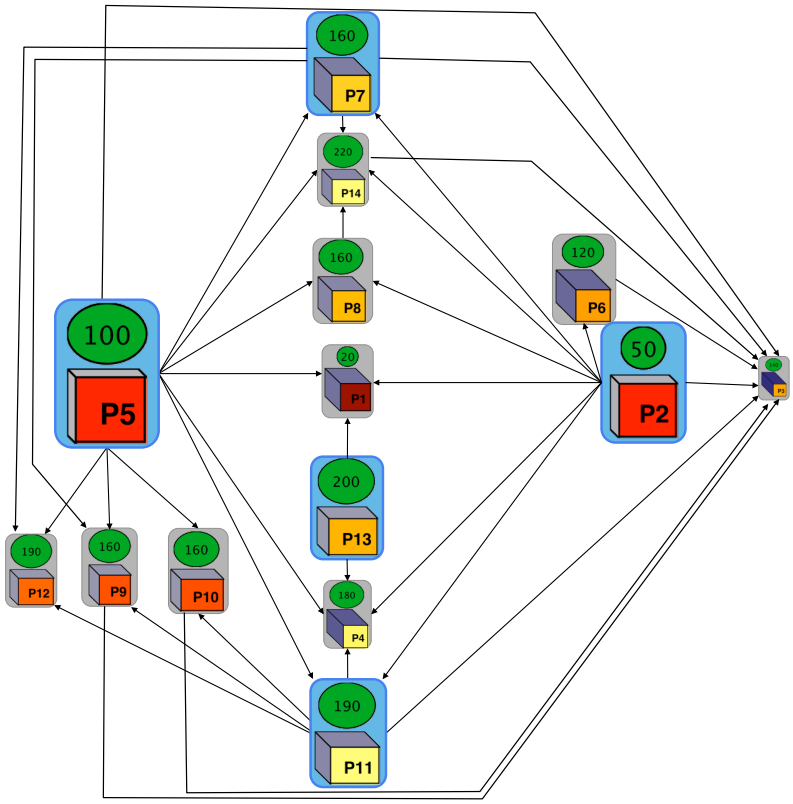


Figure 5.7: Visualization of fuzzy dominance relations

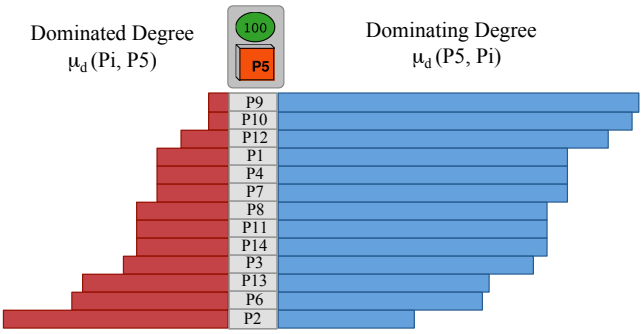


Figure 5.8: Visualization of fuzzy dominance relations between $P5$ and all the other solutions

groups.

In VMODEX, for sorting a set of Pareto optimal solutions in the predefined classes, we have utilized the UTADIS method (UTilits Additives DIScriminantes) [95]. The UTADIS method is one of the simplest and most popular methodologies for addressing the sorting problems. UTADIS employs utility functions to classify a set of alternatives (in our case Pareto optimal solutions) into predefined ordinal groups.

Before proceeding with describing the UTADIS method, the problem should be formally stated. In particular, the problem under consideration is to assign a set of n Pareto optimal solutions $s_i(i = 1, 2, ..., n)$ into q predefined ordinal classes $C_k(k = 1, 2, ..., q)$. Where C_1 indicates the group including the most preferred solutions and C_q denotes the group of the least preferred solutions ($C_1 > C_2 > ... > C_q$).

The UTADIS method uses the global utilities for assigning solutions to the predefined classes. The global utility of solution s_i is computed using Equation 5.2, in which the weighted sum of the utility values of all criteria is considered as a global utility. The sorting process is performed through comparing the global utilities of solutions with some utility thresholds that specify the lower bound of each class, as follows:

$$
\begin{aligned}
& U(s_i) \geq u_1 \Rightarrow s_i \in C_1 \\
& u_2 \leq U(s_i) < u_1 \Rightarrow s_i \in C_2 \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
& u_k \leq U(s_i) < u_{k-1} \Rightarrow s_i \in C_k \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
& U(s_i) < u_{q-1} \Rightarrow s_i \in C_q
\end{aligned}
$$

Where $u_1 > u_2 > ... > u_{q-1}$ are the utility thresholds that are considered as cut-off points to discriminate the classes. The utility threshold u_k separates two consecutive classes C_k and C_{k+1}. The parameters of the UTADIS method (utility functions and thresholds) are estimated through a linear programming formulation, so that the minimum classification error is achieved. For further details on the model development process, one is referred to the work of [96].

5.3.3 Clustering Problem

The goal of clustering is to organize a collection of objects into several groups (clusters), such that objects within the same cluster are similar (in some way), while objects belonging to different clusters are dissimilar. Cluster analysis enables us to look at the properties of clusters instead of individual objects and provides simplification of data with minimal loss of information. The clustering methods use a similarity/distance measure as fundamental to evaluate the closeness/separation of the objects under consideration. In general, a similarity measure maps the similarity between the parameterized descriptions of two objects into a single numeric value. The definition of the similarity measure is a key issue for accurate clustering and is closely related to

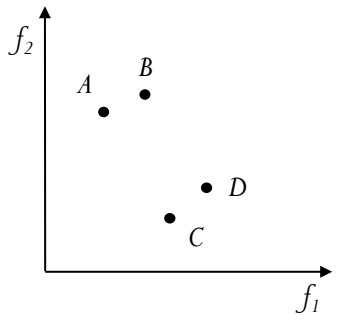


Figure 5.9: A trivial example for illustrating the importance of preference similarity measure

the problem context. Note that the term similarity is often conceived in terms of dissimilarity or distance as well. Dissimilarity corresponds to the notion of distance. Thus, smaller distance means smaller dissimilarity, and thus larger similarity. A variety of similarity/distance measures have been proposed and widely applied, such as Euclidean distance, cosine similarity, Jaccard coefficient, etc.

In the conventional clustering, the similarity measure is based on the geometric distances or related metrics. However, such metrics are not well suited for the clustering problems in the context of MODM. They do not incorporate the decision maker's preferences in the cluster analysis. In the MODM field, a clustering problem is considered as a preference similarity oriented problem in which clusters should be conceived in terms of preference closeness. Thus, the similarity measure should be defined based on the preference relations between the objects such that all objects inside the same cluster are similar in the sense that they have the same preference structure. The preference similarity measures enable us to partition a set of objects into clusters that are meaningful from the MODM perspective.

To clarify the importance of a preference similarity measure, in the MODM context, consider the example shown in Figure 5.9. In this example, both objectives (f_1 and f_2) are decreasing preference criteria (lower value indicates higher preference) and they have similar importance (same weight). Suppose that we intend to group the four solutions A,B,C and D into two clusters. By applying the Euclidian distance as a similarity measure, the two solutions A and B would be in the same cluster and the two solutions C and D would be in the other cluster. However, note that, A dominates B (is better in both objectives) and C dominates D. Therefore, the decision maker prefers A over B and C over D. So, in terms of preference, A and B are not considered the same and thus they should not be grouped in the same cluster. This is true for solutions C and D as well. So, the oriented preference clustering should separat A from B and C from D. By involving the decision maker's preferences in the clustering, the two solutions A and C would be in the same cluster, since they are not dominated by any other solutions, and the two solutions B and D would be in the other cluster, since they are dominated by the other solutions.

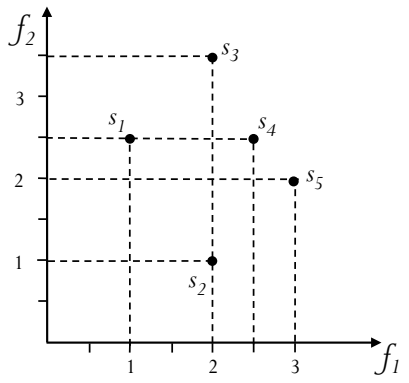


Figure 5.10: A numerical example for describing our proposed preference similarity measure

The importance of integrating decision maker's preferences in the multi-objective cluster analysis was firstly pointed out by [97]. They introduced a distance based on the preference structure defined by the decision maker. Up to now, however, relatively little research has been done on addressing the multi-objective preference clustering [98–100]. In this thesis, we propose a new preference similarity measure, which is based on the preference relations between solutions. In our approach, the preference functions (see Section 5.2.3) are utilized to identify the preference structure. In the following section, we will give the detailed description of our proposed preference similarity measure. In the rest of this chapter, we simply refer to preference similarity measure as the similarity measure.

Proposed Preference Similarity Measure

For better understanding the procedure of calculating the similarity between two solutions, we explain our approach via a simple numerical example. This example is shown in Figure 5.10. Suppose that both objectives are decreasing preference and the objective f_1 is more important. Their relative importance is such that $w_1 = 0.7$ and $w_2 = 0.3$. For both objectives, the "Level Criterion" preference function is used for constructing the preference structure. This preference function is displayed in Figure 5.3.

As we described in Section 5.2.3, for each two solutions s_i and s_k, the global preference degree of s_i over s_k, $\pi(s_i, s_k)$, is defined as the weighted sum of the preference functions $P_j(s_i, s_k)$ for all the criteria:

$$\pi(s_i, s_k) = \sum_{j=1}^{m} w_j \, P_j(s_i, s_k)$$

$$for\ which \quad 0 \leq \pi(s_i, s_k) \leq 1$$

Table 5.1: π values (global preference) of pairwise comparisons between the solutions in our numerical example

π	s_1	s_2	s_3	s_4	s_5
s_1	0	0.35	0.5	0.7	0.7
s_2	0.3	0	0.3	0.3	0.5
s_3	0	0	0	0	0.35
s_4	0	0	0.15	0	0
s_5	0	0	0.3	0	0

Table 5.2: Π values (net preference) of pairwise comparisons between the solutions in our numerical example

Π	s_1	s_2	s_3	s_4	s_5
s_1	0	0.05	0.5	0.7	0.7
s_2	-0.05	0	0.3	0.3	0.5
s_3	-0.5	-0.3	0	-0.15	0.05
s_4	-0.7	-0.3	0.15	0	0
s_5	-0.7	-0.5	-0.05	0	0

Definition 1 (Net Preference): We define $\Pi(s_i, s_k)$ which represents the net preference of s_i over s_k and is computed using Equation 5.5. The net preference $\Pi(s_i, s_k)$ is the balance between the degree in which s_i is preferred to s_k and the preference degree of s_k over s_i.

$$\Pi(s_i, s_k) = \pi(s_i, s_k) - \pi(s_k, s_i) \tag{5.5}$$

$$for\ which \quad -1 \leq \Pi(s_i, s_k) \leq 1$$

From the definition it is clear that $\Pi(s_i, s_k) = -\Pi(s_k, s_i)$

The results of pairwise comparisons between the solutions in our numerical example (shown in Figure 5.10) are pointed out in Tables 5.1 and 5.2. Table 5.1 shows the π values and Table 5.2 represents the Π values.

Definition 2 (Preference Relations): Based on the value of $\Pi(s_i, s_k)$, we define three different relations between s_i and s_k: positive Preference (P^+), negative Preference (P^-) and Indifference (I):

$$Preference\ Structure = \begin{cases} if \quad \Pi(s_i, s_k) > 0 \ \Rightarrow \ s_i \, P^+ s_k \\ if \quad \Pi(s_i, s_k) < 0 \ \Rightarrow \ s_i \, P^- s_k \\ if \quad \Pi(s_i, s_k) = 0 \ \Rightarrow \ s_i \, I s_k \end{cases}$$

Table 5.3: 3-tuple representations of the solutions in our numerical example

s_1	s_2	s_3	s_4	s_5
$P^+(s_1)=\{s_2,s_3,s_4,s_5\}$	$P^+(s_2)=\{s_3,s_4,s_5\}$	$P^+(s_3)=\{s_5\}$	$P^+(s_4)=\{s_3\}$	$P^+(s_5)=\{\}$
$P^-(s_1)=\{\}$	$P^-(s_2)=\{s_1\}$	$P^-(s_3)=\{s_1,s_2,s_4\}$	$P^-(s_4)=\{s_1,s_2\}$	$P^-(s_5)=\{s_1,s_2,s_3\}$
$I(s_1)=\{s_1\}$	$I(s_2)=\{s_2\}$	$I(s_3)=\{s_3\}$	$I(s_4)=\{s_4,s_5\}$	$I(s_4)=\{s_4,s_5\}$

Definition 3 (3-tuple representation): Once the preference relations are elicited, any solution can be characterized as a 3-tuple $< P^+, P^-, I >$, where:

$$\lambda^1(s_i) = P^+(s_i) = \left\{s_k \in S \mid s_i\, P^+ s_k\right\}$$
$$\lambda^2(s_i) = P^-(s_i) = \left\{s_k \in S \mid s_i\, P^- s_k\right\}$$
$$\lambda^3(s_i) = I(s_i) = \{s_k \in S \mid s_i\, I s_k\}$$

The 3-tuple representations of the solutions in our example are shown in Table 5.3.

Definition 4 (Intersection Set): We define the set $\Lambda^g(s_i, s_k)$ such that $\Lambda^g(s_i, s_k) = \lambda^g(s_i) \cap \lambda^g(s_k)$. So, $\Lambda^g(s_i, s_k)$ contains the common solutions in the g^{th} elemnt of the 3-tuple of s_i and s_k. For instance, in our example, the intersection sets for two solutions s_2 and s_4 are as follows:

$$\Lambda^1(s_2, s_4) = \lambda^1(s_2) \cap \lambda^1(s_4) = P^+(s_2) \cap P^+(s_4) = \{s_3\}$$
$$\Lambda^2(s_2, s_4) = \lambda^2(s_2) \cap \lambda^2(s_4) = P^-(s_2) \cap P^-(s_4) = \{s_1\}$$
$$\Lambda^3(s_2, s_4) = \lambda^3(s_2) \cap \lambda^3(s_4) = I(s_2) \cap I(s_4) = \{\}$$

Definition 5 (Similarity Measure): The basic idea for defining the similarity of two solutions is that two solutions are as similar as their 3-tuple and net preferences are alike. This means that, all solutions inside the same cluster are similar in the sense that they are preferred (P^+), not preferred (P^-) and indifferent (I) to more or less the same solutions and with the same degree (Π). The measure of similarity between two solutions s_i and s_k is given by:

$$Sim(s_i, s_k) = \frac{\displaystyle\sum_{g=1}^{3} \sum_{\substack{t=1,\\ \forall s_t \in \Lambda^g(s_i,s_k)}}^{|\Lambda^g(s_i,s_k)|} (1 - abs(\Pi(s_i, s_t) - \Pi(s_k, s_t)))}{n} \tag{5.6}$$

Where n is the number of solutions in the set $S = \{s_1, s_2, ..., s_n\}$, abs denotes the absolute value and $||$ indicates the number of elements in a set. By expanding the outer sum in Equation 5.6, the similarity measure can be expressed as:

$$Sim(s_i, s_k) = \frac{1}{n}[\sum_{\substack{t=1, \\ \forall s_t \in P^+(s_i) \cap P^+(s_k)}}^{|P^+(s_i) \cap P^+(s_k)|} (1 - abs(\Pi(s_i, s_t) - \Pi(s_k, s_t))) + \sum_{\substack{t=1, \\ \forall s_t \in P^-(s_i) \cap P^-(s_k)}}^{|P^-(s_i) \cap P^-(s_k)|} (1 - abs(\Pi(s_i, s_t) - \Pi(s_k, s_t))) + |I(s_i) \cap I(s_k)|]$$

As we explained before, the value of $\Pi(s_i, s_t)$ when s_i is indifferent to s_t (s_i I s_t) is zero. Thus, in the third sum, the values of both $\Pi(s_i, s_t)$ and $\Pi(s_k, s_t)$ is zero. So, the third sum simply counts the number of common solutions in the sets $I(s_i)$ and $I(s_k)$.

Our proposed similarity measure has the following properties:

$$0 \leq Sim(s_i, s_k) \leq 1$$
$$Sim(s_i, s_i) = 1$$
$$Sim(s_i, s_k) = Sim(s_k, s_i)$$

$Sim(s_i, s_k)$ ranges between 0 and 1, where 1 means that two solutions are the same and 0 means they are completely different. The corresponding distance measure is $Dis(s_i, s_k) = 1 - Sim(s_i, s_k)$ and we will use $Dis(s_i, s_k)$ instead in the clustering procedure.

To clarify the process of measuring the similarity between two solutions, in the following we calculate the similarity between the two solutions s_2 and s_4 of our example:

$$\begin{aligned} Sim(s_2, s_4) &= \frac{\overbrace{(1 - abs(\Pi(s_2, s_3) - \Pi(s_4, s_3)))}^{s_3 \in \Lambda^1(s_2, s_4)} + \overbrace{(1 - abs(\Pi(s_2, s_1) - \Pi(s_4, s_1)))}^{s_1 \in \Lambda^2(s_2, s_4)}}{5} \\ &+ \frac{\overbrace{0}^{\Lambda^3(s_2, s_4) = \{\}}}{5} \\ &= \frac{(1 - abs(0.3 - 0.15)) + (1 - abs(-0.05 - (-0.7)))}{5} \\ &= \frac{0.85 + 0.35}{5} = 0.24 \end{aligned}$$

Table 5.4: The similarity measures for each pair of solutions in our numerical example

Sim	s_1	s_2	s_3	s_4	s_5
s_1	1	0.44	0.16	0.13	0
s_2	0.44	1	0.22	0.24	0.07
s_3	0.16	0.22	1	0.36	0.32
s_4	0.13	0.24	0.36	1	0.76
s_5	0	0.07	0.32	0.76	1

Table 5.4 shows the measure of similarity between each pair of solutions in our example.

Definition 6 (Cluster Center): Some clustering approaches, such as leader-follower [101], k-means [102] and Partitioning Around Medoids (PAM) [103] algorithms, utilize the concept of cluster center for assigning the objects to different clusters. In these approaches, the similarity of a solution to a cluster is measured based on its similarity to the cluster center. Taking into account the characteristics of all objects inside a cluster identifies the center of that cluster. In the following, we explain the way that we define a cluster center. Suppose that C_r is the r^{th} cluster and α_r is its corresponding center. The 3-tuple representation of the cluster center α_r is given by the union of the 3-tuple of solutions belonging to the cluster C_r, as follows:

$$\lambda^1(\alpha_r) = P^+(\alpha_r) = \bigcup_{s_i \in C_r} P^+(s_i)$$

$$\lambda^2(\alpha_r) = P^-(\alpha_r) = \bigcup_{s_i \in C_r} P^-(s_i)$$

$$\lambda^3(\alpha_r) = I(\alpha_r) = \bigcup_{s_i \in C_r} I(s_i)$$

Suppose that in our example, the two solutions s_2 and s_3 are grouped in the same cluster (C_1). The 3-tuple of their cluster center (α_1) is:

$$\lambda^1(\alpha_1) = P^+(\alpha_1) = \{s_3, s_4, s_5\}$$

$$\lambda^2(\alpha_1) = P^-(\alpha_1) = \{s_1, s_2, s_4\}$$

$$\lambda^3(\alpha_1) = I(\alpha_1) = \{s_2, s_3\}$$

Note that in the 3-tuple of a solution, there is no common solution between different elements of its 3-tuple (e.g. there is no solution that belongs to both P^+ and P^-). On the other hand, for each solution, the intersection between the sets P^+, P^- and I is empty. However, there might be some common solutions between different elements of the 3-tuple of a cluster center, such as s_4 in the above example. s_4 is a member of

both $P^+(\alpha_1)$ and $P^-(\alpha_1)$. Thus, for each solution s_i inside a cluster, two different net preferences of cluster center α_r over solution s_i is defined: the positive net preference $\Pi^+(\alpha_r, s_i)$ for those solutions belong to $P^+(\alpha_r)$, and the negative net preference $\Pi^-(\alpha_r, s_i)$, for those solutions belong to $P^-(\alpha_r)$.

The positive net preference of cluster center α_r over solution s_i is the average of the net preferences of all solutions (s_k) in the cluster C_r such that $s_k P^+ s_i$, formally:

$$\forall s_i \in \lambda^1(\alpha_r) \quad \Pi^1(\alpha_r, s_i) = \Pi^+(\alpha_r, s_i) = \frac{\sum_{s_k \in C_r \mid s_k P^+ s_i} \Pi(s_k, s_i)}{|s_k \in C_r \mid s_k P^+ s_i|}$$

$$for\ which \quad 0 < \Pi^+(\alpha_r, s_i) \leq 1$$

In the same way, the negative net preference is computed, formally:

$$\forall s_i \in \lambda^2(\alpha_r) \quad \Pi^2(\alpha_r, s_i) = \Pi^-(\alpha_r, s_i) = \frac{\sum_{s_k \in C_r \mid s_k P^- s_i} \Pi(s_k, s_i)}{|s_k \in C_r \mid s_k P^- s_i|}$$

$$for\ which \quad -1 \leq \Pi^-(\alpha_r, s_i) < 0$$

For the solutions that belong to $I(\alpha_r)$, the net preference is zero:

$$\forall s_i \in \lambda^3(\alpha_r) \quad \Pi^3(\alpha_r, s_i) = 0$$

In our example, the net preferences of cluster center α_1 over different solutions are:

$$\Pi^+(\alpha_1, s_5) = \frac{\overbrace{\Pi(s_2, s_5)}^{s_5 \in P^+(s_2)} + \overbrace{\Pi(s_3, s_5)}^{s_5 \in P^+(s_3)}}{2} = \frac{0.5 + 0.05}{2} = 0.275$$
$$\Pi^+(\alpha_1, s_3) = \Pi(s_2, s_3) = 0.3$$
$$\Pi^+(\alpha_1, s_4) = \Pi(s_2, s_4) = 0.3$$

$$\Pi^-(\alpha_1, s_1) = \frac{\overbrace{\Pi(s_2, s_1)}^{s_1 \in P^-(s_2)} + \overbrace{\Pi(s_3, s_1)}^{s_1 \in P^-(s_3)}}{2} = \frac{-0.05 - 0.5}{2} = -0.275$$
$$\Pi^-(\alpha_1, s_2) = \Pi(s_3, s_2) = -0.3$$
$$\Pi^+(\alpha_1, s_4) = \Pi(s_3, s_4) = -0.15$$

The measure of similarity between a solution and a cluster center is the same as measuring similarity between two solutions except that the notation of Π^+ and Π^- should be considered. Equation 5.7 shows the calculation of the similarity between the solu-

tion s_i and cluster center α_r.

$$Sim(\alpha_r, s_i) = \frac{\sum_{g=1}^{3} \sum_{\substack{t=1, \\ \forall s_t \in \Lambda^g(\alpha_r, s_i)}}^{|\Lambda^g(\alpha_r, s_i)|} (1 - abs(\Pi^g(\alpha_r, s_t) - \Pi(s_i, s_t)))}{n} \tag{5.7}$$

Note that for the solutions in the set $\Lambda^3(\alpha_r, s_i) = I(\alpha_r) \cap I(s_i)$ the value of both $\Pi^3(\alpha_r, s_t)$ and $\Pi(s_i, s_t)$ is zero. Thus, the third outer sum (i.e. $g = 3$) in Equation 5.7 simply counts the number of common solutions in the sets $I(\alpha_r)$ and $I(s_i)$.

For instance, in our example, the similarity of s_4 to the cluster center α_1 is:

$$\begin{aligned}
Sim(\alpha_1, s_4) &= \frac{\overbrace{(1 - abs(\Pi^+(\alpha_1, s_3) - \Pi(s_4, s_3)))}^{s_3 \in \Lambda^1(\alpha_1, s_4)}}{5} \\
&+ \frac{\overbrace{(1 - abs(\Pi^-(\alpha_1, s_1) - \Pi(s_4, s_1))) + (1 - abs(\Pi^-(\alpha_1, s_2) - \Pi(s_4, s_2)))}^{s_1, s_2 \in \Lambda^2(\alpha_1, s_4)}}{5} \\
&+ \frac{\overbrace{0}^{\Lambda^3(\alpha_1, s_4) = \{\}}}{5} \\
&= \frac{(1 - abs(0.3 - 0.15)) + (1 - abs(-0.275 + 0.7)) + (1 - abs(-0.3 + 0.3))}{5} \\
&= \frac{0.85 + 0.575 + 1}{5} = 0.485
\end{aligned}$$

The originality of our proposed similarity measure comes from the definition of the similarity that takes into account the preference structure defined by the decision maker. So, two solutions are considered as similar solutions in the sense that they are preferred (P^+), not preferred (P^-) and indifferent (I) to more or less the same solutions and with the same degree (Π).

For better understanding the similarity relations between solutions, we propose a visual representation of the preference structure. Figure 5.11 shows the visual form of the preference structures for 5 solutions in our example. For each solution, a circular bar chart is drawn that shows the preference relations between the corresponding solution and all the other solutions. The name written in the center of the bar chart indicates its corresponding solution. In each bar plot, the number of bars is equal to the total number of solutions (n) (one bar per solution). Solutions from s_1 to s_n are placed clockwise in the circular bar chart. For each solution s_i, the length of the colored portion of the bar corresponding to s_k is equal to value of $\Pi(s_i, s_k)$. If $\Pi(s_i, s_k)$ is greater than zero (means that $s_i \, P^+ s_k$) a blue color is used. When $\Pi(s_i, s_k)$ is less than zero (means that $s_i \, P^- s_k$) a red color is used. In the case that $s_i \, I \, s_k$, the length of the colored part is zero, since $\Pi(s_i, s_k) = 0$. By using the visual representation of

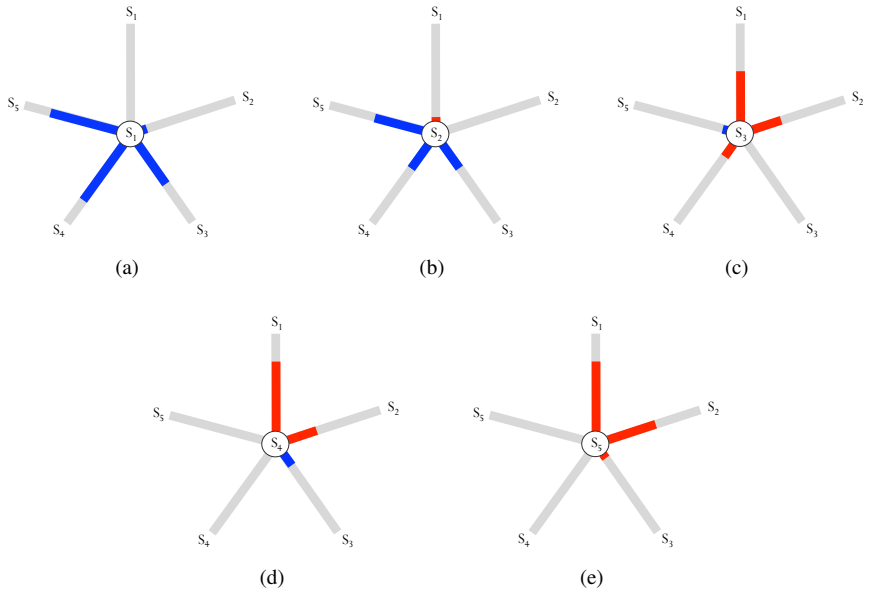


Figure 5.11: Visualization of the preference structure

preference structures, it is easy to find out the similarity relations between solutions without doing the respective mathematical calculations. Two solutions are as similar as their visual form is alike. For instance, we can see in Figure 5.11 that the bar charts of s_4 and s_5 are relatively similar to each other. Thus, we can conclude that their similarity is relatively high. This conclusion can be verified by looking at Table 5.4, which shows the computed similarity values for each pair of solutions. As can be seen in this table, the similarity measure between s_4 and s_5 is 0.76, which is fairly high (Note that the similarity measure is in the range [0,1]). From Figure 5.11, we can also see that the two solutions s_1 and s_5 have completely different visual forms and thus their similarity should be too small. As can be seen in Table 5.4 the similarity of s_1 and s_5 is zero.

Clustering Methods

The clustering methods can be broadly classified into two types: exclusive and hierarchical. In exclusive clustering, data are grouped into a set of disjoint clusters. Therefore, if an object is a member of a certain cluster then it cannot be included in another cluster. Instead, the hierarchical clustering is based on the union between the two nearest clusters. At the beginning, each object is in a separate cluster and then pairs of clusters are successively merged until all clusters have been merged into a single cluster that contains all objects.

Algorithm 5.1 Basic Leader-Follower Clustering

Notations: C_r = the r^{th} cluster, α_r = the center of the r^{th} cluster, s_i = the i^{th} solution, and N = number of clusters

Input: Distance threshold (θ) and a set of solutions (S)

Output: A set of N clusters (Cluster assignment for each solution)

Begin

$C_1 = \{s_1\}$, $N = 1$ // Assign the first solution to the first cluster

$i = 2$

Do accept new solution s_i

$r = \underset{r' \in \{1,2,\ldots,N\}}{argmin} (Dis(\alpha_r, s_i))$ // Find the nearest cluster center

If $Dis(\alpha_r, s_i) < \theta$

Then

$C_r = C_r + \{s_i\}$ // Assign s_i to the closet cluster

Update α_r

Else

$C_{N+1} = \{s_i\}$ // Create new cluster that contains s_i as its first member

$N = N + 1$

$i = i + 1$

Until no more solutions

End

VMODEX allows a decision maker to cluster Pareto optimal solutions with both exclusive and hierarchical methods. The similarity measure between the Pareto optimal points is our proposed preference similarity measure described in the previous section. However, in the clustering procedure, instead of the similarity measure, the corresponding distance measure $Dis(s_i, s_k) = 1 - Sim(s_i, s_k)$ is used.

Exclusive Clustering In this thesis, we use Leader-Follower approach for grouping the solutions in an exclusive way. The Leader-Follower algorithm [101] is one of the most popular exclusive methods and is simple and efficient for clustering data sets. The basic Leader-Follower clustering procedure is described in Algorithm 5.1. The decision maker must specify a threshold (θ) for the distance. The first solution is assigned to the first cluster. From then, for each solution, the nearest cluster center is found. If the distance is less than the threshold, the solution becomes a member of that cluster and the cluster center should be updated. Otherwise, if the distance exceeds the threshold θ, it means that no current cluster is sufficiently close to the solution and a new cluster is created having the current solution as its first member. Note that in

Algorithm 5.1, $\underset{x}{argmin} f(x)$ denotes the set of values of x for which $f(x)$ attains its smallest value.

The Distance threshold should be chosen carefully, since it implicitly specifies the number of clusters obtained at the end. A large threshold leads to a small number of large clusters, which contain solutions with less similarity. While with a small threshold the probability of creation of new clusters is higher and therefore it will produce a large number of small clusters, which is somewhat contradictory with the aim of clustering.

The drawback of the Leader-Follower clustering algorithm is that it is not deterministic. The initial order of the solutions may strongly influence the structure of clusters. To overcome this weakness, we sort the solutions by their total net preferences before starting the clustering process. The total net preference of solution s_i is calculated as:

$$\Pi_{total}(s_i) = \sum_{k=1}^{n} \Pi(s_i, s_k) \tag{5.8}$$

Where n is the number of solutions in the set. The sorting is performed in descending order. This means that the solution with the highest total net preference value is used as the first solution in the clustering procedure and the solution with the smallest Π_{total} value is taken as the last solution. For assigning a solution to a cluster, we use a deterministic way of choosing between clusters with an equal distance from the solution in question. If the solution that is in process of assigning a cluster has the same nearest distance to two or more cluster centers, it becomes a member of the cluster that contains the closest solution to the current solution in the ordered sequence of the solutions. For instance, consider the set $\{A, B, C, D, E, F\}$ as an ordered set of solutions. Suppose that A and D are in the same cluster (C_1) and B, C and E are in the other cluster (C_2). Assume that F has the same distance from the centers of both clusters. By applying our deterministic approach, it will be assigned to cluster C_2 since this cluster contains E which is closest solution to F in the ordered set.

In each cluster, the solution with the highest Π_{total} value is chosen as a representative solution. A representative solution can be considered as the most promising solution among the solutions in its cluster. Therefore, by applying the clustering, the Pareto optimal set can be reduced to a smaller subset of representative solutions.

For each cluster center, we define its objective values to be the average of all points in the cluster. For instance, if the r^{th} cluster contains two solutions $s_1 = (4, 3, 5)$ and $s_2 = (4, 1, 7)$, then the objective values of cluster center α_r is calculated as follows:

$$\alpha_r = (\frac{4+4}{2}, \frac{3+1}{2}, \frac{5+7}{2}) = (4, 2, 6)$$

To understand the similarity relation between clusters, we define a procedure to connect the clusters based on their similarity. First, a fully connected weighted graph G is created in such a way that each cluster is considered as a node in the graph and the edge weight between two nodes is the distance between the corresponding cluster

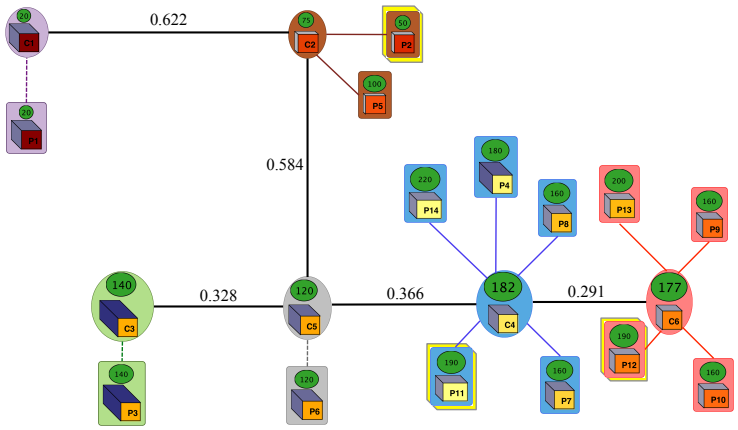


Figure 5.12: Visualization of Leader-Follower clustering

centers. Then a Minimum Spanning Tree (MST) is created for the graph G (A detailed description of MST is given in Chapter 4, Section 4.3.1). The clusters are connected together based on their connections in the corresponding MST. Thus, we get a tree of clusters with minimum weight (most similar connections).

We propose a visualization method that shows how the solutions are grouped in different clusters. This visualization allows decision makers to quickly and effectively analyze the result of clustering. Figure 5.12 shows the visualization of Leader-Follower clustering for the Pareto optimal solutions in our case study. Each cluster is shown by a different color. In Figure 5.12, the distance threshold is 0.3, which leads to the six clusters. Each solution is connected to the center of the cluster to which it belongs. The length of the connected link indicates the distance from the center. A shorter edge means closer to the center. In a cluster with only one solution, a dashed line is used for connecting the solution to the cluster center since the distance is zero (like C_1 in Figure 5.12). The cluster C_4 is the biggest cluster containing 5 solutions. In each cluster, the representative solution is highlighted by a yellow border such as solution $P11$ in cluster C_4. In Figure 5.12, the way that the cluster centers are connected together is the same as their connections in the corresponding MST. The length of the edge between two cluster centers represents the distance between them. The longer the edge implies a larger distance. The distance value between two connecting clusters is written above the corresponding edge. In Figure 5.12, clusters C_3, C_4, C_5 and C_6 are relatively close (similar) to each other and contain more than 75% (11 out of 14) of discovered Pareto optimal points.

Hierarchical Clustering The Leader-Follower clustering algorithm is efficient and conceptually simple, but it requires a specific distance threshold as input, which has a major effect on the clustering results. While the hierarchical clustering does not require any predetermined parameter and is deterministic. However, the complexity

Algorithm 5.2 Basic Agglomerative Hierarchical Clustering

Input: a set of n solutions

Output: A hierarchical cluster tree

Begin

Start with n clusters, each containing a single solution

Calculate the $n \times n$ similarity matrix between all pairs of clusters

Do accept new solution s_i

- Find the two clusters C_i and C_j such that their similarity is maximum (two closest clusters)
- Merge these clusters into a single cluster $C_r = C_i \cup C_j$
- Remove the two old clusters C_i and C_j
- Determine the similarities between the new cluster C_r and the remaining clusters

Until only one cluster is left, which contains all the solutions

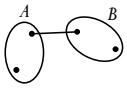
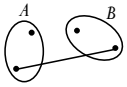
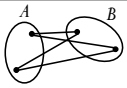
End

of hierarchical clustering increases at least quadratic with the number of data points. Several hierarchical clustering algorithms have been proposed such as SLINK [104], CURE [105] and CHAMELEON [106]. In this thesis, we use Agglomerative hierarchical clustering. The basic steps of Agglomerative hierarchical clustering are described in Algorithm 5.2.

The similarity of two clusters can be measured in three ways: single-linkage, complete-linkage and average-linkage. In the single-linkage clustering, the shortest distance from any member of one cluster to any member of another cluster is considered as the similarity between two clusters. In the complete-linkage method, the greatest distance between any two solutions in the different clusters determines the distance between the two clusters. In the average-linkage clustering, the distance between two clusters is computed as the average of distances between all pairs of solutions, in which one solution is in the first cluster and another solution is in the second cluster. Table 5.5 summarizes the definition of different linkage methods between two clusters A and B.

A hierarchical clustering is typically visualized as a binary tree (called dendrogram), which represents the hierarchical, pair-wise clustering of the items in the data set. Figure 5.13 shows the dendrogram obtained from the hierarchical clustering (single-linkage) of the Pareto optimal solutions in our case study. The individual solutions are placed at the bottom of the dendrogram and construct the leaf nodes. Each merge is represented by an internal node that connects the two joining clusters. The y-coordinate of the internal node shows the distance between two merging clusters. Each

Table 5.5: Definition of different linkage methods

Linkage Method	Definition	Formula
Single Linkage		$\min\{Dis(a,b) : a \in A, b \in B\}$
Complete Linkage		$\max\{Dis(a,b) : a \in A, b \in B\}$
Average Linkage		$\frac{1}{\lvert A\rvert\lvert B\rvert}\sum_{a\in A}\sum_{b\in B} Dis(a,b)$

internal node has exactly two subtrees of clustered solutions (a right and a left subtree). Moving from the bottom to the top of the dendrogram shows the history of merges during the clustering process. For example, we can see that in Figure 5.13 the two solutions $P9$ and $P10$ are merged first, and at the last merge the solution $P1$ is added to a cluster consisting of all the other solutions. To understand in each merging step, which solutions are considered for the similarity among the two joining clusters, the name of the two closest/ furthest solutions are written at the bottom of the corresponding internal node (this is applicable for only the single and complete linkage strategies).

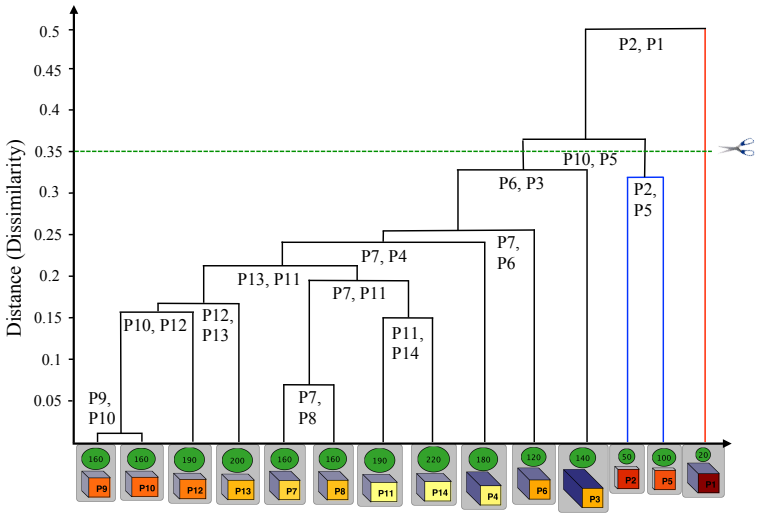


Figure 5.13: Visualization of hierarchical clustering

Once the complete hierarchical dendrogram is constructed, depending on the type of decision, different cutting policies can be applied on it, as follows:

- Cutting at a specific level of dissimilarity. In this case we get clusters of which the distance between the solutions inside the same cluster is less than the determined dissimilarity threshold. For example, in Figure 5.13 cutting the dendrogram at 0.35 yields 3 clusters: one large cluster containing 11 solutions (black lines), one cluster consisting of $P2$ and $P5$ (blue lines), and one cluster including only $P1$ (red line). Each cluster contains solutions of which the dissimilarity (distance) between them is less than 0.35.
- Cutting in such a way that a certain number of clusters remain. If the decision maker prefers to organize solutions in k cluster, he just should cut the $k-1$ links from the top. Therefore, the hierarchical clustering is flexible in the sense that the decision maker can easily change the number of clusters he is interested in, without repeating the clustering process. In Figure 5.13, cutting the dendrogram for getting 3 clusters produces the same clusters as cutting at level 0.35.
- Cutting the dendrogram at the point where the gap between two levels of dissimilarity, before and after merging a cluster to another one, is largest (the highest vertical line in the dendrogram). Such a large gap indicates that merging one more cluster significantly decreases the quality of the clustering. Therefore, cutting before this deterioration is desirable. In Figure 5.13 the greatest gap occurs at the last merging (red line), when the solution $P1$ is added to a cluster consisting of all the other solutions.

5.3.4 Ranking Problem

In the ranking approach, the decision problem is to order all the solutions in terms of their overall preference value, which is derived by taking into account all objectives. Several methods have been proposed for addressing the ranking problem. Here, we explain the three most commonly used ranking methods that are provided in VMODEX. The main differences between these ranking methods are 1) the normalization process for comparing all criteria on a common scale, and 2) the way that they aggregate the normalized objective values and their weights for obtaining an overall preference value for each solution.

For describing the ranking methods, we consider a MODM problem with a set of n solutions $s_i(i = 1, 2, ..., n)$ and m objective functions $f_j(j = 1, 2, ..., m)$. The performance of solution s_i on the objective function f_j is denoted by $f_j(s_i)$. We assume that all objective functions need to be minimized, that is, the smaller the value, the better it is. The relative importance of objective functions is indicated by the weight assigned to each objective. Suppose that w_j denotes the weight of the objective function f_j.

Weighted Sum Model (WSM)

The WSM method is the best known and simplest ranking method. However, it is applicable only when the decision criteria are expressed in identical units of measure. If criteria are not in the same unit, then normalization is required to provide a common scale for comparisons of the objectives. The normalized value of $f_j(s_i)$ can be obtained by:

$$\bar{f}_j(s_i) = \frac{f_j(s_i) - \min\limits_{k=1}^{n} f_j(s_k)}{\max\limits_{k=1}^{n} f_j(s_k) - \min\limits_{k=1}^{n} f_j(s_k)}$$

Where $\bar{f}_j(s_i)$ indicates the normalized performance of solution s_i on objective function f_j. This normalization approach linearly transforms all objective values in the range [0,1]. Thus, it preserves the relative order of magnitude of the original objective values.

The overall preference value of each solution (s_i) is calculated using the weighted sum of its normalized values over all objectives, as follows:

$$PV(s_i) = \sum_{j=1}^{m} w_j \bar{f}_j(s_i) \tag{5.9}$$

Since we consider the minimization problem, a smaller value of PV indicates the more preferred solution. For ranking the Pareto optimal solutions, they are sorted by their preference values (PV) in ascending order.

TOPSIS

TOPSIS (Technique for Order Preference by Similarity to Ideal solution) [107] is based on the concept that the most preferred solution should have the shortest distance from the positive ideal solution (IS^+) and the longest distance from the negative ideal solution (IS^-). TOPSIS ranks the solutions according to these two distances. The procedure of TOPSIS method can be summarized as follows:

1. The normalized objective values are computed by:

$$\bar{f}_j(s_i) = \frac{f_j(s_i)}{\sqrt{\sum_{k=1}^{n} f_j(s_k)^2}}$$

2. IS^+ and IS^- are obtained from the weighted normalized objective values. IS^+ is found by constructing a vector of best values and IS^- is calculated by con-

structing a vector of worst values as follows:

$$R_j(s_i) = w_j \bar{f}_j(s_i)$$

$$IS^+ = (R_1^+, ..., R_j^+, ..., R_m^+) \qquad R_j^+ = \min_{k=1}^{n} R_j(s_k)$$

$$IS^- = (R_1^-, ..., R_j^-, ..., R_m^-) \qquad R_j^- = \max_{k=1}^{n} R_j(s_k)$$

3. For each solution s_i, the Euclidian distance from the positive ideal solution (ED^+) and negative ideal solution (ED^-) is computed as follows:

$$ED^+(s_i) = \sqrt{\sum_{j=1}^{m}(R_j(s_i) - R_j^+)^2} \qquad ED^-(s_i) = \sqrt{\sum_{j=1}^{m}(R_j(s_i) - R_j^-)^2}$$

4. The overall preference value for each solution is calculated as follows:

$$PV(s_i) = \frac{ED^-(s_i)}{ED^-(s_i) + ED^+(s_i)} \qquad 0 \leq PV(s_i) \leq 1 \tag{5.10}$$

   $PV(s_i)$ indicates the relative closeness of the i^{th} solution to the positive ideal solution. Therefore, a greater value means that the solution is closer to IS^+ and thus is more preferred.

5. For ranking, solutions are sorted in descending order of their PV values.

To compare different solutions in a Pareto optimal set in terms of their distances from IS^+ and IS^- and their PV values, we visualize this information such as shown in Figure 5.14. The length of the blue bar represents the distance between the corresponding solution and IS^+ (ED^+). A longer bar indicates that the solution is further away from IS^+. Therefore, a smaller blue bar is desirable. Similarly, the length of the red bar shows the distance from IS^- (ED^-). Thus, a longer red bar is more favorable. The length of the green bar indicates the relative closeness to IS^+ (PV value). The longer the bar denotes the solution is closer to the IS^+ and thus is more preferred. In Figure 5.14, the solutions (from left to right) are sorted by their PV values in descending order. Therefore, the position of each solution is the same as its rank in the TOPSIS method. For each solution, besides the distance values (ED^+ and ED^-), the amount of difference between their objective values can be seen (color coding). Therefore, it is easy to find out which objective value(s) have a high impact on the distance values. For example, $P2$ has the same energy consumption as the IS+ (the color and size of the third dimension of their representing nodes are the same) and the difference between their costs is small. However, they are significantly different in the processing time (node color) and thus this objective has the highest impact on the distance value. In Figure 5.14, except $P12$ and $P3$, all the other solutions are closer to IS^+ than IS^- ($ED^+ < ED^-$) since their blue bars are shorter than their red bars.

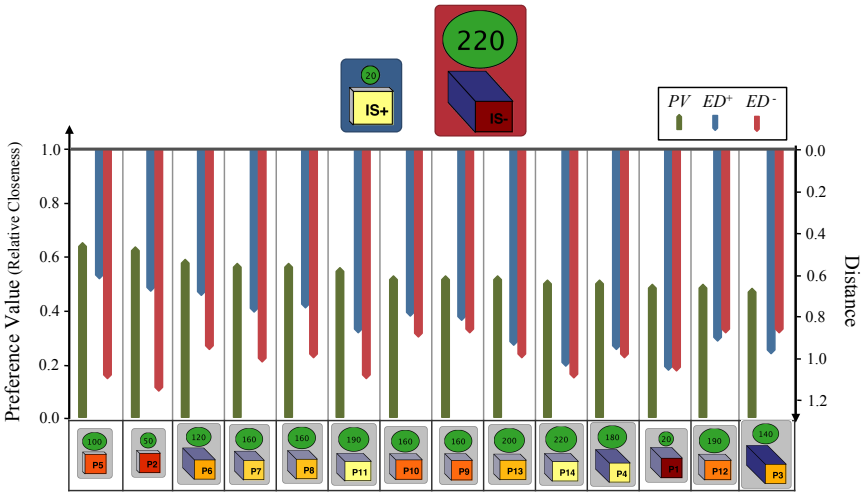


Figure 5.14: Visualization of TOPSIS ranking method

PROMETHEE

PROMETHEE (Preference Ranking Organization METHod for Enrichment Evaluations) [94, 108] is based on the pairwise comparisons of solutions for each objective. For each solution, it calculates positive and negative preference flows. The positive flow expresses how much a solution is dominating the other solutions and the negative flow indicates how much it is dominated by the other ones. Based on the balance of these two preference flows, the PROMETHEE ranks the solutions. PROMETHEE uses preference functions (see Section 5.2.3) to compute the degree of preference associated to the best solution in the case of pairwise comparisons. For each objective, a specific preference function must be defined. The procedure of PROMETHEE methodology can be summarized as follows:

1. Solutions are compared pairwise for each objective. Afterwards, preference functions are used to map their deviations on the objectives to the range [0 ,1]. For the objective function f_j, its corresponding preference function PF_j is used. The degree of preference of solution s_i over s_k on the j^{th} objective is given by:

$$P_j(s_i, s_k) = PF_j(f_j(s_i) - f_j(s_k)) \qquad 0 \leq P_j(s_i, s_k) \leq 1$$

   $P_j(s_i, s_k) = 0$ means no preference and $P_j(s_i, s_k) = 1$ indicates the strict preference of s_i over s_k. For small deviations, a small preference is allocated to the better solution and even possibly no preference if the deviation is negligible. The larger the deviation, the larger the preference.

2. The overall preference index of solution s_i over s_k for all objectives is computed

as follows:

$$\pi(s_i, s_k) = \sum_{j=1}^{m} w_j P_j(s_i, s_k)$$

3. For each solution s_i, the positive flow (ϕ^+) and the negative flow (ϕ^-) are defined as:

$$\phi^+(s_i) = \frac{1}{n-1} \sum_{k=1, k\neq i}^{n} \pi(s_i, s_k) \qquad \phi^-(s_i) = \frac{1}{n-1} \sum_{k=1, k\neq i}^{n} \pi(s_k, s_i)$$

4. For each solution s_i, the net flow $\phi(s_i)$ is considered as:

$$\phi(s_i) = \phi^+(s_i) - \phi^-(s_i) \qquad -1 \leq \phi(s_i) \leq 1 \tag{5.11}$$

The net flow is the balance between the positive and negative flows. The higher the net flow means the better the solution. $\phi(s_i) > 0$ means that the solution is more dominating all the other solutions on all objectives and $\phi(s_i) < 0$ indicates it is more dominated.

5. The complete ranking of all solutions from the best to the worst is obtained by sorting their ϕ values in descending order.

$\phi_j(s_i)$ is the single objective net flow for solution s_i when only the j^{th} objective is considered (100% of the total weight is assigned to objective j). It expresses how a solution s_i is dominating ($\phi_j(s_i) > 0$) or dominated ($\phi_j(s_i) < 0$) by all the other solutions on the j^{th} objective. $\phi_j(s_i)$ is defined as follows:

$$\phi_j(s_i) = \frac{1}{n-1} \sum_{k=1, k\neq i}^{n} [P_j(s_i, s_k) - P_j(s_k, s_i)] \tag{5.12}$$

$\phi_j(s_i)$ is particularly useful for analyzing the "quality" of solutions for each specific objective separately.

For comparing the Pareto optimal solutions based on their amount of domination on each objective, we visualize the $\phi_j(s_i)$ values such as shown in Figure 5.15 (the data are captured from our case study results explained in Section 5.1). In this figure, each solution is shown as an axis. The axes start from the same point, have the same length and are equi angular. Since the $\phi_j(s_i)$ values are in the range $[-1, 1]$, the center of the chart indicates -1 and the endpoints of axes denote 1. For each objective, a radar plot (also known as kiviat diagram) is drawn with a specific color. The points constructing a radar plot are the corresponding $\phi_j(s_i)$ values. Because the values greater than zero (means more dominating) or less than zero (means more dominated) have completely opposite meaning, the zero axis is highlighted. In this figure, the order of solutions (clockwise) is the same as their position in the PROMETHEE ranking method, which

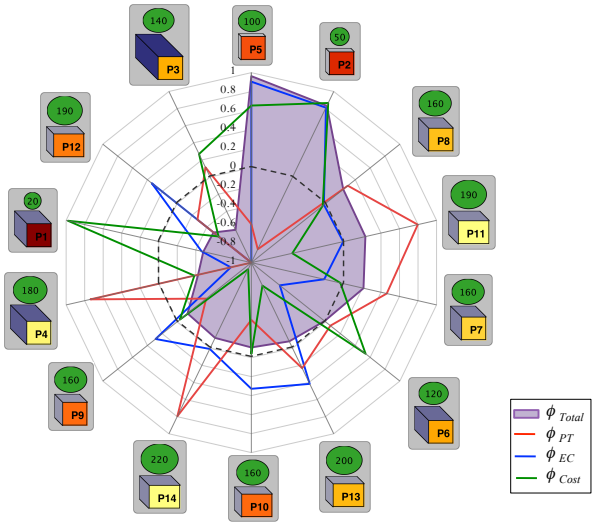


Figure 5.15: Visualization of ϕ values in PROMETHEE ranking method

is based on the sorted ϕ_{Total} values in descending order. In Figure 5.15, the filled radar plot shows the total ϕ values (considering all objectives). As can be seen in Figure 5.15, the net flow of the "cost" objective for the solution $P1$, $\phi_{Cost}(P1)$, is 1, which means it dominates all the other solutions in terms of the cost. On the other hand, the $\phi_{PT}(P1)$ is -1 which indicates that $P1$ is dominated by all the other solutions with respect to the processing time (PT). Its net flow for energy consumption (EC) is less than zero ($\phi_{EC}(P1) < 0$), which denotes that $P1$ is more dominated by all the other solutions in terms of energy consumption. As a result, its total net flow $\phi_{Total}(P1)$, which considers all the objectives is small and it is on the three worst solutions according to the PROMATHEE ranking method.

Showing ranking results

The outcomes of different ranking methods (described above) may not be the same since they use different procedures for ordering the Pareto optimal solutions. For comparing the results of different ranking methods and investigating the rank of a specific solution with respect to the different ranking methods, we propose a visualization approach as shown in Figure 5.16. This figure represents the results of ranking the Pareto optimal solutions obtained from our case study by applying three ranking methods: WSM, TOPSIS and PROMETHEE. Each ranking method is shown with a specific color. Assume there are n solutions in the Pareto optimal set. The circle is divided to n equal parts, where each part represents one solution. The number of nested circles is also n; each one denotes a rank position. The innermost one indicates the worst rank and the outermost one shows the best rank. For example, in Figure 5.16, the solutions $P2$ and $P5$ are in the top two for all ranking methods. Therefore, these

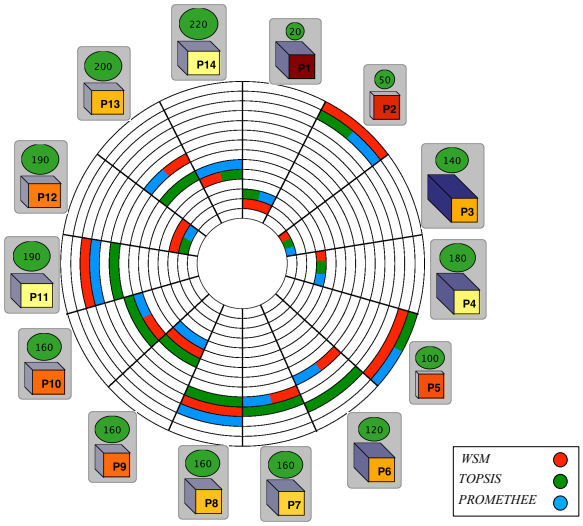


Figure 5.16: Visualization of ranking results

solutions can be a good choice for the final decision. However, the solution $P3$ is ranked as the worst solution by all the ranking methods. So, this solution is probably not appropriate as a final decision. Furthermore, from this figure we can see that all the solutions have almost the same rank in different ranking methods.

5.4 Conclusion

In this chapter, we discussed the final step in solving the multi-objective optimization problems, which is selecting the most preferred solution from the set of Pareto optimal solutions. After finding the Pareto optimal set by an optimization algorithm, we face another challenge: which solution from this set should be implemented. To solve this latter problem, Multi-Objective Decision Making (MODM) methods are used. These methods systematically apply the decision maker's preference information and provide some guidelines for choosing the most satisfying solution.

We described the four basic problem formulations in MODM, which are: choice, classification/sorting, clustering and ranking problems. For each problem formulation, some decision making methodologies are explained. We proposed a new method addressing the choice problem. Our method is based on the fuzzy dominance relations between the Pareto optimal solutions. Furthermore, we defined a new preference similarity measure for clustering the solutions. In our proposed similarity measure, unlike the conventional measures, the decision maker's preferences are integrated in the multi-objective cluster analysis. That is, two solutions are as similar as they are preferred, not preferred and indifferent to more or less the same solutions. Moreover,

we introduced a scheme for constructing the cluster centers, considering the properties of all solutions inside the same cluster. Some clustering approaches utilize the concept of cluster center for assigning the objects to different clusters.

For each MODM method, we proposed a new visualization approach that provides a detailed analysis of the results of MODM techniques. These visual representations allow decision maker to find out how and why a particular solution is considered as a most preferred solution with respect to a specific MODM method.

6 Conclusions

In this thesis, we addressed the visualization of multi-objective design space exploration of multi-processor system-on-chip architectures. Actually, we built a bridge between two previously separate research fields. We introduced the structural usage of data visualization into the field of embedded systems design. For performing a comprehensive study of the DSE process, we defined three separate stages and for each stage, we developed several methods and visualization techniques to provide users a rapid and more accurate analysis. A detailed description of each stage as well as the proposed visualization approaches were presented in the previous chapters.

In this chapter, we first summarize the thesis and then we describe the remaining open issues and future directions.

6.1 Outlook

In this thesis, we presented our novel interactive visualization tool, called VMODEX. All the proposed methods and visualization techniques for supporting the design space exploration of embedded systems are incorporated in VMODEX. The work presented in this thesis can be summarized as follows.

In Chapter 2 we gave an overview of the background information necessary for the rest of the thesis. We first described the basic knowledge about multi-objective optimization problems. Then, we explained the multi-objective optimization problem in the context of design space exploration of embedded systems. Afterwards, we discussed the benefits of using visualization techniques for exploring and analyzing the large amounts of data. Finally, we concluded this chapter and illustrated the need for employing efficient visualization methods for interpreting and gaining insight into the DSE results.

In Chapter 3 we introduced our tree model of the design space. In our DSE tree, both the design parameters and criteria are shown in a single view. Several interactive capabilities are provided to be able to handle large design spaces and allow designers to look at the data from different perspectives and at multiple levels of abstraction.

Furthermore, in our DSE tree model, we defined the concepts of subspaces and local Pareto points, which are new concepts in the multi-objective DSE process and have not been considered before. In VMODEX, besides the techniques provided for visualizing the DSE results, additional capabilities are developed to understand the dynamic search behavior of heuristic searching algorithms that are typically used in the DSE process. This chapter supports the second stage of multi-objective DSE, which is exploring the design space and analyzing the results.

Chapter 4 is dedicated to the first stage of multi-objective DSE, which is evaluating the performance of different multi-objective optimizers and finding the best one for a specific problem. In this chapter, we introduced various performance metrics (both existing and new metrics) and their visualization methods we have provided in VMODEX. In multi-objective optimization problems, several distinct goals need to be achieved and therefore there cannot be a single quality measure that indicates the performance of an optimization algorithm in an absolute sense. Thus, various metrics need to be used to perform a comprehensive analysis of the performance of an optimization approach. Most of the current performance measures concentrate on evaluating the quality of found Pareto optimal solutions in the objective space and measuring the behavior of the optimization algorithm in the decision space has mostly been disregarded. In this thesis, we turned the focus of attention from exclusively evaluating optimization success in the objective space to also considering the decision space. In this chapter, we defined new goals and subsequently new metrics to evaluate the behavior of optimization methods in the decision space.

In Chapter 5 we addressed the last stage of multi-objective DSE, which is the decision making process. As the last step of the multi-objective DSE process, the decision maker should select the most preferred design point from the set of Pareto optimal points for the final implementation. In this chapter, we explained the Multi-Objective Decision Making (MODM) methods that are provided in VMODEX and can help decision makers to understand the trade-offs between different criteria and select the final solution for the implementation. Furthermore, we proposed new visualization approaches, which provide the visual interpretation and detailed analysis of the results of the MODM methods. In this chapter, we described the four basic problem formulations in MODM, which are: choice, classification/sorting, clustering and ranking problems. For each problem formulation, some decision making methodologies and their proposed visualizations are explained. In this chapter, we defined a new method addressing the choice problem, which is based on the fuzzy dominance relations between the Pareto optimal points. Furthermore, we proposed a new preference similarity measure for clustering the solutions. In our proposed similarity measure, unlike the conventional measures, the decision maker's preferences are integrated in the multi-objective cluster analysis. In addition, we introduced a scheme for constructing the cluster centers, considering the properties of all solutions inside the same cluster. Some clustering approaches utilize the concept of cluster center for assigning the objects to different clusters.

6.2 Open Issues and Future Directions

The research presented in this thesis is still in progress. There are several interesting further research directions in terms of extending the proposed visualization techniques to support additional aspects of the design space exploration process. For instance, in this thesis, we consider the problem of the mapping of only one parallel application onto a heterogeneous MPSoC architecture. However, embedded systems often need to support multiple applications simultaneously. There is an opportunity to extend our proposed tree model of the design space to be able to support the design space exploration of embedded systems with multiple target applications. The work in [109, 110] introduced a scenario based design methodology for embedded systems with multiple applications. They distinguished two types of dynamic behavior: intra-application and inter-application. The intra application dynamism describes the changing behavior of an individual application, whereas inter application dynamism describes which subset of applications can run concurrently. New visualization approaches can be developed to enable designers to gain a better understanding of the dynamic behavior both within and between applications.

Another possible research direction is providing the interaction between the visualization and simulator. Instead of having a one-way flow of information from simulation to visualization, it is useful to provide mechanisms for interactive feedback from visualization to the simulator. This kind of interaction allows designer to directly steer the simulation process. For instance, by using the visualization, the designer may notice that in some parts of the design space there is no evaluated design point. The searching algorithm could not access to those parts of the design space. However, if the designer is interested to know about the properties of solutions in some unexplored parts of the design space, he should be able to interact with the simulator to generate some random design points in those parts and then continue the searching process for a specified number of iterations. Another example of possible interaction with the simulator is requesting to evaluate a specific design point that has not been evaluated during the exploring process. For instance, in investigating the effect of different mappings on the design criteria, some desired design points might not be visited by the searching algorithm. So, the designer can interact with the simulator and request it to evaluate those particular design points.

Another open issue that is not addressed in this work is proposing visualization methods that take into account the average behavior of several runs of an optimization algorithm. The visualization techniques that we explained in Chapter 4, address the analysis of results from a one-shot algorithm execution. Nevertheless, the same as comparing different algorithms, several runs of the same algorithm can also be evaluated and compared by our proposed visualization techniques. However, in our future work, we are going to extend our visualization approaches in order to show the average performance of an algorithm during numerous runs (with respect to a specific aspect) in a single view.

It should be mentioned that our work in this thesis is just one possible approach for

visualizing the multi-objective DSE process. It is indeed an open issue in this domain to examine other alternatives for visualization and compare the efficiency of different approaches. Furthermore, as a future work, we intend to ask end users to use our visualization tool for analyzing their DSE results. Their feedback will be used to evaluate as well as improve the efficiency and effectiveness of our tool.

Bibliography

[1] C. Erbas, S. Cerav-erbas, A. D. Pimentel, Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 358–374.

[2] M. Palesi, T. Givargis, Multi-objective design space exploration using genetic algorithms, in: Proceedings of the tenth international symposium on Hardware/software codesign, CODES '02, ACM, New York, NY, USA, 2002, pp. 67–72.

[3] A. Giuseppe, C. Vincenzo, P. Maurizio, A multi-objective genetic approach for system-level exploration in parameterized systems-on-a-chip, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24 (4) (2005) 635–645.

[4] G. Palermo, C. Silvano, V. Zaccaria, Multi-objective design space exploration of embedded systems, J. Embedded Comput. 1 (2005) 305–316.

[5] P. Marwedel, B. Sirocic, Multimedia components for the visualization of dynamic behavior in computer architectures, in: Computer architecture education, ACM, New York, NY, USA, 2003.

[6] C. Yehezkel, W. Yurcik, M. Pearson, D. Armstrong, Three simulator tools for teaching computer architecture: Little man computer, and rtlsim, ACM Journal of Educational Resources in Computing 1 (4) (2001) 60–80.

[7] R. Ibbett, Computer architecture visualisation techniques, Microprocessors and Microsystems 23 (1999) 291–300.

[8] K. Berkbigler, B. Bush, K. Davis, N. Moss, S. Smith, A la carte: A simulation framework for extreme-scale hardware architectures, in: IASTED International Conference on Modeling and Simulation, Palm Springs, CA, 2003.

[9] P. Coe, F. W. Howell, R. Ibbett, L. Williams, A hierarchical computer architecture design and simulation environment, ACM Transactions on Modeling and Computer Simulation 8 (4) (1998) 431–446.

[10] T. A. Diep, J. P. Shen, Vmw: A visualization-based microarchitecture workbench, Computer 28 (1995) 57–64.

[11] C. Stolte, R. Bosch, P. Hanrahan, M. Rosenblum, Visualizing application behavior on superscalar processors, in: IEEE Symposium on Information Visualization, IEEE Computer Society, Washington, DC, USA, 1999, pp. 10–17.

[12] W. Fang, C.-L. Wang, W. Zhu, F. C. M. Lau, Pat: a postmortem object access pattern analysis and visualization tool, in: IEEE International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, USA, 2004, pp. 379–386.

[13] E. van der Deijl, G. Kanbier, O. Temam, E. D. Granston, A cache visualization tool, Computer 30 (7) (1997) 71–78.

[14] Y. Yu, K. Beyls, E. H. D'Hollander, Visualizing the impact of the cache on program execution, in: Fifth International Conference on Information Visualisation, London, England, 2001, pp. 336–341.

[15] B. Quaing, J. Tao, W. Karl, Yaco: A user conducted visualization tool for supporting cache optimization, in: High Performance Computing and Communications (HPCC), Sorrento, Italy, 2005, pp. 694–703.

[16] Y. Yu, E. H. D'Hollander, Loop parallelization using the 3d iteration space visualizer, Journal of Visual Languages and Computing 12 (2) (2001) 163–181.

[17] H. Hlavacs, D. F. Kvasnicka, C. W. Ueberhuber, Clue-a tool for cluster evaluation, in: Distributed and Parallel Systems, Balatonfuered, Lake Balaton, Hungary,, 2000, pp. 61–64.

[18] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization, in: K. Giannakoglou, et al. (Eds.), Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, 2002, pp. 95–100.

[19] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature PPSN VI, Vol. 1917 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2000, pp. 849–858.

[20] J. D. Knowles, D. W. Corne, Approximating the nondominated front using the pareto archived evolution strategy, Evol. Comput. 8 (2000) 149–172.

[21] N. M. Salami, Ant colony optimization algorithm, Ubiquitous Computing and Communication Journal (UbiCC) 4 (3) (2009) 823–826.

[22] S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb, A simulated annealing-based multiobjective optimization algorithm: Amosa, IEEE Transactions on Evolutionary Computation 12 (2008) 269–283.

[23] C. Erbas, A. D. Pimentel, M. Thompson, S. Polstra, A framework for system-level modeling and simulation of embedded systems architectures, EURASIP J. Embedded Syst. 2007 (2007) 2–2.

[24] A. D. Pimentel, C. Erbas, S. Polstra, A systematic approach to exploring embedded system architectures at multiple abstraction levels, IEEE Trans. Comput. 55 (2006) 99–112.

[25] K. Deb, Optimization for Engineering Design: Algorithms and Examples, Prentice-Hall,, New Delhi, 1995.

[26] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd Edition, The Morgan Kaufmann Series in Data Management Systems, 2011.

[27] M. Gries, Methods for evaluating and covering the design space during early design development, Integr. VLSI J. 38 (2004) 131–183.

[28] I. P. Gent, T. Walsh, Towards an understanding of hill-climbing procedures for sat, in: Proceedings of the eleventh national conference on Artificial intelligence, AAAI'93, AAAI Press, 1993, pp. 28–33.

[29] C. A. Tovey, Hill climbing with multiple local optima, SIAM Journal on Algebraic and Discrete Methods 6 (3) (1985) 384–393.

[30] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, 1st Edition, Wiley, 2001.

[31] C. A. C. Coello, G. B. Lamont, D. A. V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[32] G. Bilchev, I. Parmee, The ant colony metaphor for searching continuous design spaces, in: T. Fogarty (Ed.), Evolutionary Computing, Vol. 993 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1995, pp. 25–39.

[33] M. Dorigo, S. Krzysztof, An Introduction to Ant Colony Optimization, IRIDIA Technical Report Series.

[34] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.

[35] P. J. M. Laarhoven, E. H. L. Aarts (Eds.), Simulated annealing: theory and applications, Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[36] L. Surhone, M. Timpledon, S. Marseken, Simulated Annealing: Global Optimization, Applied Mathematics, Global Optimum, Metropolis-Hastings Algorithm, Monte Carlo Method, Nicholas Metropolis, Quantum Annealing., Betascript Publishers, 2009.

[37] M. Mitchell, An Introduction to Genetic Algorithms , The MIT Press, Cambridge, MA, USA, 1998.

[38] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-vincentelli, E. Sentovich, K. Suzuki, B. Tabbara, Hardware-software codesign of embedded systems: the polis approach.

[39] B. Kienhuis, E. Deprettere, K. Vissers, P. van der Wolf, An approach for quantitative analysis of application-specific dataflow architectures, in: Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP '97, IEEE Computer Society, Washington, DC, USA, 1997, pp. 338–349.

[40] G. Kahn, The Semantics of a Simple Language for Parallel Programming, in: J. L. Rosenfeld (Ed.), IFIP Congress on Information Processing, New York, NY, USA, 1974, pp. 471–475.

[41] W. Werum, H. Windauer, Introduction to PEARL: process and experiment automation realtime language (2nd ed.), Heyden & Sons, Inc., Philadelphia, PA, USA, 1983.

[42] T. Grotker, System Design with SystemC, Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[43] E. Zitzler, L. Thiele, An evolutionary algorithm for multiobjective optimization: The strength Pareto approach, Tech. Rep. 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland (1998).

[44] C. M. Fonseca, P. J. Fleming, Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, in: Genetic Algorithms: Proceedings of the Fifth International Conference, Morgan Kaufmann, San Mateo, California, 1993, pp. 416–423.

[45] C. Groşan, Multiobjective adaptive representation evolutionary algorithm (marea) - a new evolutionary algorithm for multiobjective optimization, in: Applied Soft Computing Technologies: The Challenge of Complexity, Vol. 34 of Advances in Soft Computing, Springer Berlin / Heidelberg, 2006, pp. 113–121.

[46] J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1985, pp. 93–100.

[47] S. Sivanandam, S. Deepa, Introduction to genetic algorithms, Springer, 2007.

[48] J. D. Schaffer, R. Caruana, L. J. Eshelman, R. Das, A study of control parameters affecting online performance of genetic algorithms for function optimization, in: Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 51–60.

[49] G.-H. Hwang, W.-T. Jang, An adaptive evolutionary algorithm combining evolution strategy and genetic algorithm (application of fuzzy power system stabilizer), Advances in Evolutionary Algorithms.

[50] J. Yang, S. Wongsa, V. Kadirkamanathan, S. A. Billings, P. C. Wright, Self adaptive evolutionary algorithm based methods for quantification in metabolic systems, in: IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, CA, USA, 2004, pp. 260–267.

[51] N. Gershon, S. G. Eick, S. Card, Information visualization, interactions 5 (1998) 9–15.

[52] B. E. Rogowitz, D. A. Rabenhorst, J. A. Gerth, E. B. Kalin, Visual cues for data mining, in: Proceedings of the SPIE/SPSE Symposium on Electronic Imaging 2657, 1996, pp. 275–301.

[53] S. Wehrend, C. Lewis, A problem-oriented classification of visualization techniques, in: Proceedings of the 1st conference on Visualization '90, VIS '90, IEEE Computer Society Press, Los Alamitos, CA, USA, 1990, pp. 139–143.

[54] J. J. van Wijk, Views on visualization, IEEE Transactions on Visualization and Computer Graphics 12 (4) (2006) 421–433.

[55] A. Inselberg, B. Dimsdale, Parallel coordinates: A tool for visualizing multidimensional geometry, in: IEEE Visualization, 1990, pp. 361–378.

[56] J. Yang, W. Peng, M. O. Ward, E. A. Rundensteiner, Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets, in: Proceedings of the Ninth annual IEEE conference on Information visualization, INFOVIS'03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 105–112.

[57] L. Tweedie, B. Spence, H. Dawkes, H. Su, The influence explorer, in: Conference companion on Human factors in computing systems, CHI '95, ACM, New York, NY, USA, 1995, pp. 129–130.

[58] L. Tweedie, B. Spence, H. Dawkes, H. Su, Externalizing abstract mathematical models, in: Conference companion on Human factors in computing systems, CHI '96, ACM, 1996, pp. 406–412.

[59] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, S. Carpendale, Seven guiding scenarios for information visualization evaluation, Tech. Rep. 2011-992-04, University of Calgary, Computer Science (2011).

[60] C. North, Toward measuring visualization insight, IEEE Computer Graphics and Applications 26 (3) (2006) 6–9.

[61] C. Plaisant, The challenge of information visualization evaluation, in: Proceedings of the working conference on Advanced visual interfaces, AVI '04, ACM, New York, NY, USA, 2004, pp. 109–116.

[62] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, P. Hanrahan, Rivet: a flexible environment for computer systems visualization, SIGGRAPH Comput. Graph. 34 (1) (2000) 68–73.

[63] R. P. Bosch, Using visualization to understand the behavior of computer systems, Ph.D. thesis, Stanford University, Computer Science Dept, Berkeley, CA (2001).

[64] A. D. Mihalik, Vista: A visualization tool for computer architects, Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science (2004).

[65] T. Taghavi, M. Thompson, A. D. Pimentel, Visualization of computer architecture simulation data for system-level design space exploration, in: International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 149–160.

[66] T. D. Collins, Henson: a visualization framework for genetic algorithm users, in: IEEE (Ed.), Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, IEEE, 1999, pp. 562–569, cEC99 : July 6-9, 1999, Mayflower Hotel, Washington, D.C., USA, Volume 1.

[67] A. Kerren, T. Egger, Eavis: A visualization tool for evolutionary algorithms, in: IEEE Symposium on Visual Languages and Human-Centric Computing, IEEE Computer Society, Washington, DC, USA, 2005, pp. 299–301.

[68] A. S. Wu, K. A. D. Jong, D. S. Burke, J. J. Grefenstette, C. L. Ramsey, Visual analysis of evolutionary algorithms, in: In Proceedings of the 1999 Conference on Evolutionary Computation (CEC'99, IEEE Press, 1999, pp. 1419–1425.

[69] H. Pohlheim, Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization, in: Genetic and Evolutionary Computation Conference, Morgan Kaufmann, Los Altos, CA, 1999, pp. 533–540.

[70] T. D. Collins, Applying software visualization technology to support the use of evolutionary algorithms, Journal of Visual Languages and Computing 14 (2) (2003) 123–150.

[71] J. W. Sammon, A nonlinear mapping for data structure analysis, IEEE Transactions on Computers 18 (5) (1969) 401–409.

[72] E. Hart, P. Ross, Gavel - a new tool for genetic algorithm visualization, IEEE Trans. Evolutionary Computation 5 (4) (2001) 335–348.

[73] S. I. Ito, Y. Mitsukura, H. N. Miyamura, T. Saito, M. Fukumi, A visualization of genetic algorithm using the pseudo-color, in: ICONIP (2), Kitakyushu, Japan, 2007, pp. 444–452.

[74] B. De Smedt, G. Gielen, Watson: a multi-objective design space exploration tool for analog and rf ic design, in: IEEE Custom Integrated Circuits Conference (CICC), Orlando, USA, 2002, pp. 31–34.

[75] C. Ware, Information Visualization: Perception for Design, 1st Edition, Morgan Kaufmann, 2000.

[76] B. Shneiderman, The eyes have it: A task by data type taxonomy for information visualization, in: IEEE Symposium on Visual Languages, Boulder, CO , USA, 1996, pp. 336–343.

[77] A. D. Pimentel, S. Polstra, F. Terpstra, A. W. v. Halderen, J. E. Coffland, L. O. Hertzberger, Towards efficient design space exploration of heterogeneous embedded media systems, in: Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation - SAMOS, Springer-Verlag, London, UK, UK, 2002, pp. 57–73.

[78] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto evolutionary algorithm, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[79] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms - a comparative case study, in: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, PPSN V, Springer-Verlag, London, UK, 1998, pp. 292–304.

[80] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Ph.D. thesis, ETH Zurich, Switzerland (1999).

[81] T. Okabe, Y. Jin, B. Sendhoff, A Critical Survey of Performance Indices for Multi-Objective Optimisation, in: Proceedings of IEEE Congress on Evolutionary Computation (CEC-2003), IEEE Press, 2003, pp. 878–885.

[82] J. Knowles, D. Corne, On metrics for comparing non-dominated sets, in: IEEE Congress on Evolutionary Computation, 2001, pp. 711–716.

[83] K. Ang, G. Chong, Y. Li, Visualization technique for analyzing non-dominated set comparison, in: 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, 2002, pp. 36–40.

[84] D. A. Van Veldhuizen, Multiobjective evolutionary algorithms: classifications, analyses, and new innovations, Ph.D. thesis, Graduate School of Engineering of the Air Force Institute of Technology (June 1999).

[85] J. R. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization (1995).

[86] R. C. Prim, Shortest connection networks and some generalizations., Bell Systems Technical Journal (1957) 1389–1401.

[87] M. L. Fredman, R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the ACM 34 (1987) 596–615.

[88] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proceedings of The American Mathematical Society 7 (1956) 48–48.

[89] O. Boruvka, O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary), Práce Mor. Prírodoved. Spol. v Brne III 3 (1926) 37–58.

[90] W. W. Esty, J. D. Banfield, The box-percentile plot, Journal of Statistical Software 8 (17) (2003) 1–14.

[91] T. Saaty, The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation, McGraw-Hill, New york, 1980.

[92] T. Saaty, How to make a decision: The Analytic Hierarchy Process, European Journal of Operational Research 48 (1990) 9–26.

[93] J. P. Brans, P. Vincke, B. Mareschal, How to select and how to rank projects: The promethee method, European Journal of Operational Research 24 (1986) 228–238.

[94] J. P. Brans, P. Vincke, A preference ranking organisation method: (the promethee method for multiple criteria decision-making), Management Science 31 (6) (1985) 647–656.

[95] C. Zopounidis, M. Doumpos, A multicriteria decision aid methodology for sorting decision problems: The case of financial distress, Computational Economics 14 (3) (1999) 197–218.

[96] M. Doumpos, C. Zopounidis, Multicriteria Decision Aid Classification Methods, Kluwer Academic Publishers, Dordrecht, 2002.

[97] Y. De Smet, L. Montano Guzman, Towards multicriteria clustering: An extension of the k-means algorithm, European Journal of Operational Research 158 (2) (2004) 390–398.

[98] J. Figueira, Y. De Smet, J.-P. Brans, MCDA methods for sorting and clustering problems: Promethee TRI and promethee CLUSTER, Tech. Rep. TR/SMG/2004-002, SMG, Université Libre de Bruxelles (2004).

[99] P. Nemery de Bellevaux, Multicriteria clustering, Master's thesis, Université Libre de Bruxelles, Brussels, Belgium (2006).

[100] E. Fernandez, J. Navarro, S. Bernal, Handling multicriteria preferences in cluster analysis, European Journal of Operational Research 202 (3) (2010) 819–827.

[101] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification (2nd Edition), 2nd Edition, Wiley-Interscience, New York, 2001.

[102] J. B. MacQueen, Some methods for classification and analysis of multivariate observations, in: L. M. L. Cam, J. Neyman (Eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, 1967, pp. 281–297.

[103] L. Kaufman, P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, 9th Edition, Wiley-Interscience, New York, 1990.

[104] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method, The Computer Journal 16 (1) (1973) 30–34.

[105] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, SIGMOD Rec. 27 (1998) 73–84.

[106] G. Karypis, E.-H. S. Han, V. Kumar, Chameleon: A hierarchical clustering algorithm using dynamic modeling, IEEE Computer 32 (8) (1999) 68–75.

[107] Y.-J. Lai, T.-Y. Liu, C.-L. Hwang, Topsis for modm, European Journal of Operational Research 76 (3) (1994) 486–500.

[108] J. P. Brans, P. Vincke, B. Mareschal, How to select and how to rank projects: The promethee method for mcdm, European Journal of Operational Research 24 (1986) 228–238.

[109] P. v. Stralen, A. D. Pimentel, Scenario-based design space exploration of mpsocs, in: Proceedings of IEEE International Conference on Computer Design (ICCD '10), 2010.

[110] P. van Stralen, Scenario based design space exploration, Master's thesis, University of Amsterdam, Amsterdam, The Netherlands (2009).

List of Publications

International Journals

1. **T. Taghavi** and A.D. Pimentel, *VMODEX: A Novel Visualization Tool for Rapid Analysis of Heuristic- Based Multi-Objective Design Space Exploration of Heterogeneous MPSoC Architectures*, Elsevier Journal of Simulation Modeling Practice and Theory, to appear.
2. **T. Taghavi** and A.D. Pimentel, *Performance Assessment of Multi-Objective Evolutionary Optimizers: Metrics and Visualizations*, submitted to Journal of Multi-Criteria Decision Analysis: Optimization, Learning and Decision Support (WILEY).

International Conferences/Workshops

1. **T. Taghavi** and A.D. Pimentel, *Design Metrics and Visualization Techniques for Analyzing the Performance of MOEAs in DSE*, in the Proc. of the 11th International Conference on Embedded Computer Systems: Architectures, MOdeling and Simulation (SAMOS '11), Greece, July, 2011
2. **T. Taghavi** and A.D. Pimentel, *Techniques and Visualization Approaches for Analyzing Local and Global Pareto Optimal Sets in Multi-Objective Design Space Exploration*, in the Proc. of the Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures (PARMA '11) in conjunction with ARCS 2011, Lake Como, Italy, February, 2011.
3. **T. Taghavi** and A.D. Pimentel, *An Interactive Visualization Tool for the Analysis of Multi-Objective Embedded Systems Design Space Exploration*, in the Proc. of the Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (Rapido '11) in conjunction with HiPEAC'11, Crete, Greece, Jan. 2011.
4. **T. Taghavi** and A.D. Pimentel, *VMODEX: A Visualization Tool for Multi-Objective Design Space Exploration*, demo paper, in the Proc. of the International Conference on Field-Programmable Technology (FPT '10), Beijing, China, December 2010.
5. **T. Taghavi** and A.D. Pimentel, *Visualization of Multi-Objective Design Space Exploration for Embedded Systems*, in the Proc. of the Euromicro Conference on Digital System Design (DSD'10), France, Sept. 2010.
6. **T. Taghavi**, A.D. Pimentel, M. Thompson, *System-level MP-SoC Design Space Exploration using Tree Visualization*, in the Proc. of the IEEE/ACM/IFIP

Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia '09), Grenoble, France, Oct. 2009.

7. **T. Taghavi**, M. Thompson, and A. D. Pimentel, *Visualization of Computer Architecture Simulation Data for System-level Design Space Exploration*, in the Proc. of the International Symposium on Systems, Architectures, MOdeling and Simulation (SAMOS '09), Samos, Greece, July 2009.

Other Publications

1. M. Sabeghi, M. Naghibzadeh, **T. Taghavi**, *A Fuzzy Algorithm for Scheduling Soft Periodic Tasks in Preemptive Real-Time Systems*, New Mathematics and Natural Computation Journal, Vol. 3 No. 3, Nov 2007.

2. **T. Taghavi**, A. G. Bafghi, M. Kahani, *An Electronic Voting Scheme through Blind Signature*, in the Proc. of the International CSI Computer Conference (CSICC), Tehran, Iran, February 2007.

3. **T. Taghavi**, M. Kahani, A. G. Bafghi, *A Verifiable Multi-Authority E-Voting Scheme for Real World Environment*, in the Proc. of the International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE), Springer, December 2006.

4. M. Sabeghi, M. Kahani, **T. Taghavi**, *A Fuzzy Replacement Policy for Web Proxy Caches*, in the Proc. of the International e-Conference on Computer Science (IeCCS06), June 2006.

5. M. Sabeghi, M. Naghibzadeh, **T. Taghavi**, *Scheduling Non-Preemptive Periodic Tasks in Soft Real-Time Systems using Fuzzy Inference*, in the Proc. of the 9th IEEE International Symposium on Object and component-oriented Real- time distributed Computing (ISORC), Gyeongju, South Korea , April 2006.

6. M. Sabeghi, H. Deldari, V. Salmani, M. Bahekmat, **T. Taghavi**, *A Fuzzy Algorithm for Real-Time Scheduling of Soft Periodic Tasks on Multiprocessor Systems*, in the Proc. of the IADIS International Conference on Applied Computing, San Sebastian, Spain, February 2006.

7. M. Sabeghi, M. Naghibzadeh, **T. Taghavi**, *A Fuzzy Algorithm for Scheduling Soft Periodic Tasks in Preemptive Real-Time Systems*, in the Proc. of the International Conference on Systems, Computing Sciences and Software Engineering (SCSS), Springer, December 2005.

Visualisatie van Heuristische Ontwerpruimte Exploratie van Embedded Systemen

Toktam Taghavi Razavi Zadeh

Samenvatting

Het ontwerp van moderne embedded systemen is zeer complex. Vaak moeten meerdere en tegenstrijdige criteria simultaan worden geoptimaliseerd. Voorbeelden van deze criteria zijn snelheid, energie, kosten en gewicht. Om aan een dergelijk breed spectrum van non-functionele criteria te kunnen voldoen, hebben moderne embedded systemen vaak een heterogene systeem architectuur. Ze bestaan uit componenten die variëren van volledig programmeerbare processor kernen voor het bieden van flexibiliteit, tot volledige toegewijde hardware componenten voor tijdkritieke applicatie taken. Dit leidt tot een hoge mate van complexiteit van embedded systeem architecturen. Hierdoor worden ontwerpers gedwongen om systemen te modelleren en simuleren met als doel de gehele ruimte van mogelijke ontwerpkeuzes te doorzoeken. Een dergelijke ontwerpruimte exploratie is met name van essentieel belang voor de vroegtijdige fasen van het ontwerp traject waar de ontwerpruimte het grootst is en verkeerde ontwerpbeslissingen verregaande consequenties kunnen hebben op het eindproduct. Deze vroegtijdige ontwerpruimte exploratie wordt vaak op een hoog abstractie niveau uitgevoerd. Hierdoor blijft zowel de modeleertijd als de simulatietijd zo klein mogelijk.

Door de exponentiële grootte van de ontwerpruimte in hedendaagse embedded systemen is het ondoenlijk om elke mogelijke ontwerpkeuze te evalueren. Om deze reden worden vaak heuristische zoekmethoden gebruikt om zodoende met een beperkt aantal evaluaties de ontwerpruimte te doorzoeken voor optimale ontwerpkeuzes. Echter, dit beperkte doorzochte deel van de ontwerpruimte is nog steeds groot. Vandaar dat het interpreteren van deze resultaten, het trekken van de juiste conclusies en het begrijpen van hoe de ontwerpruimte is doorzocht door de heuristische zoekmethoden nog steeds extreem omslachtig kan zijn. Een dergelijke analyse is echter essentieel voor de ontwerper omdat het inzicht biedt hoe de ruimte van mogelijke ontwerpkeuzes eruit ziet.

Dit proefschrift adresseert het probleem van het interpreteren en analyseren van de grote hoeveelheden data die worden gegenereerd gedurende de ontwerpruimte exploratie van embedded systemen. Hierbij wordt data visualisatie ingezet als middel om de exploratie data te interpreteren en te visualiseren. Hiermee introduceren we het

structurele gebruik van visualisatie technieken in het ontwerp van embedded systemen, waarbij we specifiek ingaan op het gebruik van de visualisatie gedurende de ontwerpruimte exploratie. We hebben alle ontwikkelde technieken gecombineerd in een enkele applicatie: VMODEX. In zekere zin is er met dit werk een koppeling gemaakt tussen twee gescheiden onderzoeksgebieden: data visualisatie en het ontwerp van embedded systemen.

Voor een uitgebreide analyse van ontwerpruimte exploratie proces, definiëren we drie verschillende fases. Voor elke fase hebben we verschillende methodes en visualisatie technieken ontwikkeld om gebruikers een snelle en accurate analyse te bieden. De eerste fase betreft het ontwikkelen van een optimalisatie methode dat op een efficiënte en effectieve manier de ontwerpruimte doorzoekt. De ontwerpers van dit soort algoritmes kunnen VMODEX gebruiken om een uitgebreide studie te doen naar de kwaliteit van verschillende multi-criteria optimalisatie algoritmes, vanuit verschillende perspectieven, om het beste algoritme te vinden voor hun specifieke probleem.

In de tweede fase wordt het meest efficiënte algoritme gebruikt om een bepaalde ontwerpruimte exploratie te doen. VMODEX biedt vervolgens verscheidene mogelijkheden om de resultaten van deze ontwerpruimte exploratie te analyseren en te interpreteren. Deze analyse kan worden uitgevoerd vanuit verschillende perspectieven en op meerdere abstractieniveaus, wat een goed inzicht verschaft in het landschap van de ontwerpruimte.

De derde en laatste fase is het uitzoeken van de ontwerpkeuze die de meeste voorkeur geniet. Verscheidene benaderingen tot de besluitvorming zijn geïmplementeerd in VMODEX. Hierdoor worden de ontwerpers ondersteund in het kiezen van het uiteindelijke ontwerp. Daarnaast wordt er inzicht verschaft in de afwegingen tussen de verschillende criteria, wat uiteindelijk kan leiden tot het maken van betere beslissingen.

About the Author

Toktam Taghavi Razavi Zadeh was born in Mashhad, Iran, on August 12, 1982. In September 2004, she received her B.Sc. degree in Computer Engineering (Software) from Ferdowsi University of Mashhad, Mashhad, Iran. In February 2007, she obtained her M.Sc. degree in Computer Engineering from the same university.

Afterwards, She joined the Computer Systems Architecture (CSA) group in the University of Amsterdam as a PhD candidate in October 2007. She conducted her PhD research under the supervision of Dr. Andy D. Pimentel. Her doctoral research focused on the visualization of multi-objective design space exploration process of embedded systems.

Her research interests include, Visualization, Embedded systems, System-level design, Multi-objective optimization, Heuristic search techniques, Multi-objective decision making.