Visualization of heuristic-based multi-objective design space exploration of embedded systems
Taghavi Razavi Zadeh, T.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
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, $\sigma_{mst}$ and DFPOS), which eval-
4.1. INTRODUCTION

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 Chapter 2, 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.

![Diagram](f1.png)

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 optimum 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
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](image-url)
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 (\(PF_{\text{known}}\)) from the true Pareto front (\(PF_{\text{true}}\)). \(PF_{\text{true}}\) 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 \cite{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 \(PF_{\text{known}}\) that are not member of the \(PF_{\text{true}}\) as follows:

\[
ER = \frac{1}{|PF_{\text{known}}|} \sum_{i=1}^{|PF_{\text{known}}|} e_{s_i} = \begin{cases} 0 & s_i \in PF_{\text{true}} \\ 1 & s_i \notin PF_{\text{true}} \end{cases}
\] (4.1)

Where \(|\cdot|\) denotes the number of elements in a set and \(s_i\) indicates the \(i^{th}\) solution in the \(PF_{\text{known}}\). 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 \(PF_{\text{known}}\) are member of the \(PF_{\text{true}}\) 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 \(PF_{\text{known}}\) is in the \(PF_{\text{true}}\), 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 \(PF_{\text{true}}\), so its error ratio is 0.5. Algorithm \(A_2\) finds 8 Pareto optimal solutions of which 3 of them are in \(PF_{\text{true}}\) 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 \(PF_{\text{known}}\) belongs to the \(PF_{\text{true}}\), this metric finds an average distance of the solutions of \(PF_{\text{known}}\) from \(PF_{\text{true}}\):

\[
GD = \frac{1}{|PF_{\text{known}}|} \sqrt{\sum_{i=1}^{|PF_{\text{known}}|} d_i^2} \] (4.2)

The distance measure \((d_i)\) is the Euclidean distance (in the objective space) between the solution \(s_i \in PF_{\text{known}}\) and the nearest member of \(PF_{\text{true}}\). It is clear that a value
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

4.3.1. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

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 PFknown are in PFtrue. A Pareto optimal set having a smaller value of GD is better. The drawback of this metric is that when all solutions in PFknown 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 A1 finds 2 Pareto optimal solutions of which both of them are in PFtrue and algorithm A2 finds 7 Pareto optimal solutions of which all of them are in PFtrue. The value of GD for both of them is zero. However, it is clear that the algorithm A2 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 PFknown, which are relatively close to the PFtrue, via a weighted sum, as follows:

\[
WSGR = \frac{w_1}{|PFtrue|} \sum_{i=1}^{P|PFtrue|} f_{s_i} + \frac{w_2}{|PFknown|} \sum_{i=1}^{C|PFknown|} c_{s_i} \quad (4.3)
\]

\[
f_{s_i} = \begin{cases} 
1 & s_i \in PFknown \\ 
0 & s_i \notin PFknown 
\end{cases}
\]

\[
c_{s_i} = \begin{cases} 
1 & d_i < \theta \\ 
0 & d_i > \theta 
\end{cases}
\]

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 \((\theta)\) needs to be defined. If the minimum distance of solution \(s_i \in PFknown\) from \(PFtrue\) 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.

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

\( WSGR = 1 \) 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 \( PF_{true} \).
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

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 fluctu-
ation 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 solu-
tions. 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 visualiza-
tion 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 \( P^7 \) 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.

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

\[
\nabla = \prod_{m=1}^{M} (f_{m}^{\max} - f_{m}^{\min})
\]  

(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 \( \nabla \) 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.
4.3. Performance Metrics and Their Visualizations

Visualisation 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 $\nabla$-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.

σ_{max}-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
4.3. Performance Metrics and Their Visualizations

Figure 4.9: Illustration of the drawback of SS-metric

easily. Most of these metrics, such as the \( \Delta \) \([1]\) and \( \Delta' \) \([19]\) 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 (\( \sigma_{mst} \) 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 \( \sigma_{mst} \), 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, \( \sigma_{mst} \) can be computed using Equation 4.5.

\[
\sigma_{mst} = \sqrt{\frac{1}{|E| - 1} \sum_{i=1}^{|E|} (\bar{w} - w_i)^2}
\]

(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 \( \sigma_{mst} \) 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
4.3. **Performance Metrics and their Visualizations**

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, \( \bar{w} + \sigma_{\text{mst}} \) 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 \( \sigma_{\text{mst}} \) 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
4.3. Performance Metrics and Their Visualizations

Figure 4.11: Visualization of the $\sigma_{mst}$-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:

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

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.
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

The **(GD)** metric for measuring the closeness with the \( \sigma_{mst} \) metric for evaluating the diversity. As we described before, **GD** takes a small value for a good closeness and \( \sigma_{mst} \) 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 \((f_1 \text{ and } f_2)\) 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
4.3. Performance Metrics and Their Visualizations

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
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

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.

$$HV = \sum V_i = (0.2 \times 0.5 \times 0.8) + (0.2 \times 0.5 \times 0.0) + (0.2 \times 0.2 \times 0.6) + (0.5 \times 0.9 \times 0.9) = 0.08 + 0.00 + 0.024 + 0.495 = 0.599$$
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
4.3. Performance Metrics and their Visualizations

4.3.1 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 func-
tion 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 eval-
uate 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:

\[
CSR = \frac{|\text{Searched Combinations}|}{|\text{Total Feasible Combinations}|}
\]  

(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:

\[
CSR = \frac{|\text{Searched Cells}|}{|\text{Total Cells}|}
\]  

(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 is 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 4.17:

- The **Root** node represents the entire design space.
- The **Number of Processors Level** shows the number of processors as 1.
- The **Processor Type Level** indicates the processor type as ASIP.
- The **Number of Memories Level** displays the number of memories as 1.
- The **Memory Type Level** shows the memory type as DRAM.

**Figure 4.17:** an example design space visualized by VMODEX
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
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

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).
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

Figure 4.19: Histograms of two different distributions

(a) Trimodal distribution

(b) Uniform distribution with two outliers

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
4.3. Performance Metrics and their Visualizations

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 subspace 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:

\[
UR = \frac{|\text{Unique evaluated design points}|}{|\text{Total evaluated design points}|} \quad (4.9)
\]

Figure 4.22: A box percentile plot showing the typical pattern for skewed distributions
4.3. PERFORMANCE METRICS AND THEIR VISUALIZATIONS

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.
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 20th generation, the algorithm has a good performance in discovering new parts in the design space. However, after the 20th 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
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
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 ($\sigma_{\text{mst}}$-metric). The Random strategy has the best spread of Pareto optimal solutions in the objective space ($\nabla$-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.
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.
### 4.4. A Case Study

<table>
<thead>
<tr>
<th>Reference Set</th>
<th>NSGA-II-dh</th>
<th>NSGA-II</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>190</td>
<td>195</td>
<td>190</td>
<td>130</td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>195</td>
<td>190</td>
<td>130</td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>195</td>
<td>190</td>
<td>130</td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
</tbody>
</table>

Reference Set: NSGA-II-dh, NSGA-II, Random

**Note:** The diagram illustrates the flow or sequence between the reference sets and optimization algorithms, indicating the transition or interaction between them.
### 4.4. A Case Study

<table>
<thead>
<tr>
<th>Reference Set</th>
<th>NSGA-II-dh</th>
<th>NSGA-II</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Diagram Description:**
- The diagram illustrates the comparison of different reference sets and algorithms (NSGA-II-dh, NSGA-II, Random) in a case study.
- Each row represents a set of objectives or criteria.
- The connections between the sets indicate the performance or selection criteria across the algorithms.
- The diagram likely visualizes Pareto fronts or similar optimization results, showcasing how different methods perform relative to each other in achieving the objectives.

---

**Note:** The specific details and criteria within the diagram are not clearly visible due to the image quality, but the general structure and the comparison between the algorithms are evident. Further analysis or context is required for precise understanding.
4.4. A Case Study

<table>
<thead>
<tr>
<th></th>
<th>WSGR ((W_1 = 0.6, w_2 = 0.4))</th>
<th>ER</th>
<th>GD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NSGA-II</strong></td>
<td>0.6 ((6 / 17) + 0.4 (9 / 22) = 0.375)</td>
<td>16 / 22 = 0.727</td>
<td>0.0096</td>
</tr>
<tr>
<td><strong>NSGA-II-dh</strong></td>
<td>0.6 ((17 / 17) + 0.4 (17 / 17) = 1.0)</td>
<td>0 / 17 = 0.0</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>Random</strong></td>
<td>0.6 ((12 / 17) + 0.4 (13 / 21) = 0.671)</td>
<td>9 / 21 = 0.428</td>
<td>0.0083</td>
</tr>
</tbody>
</table>

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
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.
**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.

\[
\begin{align*}
\bar{w} \text{ (NSGA-II)} &= 0.080 & \sigma_{mst} \text{ (NSGA-II)} &= 0.102 \\
\bar{w} \text{ (NSGA-II-dh)} &= 0.097 & \sigma_{mst} \text{ (NSGA-II-dh)} &= 0.117 \\
\bar{w} \text{ (Random)} &= 0.083 & \sigma_{mst} \text{ (Random)} &= 0.104
\end{align*}
\]

*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.

\[ \text{HV(NSGA-II)} = 0.834 \]

\[ \text{HV(NSGA-II-dh)} = 0.845 \]

\[ \text{HV(Random)} = 0.841 \]

**Figure 4.31:** Comparing performance with \( HV \) metric
4.4. A Case Study

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.
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...
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 50th 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,
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 4.35:** Comparing Distribution of Design Points in the Decision Space using box-percentile plot
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 ratio 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-

![Dynamic Unique Ratio](image)

(a)

![Dynamic Cumulative Unique Ratio](image)

(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 ($\sigma_{\text{mst}}$ 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.