Genetic and Evolutionary Computation

Dhish Kumar Saxena
Sukrit Mittal
Kalyanmoy Deb
Erik D. Goodman

Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization

Springer

Genetic and Evolutionary Computation

Series Editors

Wolfgang Banzhaf, Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA

Kalyanmoy Deb, Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA

Dhish Kumar Saxena · Sukrit Mittal ·
Kalyanmoy Deb · Erik D. Goodman

Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization

Springer

Dhish Kumar Saxena
Department of Mechanical and Industrial Engineering
Indian Institute of Technology Roorkee
Roorkee, Uttarakhand, India

Sukrit Mittal
Franklin Templeton
Hyderabad, India

Kalyanmoy Deb
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, MI, USA

Erik D. Goodman
Michigan State University
East Lansing, MI, USA

ISSN 1932-0167 ISSN 1932-0175 (electronic)
Genetic and Evolutionary Computation
ISBN 978-981-99-2095-2 ISBN 978-981-99-2096-9 (eBook)
https://doi.org/10.1007/978-981-99-2096-9

Foreword

Evolutionary multi-objective optimization (EMO) has established itself as a major area of research and application within the broad field of Evolutionary Computation (EC). EMO uses natural evolutionary principles in a computer algorithm to solve various practical problems involving multiple conflicting criteria and finds a set of alternative optimal solutions. The concept is attractive, as it allows human decision makers to analyze these alternative high-performance solutions to pick a preferred solution for implementation, rather than making the overall multi-objective problem-solving approach completely automated. Over the past three decades, EMO has taken the idea of EC outside the realm of computer science and engineering and enabled multi-objective problem solving of various scientific, financial, and societal problems, including problems in astronomy, biology, economics, and medicine.

Machine learning (ML) is one of the most engaging research and application areas within computer science today. Its ability to find patterns, clusters, and hidden knowledge from data has allowed us to understand, model, and predict behaviors of complex systems that are difficult to analyze using conventional methods.

This book, entitled *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization* by Dhish Kumar Saxena, Sukrit Mittal, Kalyanmoy Deb, and Erik D. Goodman, brings together two emerging fields of computation—EMO and ML. If seen from the point of view of ML, EMO algorithms produce promising datasets in the form of populations across multiple generations, each comprising of high-performance solutions characterized by their decision variables, objective, and constraint values. The authors, through their years of collaboration, have employed contemporary ML methods to analyze such EMO population data to better understand the problem structure; understand as to what makes the solutions optimal; and to create new and often superior solutions by enhancing and balancing convergence and diversity properties. Besides compiling such contributions in a coherent manner, this book also sheds light on the potential ML-assisted enhancements to various operations of an EMO algorithm, as yet unexplored, providing many interesting ideas for future research.

The book is self-contained and written for both novices and experts in these fields. The topics are discussed in a clear and simple manner with illustrative graphs and tables, references, and other details, so that the readers can find relevant past studies and simultaneously get introduced to several new directions of research combining ML and EMO ideas in a single coherent framework.

I am particularly excited about the timing of this book. At a time when top EC conferences and journals are increasingly dedicating special tracks and special issues to the topics themed around ML-EMO collaborative research, this book may serve as an important resource for setting the foundation of this important EMO-ML intersection. While the book is mostly aimed at enhanced EMO research and applications, ML researchers may find EMO a fertile area for ML applications. In my opinion, the book should motivate everyone working in EMO and EC to make their methods more effective and efficient and those working in ML to discover a new and promising area to apply their future research.

Una-May O'Reilly
MIT Computer Science and Artificial
Intelligence Lab
Boston, MA, USA

Preface

Single-objective optimization algorithms had a monopolistic prevalence for decades until optimization problems with multiple conflicting objectives, such as cost and quality, started receiving greater interest in the 1970s. Most of the initial efforts to address these problems involved *scalarization*, where a single objective was formulated, allowing the use of a single-objective optimization algorithm. The downside of this approach was that it needed several runs of a single-objective optimization algorithm, each with a different weight vector for the objectives, before the objectives' trade-off could be witnessed. In the early 1990s, population-based evolutionary optimization algorithms were extended to solve multi-objective optimization problems. These algorithms soon became popular due to their ability to find and store multiple trade-off solutions in a single run. Their popularity is clearly evident through the large number of citations for evolutionary multi-objective optimization (EMO) papers in leading journals of evolutionary computation, a larger number of Ph.D. theses in the domain, and the emergence of a number of commercial EMO software.

Machine learning (ML) methods have gained prominence over the past three decades or so for various computational intelligence tasks, including classification, clustering, pattern recognition, modeling, prediction, autonomous detection, intelligent control, and other data analytics. Over the years, they have demonstrated their ability to extract and capture the inherent relationships between the input and output of a system by processing available data. So far, research and applications in the EMO and ML domains have been to a great extent independent of each other. However, some recent studies have begun to use one to complement the other; particularly there has been a rise in research in the 'evolutionary machine learning' in which evolutionary computation methods are employed to benefit ML methods and applications. The scope for EMO-based enhancements in ML methods is also natural and intuitive, since ML methods usually have to cater to conflicting goals. For instance, feature selection seeks to minimize the number of features, while maximizing their quality; model selection is driven by the trade-off between model complexity and approximation/classification accuracy; and the generation of a diverse set of Pareto-optimal models is desired for constructing ensembles. In comparison, though the efforts toward ML-assisted EMO in the last two decades or so have only been sporadic, its immense potential is increasingly being recognized lately. This book focuses on this latter aspect highlighting the use of ML methods in improving the performance of EMO algorithms and applications.

This book is an outcome of the SPARC project, funded by the Government of India, between the Indian Institute of Technology Roorkee, India, and Michigan State University, East Lansing, MI, USA. This project was envisioned to explore the immense potential for ML-based enhancements in the EMO domain. The motivation for this project was rooted in the recognition that an EMO algorithm can be viewed as a time-varying process in which a population of solutions, starting at random regions in the search space, gradually migrates toward the optimal regions. Hence, the chronological populations of solution vectors can be considered as time-series data which, if analyzed through ML methods, can help to capture efficient search directions, which in turn could be utilized to improve the search efficiency of an EMO algorithm. Notably, the successful accomplishment of the SPARC project's goals paved the way for this book, in that, the project's key deliverables constitute some of the core chapters in this book. Toward a more holistic perspective, and wider appeal to both the novice in the EMO domain, and the experienced researchers and practitioners, this book discusses the fundamental concepts associated with optimization (problem and algorithm types) sufficiently and includes existing studies beyond the project, including some of the preceding and most recent ones. In doing so, the chapters in this book cover ML interventions in all three phases related to the use of an EMO algorithm, namely the *search* phase, the *post-optimality* phase, and the *decision-making* phase.

This book is not intended to provide a comprehensive survey of ML-assisted EMO. Instead, it aims to highlight some of the key existing studies in a structured manner and to provide perspectives that may potentially trigger more interest in ML-EMO collaboration, toward the mutual benefit of both domains. The recent launch of ACM Transactions on Evolutionary Learning and Optimization augurs well for collaborative research across different subdomains of evolutionary computation and ML, toward effective and efficient practical problem solving.

Roorkee, India
East Lansing, MI, USA
February 2024

Dhish Kumar Saxena
Sukrit Mittal
Kalyanmoy Deb
Erik D. Goodman

Acknowledgements

First and foremost, the authors acknowledge the funding provided by the Government of India through the SPARC project (code: P66). The authors wish to sincerely thank the Dean, International Relations; Dean, Sponsored Research and Industrial Consultancy; Dean, Resources and Alumni Affairs; and Head, Department of Mechanical and Industrial Engineering, at IIT Roorkee, for their continuous support and for hosting the MSU investigators and researcher. The authors also thank the Department of Electrical and Computer Engineering at Michigan State University (MSU) and the Koenig Chair Endowment for hosting the IIT Roorkee researcher.

The authors wish to particularly acknowledge the discussions they had with Prof. G. K. Ananthasuresh of IISc Bengaluru, India toward the preparation of the SPARC proposal. The authors also wish to thank all the speakers, including Prof. Ananthasuresh, and the participants of the workshop on *Computational Intelligence and Its Applications* and the short course on *Pragmatic Optimization for Practical Problem Solving*, organized with SPARC funding support, at GNEC (IIT Roorkee campus), Greater Noida, India.

The authors wish to acknowledge the IIT Roorkee students, Drishti Bhasin, Sajag Swami, Sarthak Sharma, Saumya Sah and Jigyasa Kumari, for the extensive testing of *innovized progress operators* with different machine learning methods and the analysis of the trade-off between corresponding performance and run-time. Acknowledgments are also due to two of the MSU researchers, Anirudh Suresh, and Aryan Gondkar, for implementing the *pseudo-weight-based analysis* of the Pareto-optimal solutions, presented in one of the chapters of this book. The authors also acknowledge their current and past students for their dedication and hard work in executing prior foundational studies which lead to the ideas portrayed in this book.

The authors also thank Springer Nature Singapore Private Limited for publishing this book under the Genetic and Evolutionary Computation (GEC) series.

Finally, the authors express their sincere appreciation to their family members for their understanding and patience during extended periods of absence from home to make this collaboration fruitful. This book is dedicated to them.

Contents

Chapter 1
Introduction

1.1 Overview of Multi-and Many-objective Optimization

The formulation of an optimization problem, in generic terms, can be given by Eq. 1.1.

$$\begin{aligned}&\text{Minimize } F(X) = \{f_1(X), f_2(X), \ldots f_M(X)\}^T,\\&\text{where } X \equiv \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}.\end{aligned} \tag{1.1}$$

Here, M denotes the number of objectives; n denotes the number of decision variables; X represents a solution constituted by specific values of the n variables; $\mathcal{X} \subseteq \mathbb{R}^n$ represents the permissible variable (decision) space; and $\mathcal{Z} \equiv F(\mathcal{X}) \subseteq \mathbb{R}^M$ represents the permissible objective space. A given problem is referred to as a single-objective problem, an SOP, if $M = 1$; a multi-objective problem, an MOP, if $M = 2$ or 3; and a many-objective problem, an MaOP, if $M \geq 4$. The genesis for the distinction between MOPs and MaOPs is highlighted later in this section.

In the case of an SOP, a single solution can offer the best possible objective function value, and such a solution can be described as the global best solution or optimum. In the case of MOPs and MaOPs, unless all the objective functions are correlated (increase or decrease together), no single solution can offer the best value for all the objective functions simultaneously. However, a set of solutions can offer the best possible trade-offs among the objectives. Such a set, in which one objective cannot be improved without worsening at least one other objective, constitutes the Pareto-optimal set (PS), and the image of this set in the $\mathcal{Z}$ space is referred to as the Pareto-optimal front, or simply the Pareto front (PF). It is worth noting here that the multi-criterion decision-making (MCDM) literature refers the image of set $\mathcal{Z}$ as the *efficient* set.

Optimization refers to the task of finding the global best solution or the Pareto-optimal set, depending on the number of objectives involved. Research in the field of optimization started with the development of mathematical-programming-based algorithms [29] pertaining to SOPs. Such algorithms are predominantly *point*-based,

D. K. Saxena et al., *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization*, Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-2096-9_1

that is, they rely on the use of a single solution which is updated in each iteration, eventually leading to the global best solution. In particular, if MOPs are to be tackled using point-based algorithms, then multiple *scalarizations* [27] become essential. Here, scalarization implies the formulation of an SOP such that its optimum corresponds to one of the Pareto-optimal solutions of the MOP. In that situation, the use of different parameters for the scalarization produces different Pareto-optimal solutions for the MOP. However, the utility of this approach is marred by some challenges. In many real-world problems, the identification of scalarization parameters (which directly or indirectly represent preferences between different objectives) that are agreeable to a diverse set of stakeholders is quite a challenge. Furthermore, the identification of scalarization parameters that can lead to a uniformly distributed PF-approximation is a non-trivial task.

Plausibly, the above limitations of point-based algorithms motivated the development of several *metaheuristics* [34], among which Evolutionary Algorithms [19] have been quite popular, paving the way for evolutionary multi-objective optimization algorithms (EMOAs) [7, 9]. In general, EMOAs iteratively evolve a randomly initialized population (a set of solutions), using the principles of natural evolution, namely *variation* and *selection*, toward a good PF-approximation. In such algorithms, while variation seeks to create new solutions from the existing ones by imitating the natural capability of living beings via recombination and mutation, selection seeks to promote solutions with higher fitness by mimicking the competition for reproduction and resources among living beings. The fact that EMOAs are *population*-based makes them naturally suitable for solving MOPs, as a reasonable PF-approximation could be achieved by a single algorithmic run of an EMOA.

The Vector Evaluated Genetic Algorithm (VEGA) [32], proposed in the mid-1980s, is credited as the first EMOA. This was followed by the proposal of the key concept of *Pareto-dominance-based ranking*, to guide selection [19]. This paved the way for EMOAs such as MOGA [17], NSGA [33], and NPGA [21]. In the late 1990s, the importance of the notion of *elitism* (retaining the best solutions throughout the sequential generations of an EMOA) was recognized and SPEA [38] and NSGA-II [12] were proposed. After the predominance of Pareto-ranking-based EMOAs for more than a decade, two new philosophies for EMOAs were launched around the mid-2000s. One of them sought to incorporate a performance *indicator* into the selection mechanism [37], while the other relied on the use of *decomposition* [36] to transform an MOP into several SOPs, whose solutions approximate PF of the original problem. Several versions of indicator-based and decomposition-based EMOAs have been proposed since then. A wider coverage of EMOAs can be found in [6].

Another significant trend in the last two decades or so pertains to the development of evolutionary many-objective optimization algorithms (EMaOAs). Here, the term *many-objective* indicates problems with four or more objectives. The distinction between multi- and many-objective problems became imperative owing to the staggering revelations, between 2000 and 2005, that the performance of most well-known Pareto-dominance-/ranking-based EMOAs severely deteriorates when the number of objectives scales up beyond three. This was attributed to the fact that in such problems, a significant proportion of the entire population becomes non-dominated from the early generations [14]. Given this, the selection pressure for convergence ceases to exist (referred to as *dominance resistance*), and the density-based diversity preservation tends to worsen the performance by further diversifying poorly converged solutions (referred to as *active diversity promotion*) [28]. In a notable contribution, the authors in [16] argued that—when dealing with many-objective problems—Pareto-dominance is often inefficient in modeling and incorporating human decision-making (HDM) elements. In that situation, when comparing two solutions, Pareto-dominance fails to account for: (a) *number* of improved objectives, (b) *extent* of improvements, and (c) *relative preferences among the objectives*.

Recognizing the above challenges, three broad research directions emerged. The first approach, called the *objective reduction* approach [5, 22, 30], discriminates between the number of objectives in a given problem and the number of objectives that are essential to define the complete PF. The second approach seeks to counter the adverse impact of active diversity promotion, as in NSGA-II/DM1 [1] and SPEA2+SDE [25] by revising the density criterion; and NSGA-III [11, 23] by adopting a reference vector (RV)-based framework. The third approach seeks to counter the challenge of dominance resistance by inducing a preference-order over the non-dominated solutions. This includes the proposition of (nearly two dozen) dominance principles other than Pareto-dominance, such that the area dominated by a solution gets enlarged, and its chances of being non-dominated with another solution reduce. However, none of these alternative principles could overcome the fundamental challenges highlighted in [16]. An exception to this trend is the recently proposed *high-fidelity*-dominance (*hf*-dominance) [31], since it successfully incorporates all the HDM elements, explicitly and simultaneously, and without requiring tuning of any parameter. Notably, *hf*-dominance has been implemented in an RV-based framework, leading to an efficient EMâOA, namely LHFiD [31].

For the benefit of the readers, the terminology used for the various subdomains of evolutionary optimization and the corresponding algorithms is reiterated in Table 1.1:

- The terms EMO and EMaO are used discretionarily whenever it is necessary to distinguish between features of the underlying multi- and many-objective problems. When such a distinction is not necessary, the subdomains of evolutionary multi- and many-objective optimization are collectively referred to as EMâO.
- Since algorithms designed with the ability to handle many-objective problems are expected (by default) to handle multi-objective problems as well, the more generic term EMâOA has been used to denote an evolutionary multi- and many-objective optimization algorithm.

Table 1.1 Terminology of different subdomains of evolutionary optimization

Domain	Abbreviation for	
	Domain	Corresponding algorithm(s)
Evolutionary Multi-objective Optimization	EMO	EMOA(s)
Evolutionary Many-objective Optimization	EMaO	EMâOA(s)
Evolutionary Multi- and Many-objective Optimization	EMâO	

1.2 Overview of Machine Learning

Machine learning (ML) is a field under the umbrella of artificial intelligence, devoted to understanding and building methods that *learn*—that is, methods that leverage the available data to improve performance in a given set of tasks. At a higher level, ML methods build a model based on sample data, called a *training-dataset*, which is used to make *decisions* or *predictions* without explicitly programming them. Some well-known ML methods include artificial neural network (ANN) [20], random forest (RF) [4], k-nearest neighbors (kNN) [8], and principal component analysis (PCA) [24]. These methods are used in a wide variety of applications, including medicine, text mining, speech recognition, and computer vision.

Typically, the use of an ML model for any application consists of three steps: constructing a training-dataset, training the ML model on that dataset, and using the trained model to make decisions or predictions [15]. Most of these ML methods require some user-defined parameters pertaining to the model training, which directly affect the model's accuracy. For example, training an ANN requires several parameters to be defined, including the number of layers, the number of nodes in each layer, the activation function, the learning rate, and the loss function. Despite this, the quality of the training-dataset primarily affects the accuracy of the trained ML model.

ML methods can generally be divided into supervised and unsupervised learning [2]. The fundamental difference between the two is that in supervised learning, each sample in the training-dataset consists of an input and a desired output, whereas in unsupervised learning, each sample consists of only an input. While the former is suitable for tasks such as *classification* and *regression*, the latter is mainly used for tasks like *clustering*, as described below.

- **Classification:** When the desired output in each sample of the training-dataset is part of a set with a finite number of outputs, it is known as a classification task. An example could be the use of an ML method to determine whether a given solution is Pareto-optimal or not. Here, in each sample, the input would be the variable vector of a solution, while the output can have only two possible values, that is, 'Pareto-optimal' or 'Non-Pareto-optimal'.

- **Regression:** When the desired output in each sample of the training-dataset can be any numerical value, it is known as a regression task. An example could be the use of an ML method to determine the objective value of a given solution. Here, in each sample, the input would be the variable vector of a solution, while the output would be its corresponding objective value, which can possibly be any real number.
- **Clustering:** When the samples in the training-dataset need to be divided into different clusters, but the cluster properties are not known beforehand, the task is called clustering. An example could be the use of an ML method to detect outliers in a given dataset. Here, the ML method would group the meaningful samples together and the ungrouped samples would be marked as outliers. Such an approach is often used to reduce noise in the ML training-dataset.

It is critical to note that some ML methods, including RF and *k*NN, have classification and regression variants [26, 35]. Therefore, depending on the need, these ML methods can be used for either of the tasks.

1.3 Scope of the Book

This book reinforces the fact that the availability of multiple sets of solutions over successive generations of EMâOAs makes them amenable to the application of ML, for different pursuits. To this end, this book attempts to present and discuss several instances of ML-assisted EMâO. The structure of the remaining book is highlighted below.

Chapter 2 begins by highlighting some of the practical problem domains in which the role of optimization is critical. Then the different problem types (uni-modal SOPs, multi-modal SOPs, MOPs, and MaOPs) and different algorithm classes (point-based and population-based) are discussed. Here, the suitability of different algorithm classes for different problem types is also highlighted through examples. From a realistic point of view, this chapter concludes by discussing *no-free-lunch (NFL) theorem*, asserting that no single algorithm could be the best for solving all classes of optimization problems. That is, the superior performance of some algorithms for specific problem classes shall be accompanied by their inferior performance on some other problem classes. This calls for customized optimization algorithms for routine applications to specific problem classes.

Chapter 3 highlights foundational studies on ML-based enhancements in the domain of evolutionary multi-objective optimization. These studies relate to the notion of *innovization* and *online innovization* leading up to innovized repair of offspring populations; construction of appropriate surrogate models; and design of efficient mutation operators. It may be noted that the notion of online innovization—learning effective features from the trade-off solutions across the intermediate generations of an EMOA run, and propagating these features in subsequent generations—forms the basis for Chaps. 5–7.

Chapter 4 focuses on *learning* to understand the optimization problem structure. To this end, an ML-based *objective reduction* approach is discussed. It enables the identification of redundant objectives (not essential to define the true *PF*) and preference ranking of the essential objectives. Such knowledge discovery may facilitate a better understanding of the physics of the problem, in addition to reducing its complexity and promising higher search efficiency for EMâOAs. This knowledge is shown to offer decision support by revealing: (i) the smallest subset of objectives, ensuring that the error associated with the omission of the remaining objectives does not exceed a user-defined δ, and (ii) the subset of k objectives (for a user-defined k) ensuring that the error associated with the omission of the remaining objectives is the minimum, compared to any other possible combination of k objectives. The implementation and benefits of this approach are demonstrated in several real-world and test problems.

Chapters 5–7 present the authors' most recent and more direct efforts toward ML-based performance enhancement for RV-based EMâOAs, abbreviated as RV-EMâOAs. Although future efforts will seek to demonstrate the applicability and efficacy of such enhancements for EMâOAs of arbitrary form, RV-EMâOAs have been chosen initially, since they offer better tractability of solutions along and across the RVs. The hallmark of the proposed enhancements is that they are guided by the overarching considerations of convergence–diversity balance, ML-based risk–reward trade-off, and avoidance of extra solution evaluations. Thus, these chapters could provide a template for further research in this direction. After this background, the individual contributions of these three chapters are as highlighted below.

- Chapter 5 focuses on *learning* to better converge, toward which the *Innovized Progress 2 (IP2)* operator is discussed. In any intermediate generation of an RV-EMâOA run, the IP2 operator relies on the use of an ML method to learn meaningful search directions in the $\mathcal{X}$ space, based on a suitable mapping of *inter-generational* solutions for each RV, which implies a mapping of previous generation solutions (in the $\mathcal{X}$ space) to the current generation solution (in the $\mathcal{X}$ space) for each RV. The learned ML model is then used to create pro-convergence offspring solutions, eventually improving the quality and/or rate of convergence to *PF*. The utility of the IP2 operator is demonstrated through proof-of-concept results.
- Chapter 6 focuses on *learning* to better diversify, toward which the *Innovized Progress 3 (IP3)* operator is discussed. In any intermediate generation of an RV-EMâOA run, the IP3 operator relies on the use of an ML method to learn meaningful search directions in $\mathcal{X}$ space, based on a suitable mapping of *intra-generational* solutions, across different RVs. It implies a judicious mapping of the same (current) generation solutions that are associated with different RVs. The learned ML model is then used to create pro-diversity offspring solutions, eventually improving the diversity (both coverage and uniformity) of solutions in the obtained *PF*-approximation. The utility of the IP3 operator is demonstrated through proof-of-concept results.
- Chapter 7 focuses on *learning* to simultaneously converge and diversify better, toward which the *Unified Innovized Progress (UIP)* operator is discussed. The UIP

operator integrates the capabilities of the pro-convergence IP2 and pro-diversity IP3 operators, in a manner that is *generic*—applicable to different RV-EMâOAs; and *practicable*—not requiring any extra solution evaluations over the base RV-EMâOA. The utility of the UIP operator is demonstrated on a wide range of test MOPs and MaOPs.

Chapter 8 focuses on investigating the sensitivity of the performance of *innovized progress* operators (IP2, IP3, and UIP), vis-à-vis the underlying ML methods. It covers a comprehensive set of eight ML methods from different classes, including linear methods, trees, boosting algorithms, and nonlinear methods. The performance sensitivity, in terms of the solution's quality and computational run-time, has been analyzed across a wide range of MOPs and MaOPs.

Chapter 9 introduces an approach based on post-optimality analysis that can facilitate contrasting features as per the requirement—a more uniform distribution of solutions across PF, and also a biased distribution across PF (higher concentration in specific parts) as may be desired by the decision maker. Given a PF approximation by an EMâOA, this approach relies on training an ML model to capture the relationship between pseudo-weight vectors derived from objective vectors in the PF-approximation (F in $\mathcal{Z}$), and their underlying variable vectors (X in $\mathcal{X}$). Subsequently, the trained ML model is used to create new non-dominated solutions in any desired region of the obtained PF-approximation. The results of some test problems are presented to demonstrate the utility of this approach. This chapter is adapted from a recent paper by Deb and his students [10].

Chapter 10 contains concluding remarks about this book. Notably, the seeds of ML-assisted EMâO had been laid by two of the authors more than 15 years ago through an ML-based objective reduction study [13] in 2006. This was further propagated over the timeline through automated innovization [3] starting in 2010 and online innovization [18] starting in 2017. However, the authors' motivation for more direct and dedicated efforts for ML-based enhancements in EMâO was stimulated by the SPARC project. The authors believe that the success of chronological advances portrayed in this book is inspiring and marks only the tip of the iceberg. The authors are confident that the template laid down in this book, structured around the key considerations of convergence–diversity balance, ML-based risk–reward trade-off, and avoidance of extra solution evaluations, will guide further research in this direction. Toward it, Chap. 10 also lists a number of potential future research directions pertaining to ML-assisted EMâO, which must be taken up by EMO-ML researchers to unveil a comprehensive potential of ML methods in improving EMâO's performance.

The authors hope that this book will stimulate greater interest among EMâO researchers and practitioners to exploit the natural synergy that exists with the ML domain, toward solving challenging real-world problems computationally quickly and more efficiently.

References

1. Adra, S.F., Fleming, P.J.: Diversity management in evolutionary many-objective optimization. IEEE Trans. Evol. Comput. **15**(2), 183–195 (2011). https://doi.org/10.1109/TEVC.2010.2058117
2. Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., Aljaaf, A.J.: A systematic review on supervised and unsupervised machine learning algorithms for data science. In: Supervised and Unsupervised Learning for Data Science, pp. 3–21. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-22475-2_1
3. Bandaru, S., Deb, K.: Automated discovery of vital knowledge from Pareto-optimal solutions: first results from engineering design. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010). https://doi.org/10.1109/CEC.2010.5586501
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
5. Brockhoff, D., Zitzler, E.: Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In: Runarsson, T.P., Beyer, H.G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) Parallel Problem Solving from Nature - PPSN IX, pp. 533–542. Springer, Berlin Heidelberg, Berlin, Heidelberg (2006)
6. Coello, C.A.C., Brambila, S.G., Gamboa, J.F., Tapia, M.G.C., Gómez, R.H.: Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. Complex & Intell. Syst. **6**, 221–236 (2020). https://doi.org/10.1007/s40747-019-0113-4
7. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-objective Problems. Springer, New York (2007)
8. Cover, T.: Estimation by the nearest neighbor rule. IEEE Trans. Inf. Theory **14**(1), 50–55 (1968). https://doi.org/10.1109/TIT.1968.1054098
9. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
10. Deb, K., Gondkar, A., Anirudh, S.: Learning to predict Pareto-optimal solutions from pseudo-weights. In: Emmerich, M., Deutz, A., Wang, H., Kononova, A.V., Naujoks, B., Li, K., Miettinen, K., Yevseyeva, I. (eds.) Evolutionary Multi-Criterion Optimization, pp. 191–204. Springer Nature Switzerland, Cham (2023)
11. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
13. Deb, K., Saxena, D.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of the World Congress on Computational Intelligence (WCCI-2006), pp. 3352–3360 (2006)
14. Deb, K., Saxena, D.K.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: IEEE Congress on Evolutionary Computation, pp. 3353–3360 (2006)
15. Deisenroth, M.P., Faisal, A.A., Ong, C.S.: In: Mathematics for Machine Learning. Cambridge University Press (2020). https://doi.org/10.1017/9781108679930
16. Farina, M., Amato, P.: A fuzzy definition of "optimality" for many-criteria optimization problems. IEEE Trans. Syst. Man Cybern. - Part A: Syst. Humans **34**(3), 315–326 (2004). https://doi.org/10.1109/TSMCA.2004.824873
17. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In: S. Forest (ed.) Fifth International Conference on Genetic Algorithms, pp. 416–423 (1993)

18. Gaur, A., Deb, K.: Effect of size and order of variables in rules for multi-objective repair-based *innovization* procedure. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2177–2184 (2017). https://doi.org/10.1109/CEC.2017.7969568
19. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, USA (1989)
20. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge (1995)
21. Horn, J., Nafpliotis, N., Goldberg, D.: A niched Pareto genetic algorithm for multiobjective optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, vol. 1, pp. 82–87 (1994). https://doi.org/10.1109/ICEC.1994.350037
22. Jaimes, A.L., Coello, C.A.C., Chakraborty, D.: Objective reduction using a feature selection technique. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 673–680 (2008)
23. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. IEEE Trans. Evol. Comput. **18**(4), 602–622 (2014). https://doi.org/10.1109/TEVC.2013.2281534
24. Jolliffe, I.T.: Principal Component Analysis. Springer (2002)
25. Li, M., Yang, S., Liu, X.: Shift-based density estimation for Pareto-based algorithms in many-objective optimization. IEEE Trans. Evol. Comput. **18**(3), 348–365 (2014)
26. Liaw, A., Wiener, M.: Classification and regression by randomforest. Forest **23** (2001)
27. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
28. Purshouse, R.C., Fleming, P.J.: On the evolutionary optimization of many conflicting objectives. IEEE Trans. Evol. Comput. **11**(6), 770–784 (2007)
29. Rao, S.S.: Engineering Optimization Theory and Practice. Wiley, USA (2019)
30. Saxena, D., Deb, K.: Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science, vol. 4403, pp. 772–787. Springer, Berlin/Heidelberg (2007)
31. Saxena, D.K., Mittal, S., Kapoor, S., Deb, K.: A localized high-fidelity-dominance based many-objective evolutionary algorithm. IEEE Trans. Evol. Comput. 1–1 (2022). https://doi.org/10.1109/TEVC.2022.3188064
32. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. Ph.D. thesis, Vanderbilt University (1984)
33. Srinivas, N., Deb, K.: Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. **2**(3), 221–248 (1994). https://doi.org/10.1162/evco.1994.2.3.221
34. Talbi, E.: Metaheuristics: From Design to Implementation. Wiley (2009)
35. Taunk, K., De, S., Verma, S., Swetapadma, A.: A brief review of nearest neighbor algorithm for learning and classification. In: 2019 International Conference on Intelligent Computing and Control Systems (ICCS), pp. 1255–1260 (2019). https://doi.org/10.1109/ICCS45141.2019.9065747
36. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007). https://doi.org/10.1109/TEVC.2007.892759
37. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., Burke, E., Lozano, J., Smith, J., Merelo-Guervós, J., Bullinaria, J., Rowe, J., Tino, P., Kabán, A., Schwefel, H.P. (eds.) Parallel Problem Solving from Nature - PPSN VIII. Lecture Notes in Computer Science, vol. 3242, pp. 832–842. Springer, Berlin/Heidelberg (2004)
38. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999). https://doi.org/10.1109/4235.797969

Chapter 2
Optimization Problems and Algorithms

This chapter starts by highlighting some domains of practical problems where optimization is or can be commonly applied. Then, the focus is shifted to different problem classes based on the number of objectives, and also on the popular point- and population-based optimization algorithms. Finally, to contextualize the suitability of different optimization algorithms for different problem types, the No-free-lunch (NFL) theorem is discussed.

2.1 Optimization Problems

Optimization problems are omnipresent. In the following, we list a few problem domains that optimization algorithms are routinely used to solve.

- Design and manufacturing.
- Modeling.
- Prediction.
- Inverse problems.
- Design of experiments.
- Scheduling and resource allocation problems.
- Control systems.
- Machine learning problems.

The lion's share of optimization problems come from the domain of design and manufacturing. In design-related problems, the key design parameters are usually considered as variables that can be changed to create new designs; the key performance indicators (KPIs) of the design are used as objectives; and those KPIs which must be strictly met for the design to be acceptable are used as constraints. In manufacturing-related problems, the key parameters of the manufacturing process are considered as variables; the qualities of the final product are used as objectives; and

D. K. Saxena et al., *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization*, Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-2096-9_2

the resource restrictions and/or certain process-related KPIs are used as constraints. Often, these problems require a time-dependent simulation process to acquire the objective and constraint values.

Modeling a system or process is important to understand its dynamics. The models can be derived either from empirical data or from more fundamental relationships (first principles, physics-based) that rely on knowledge of the system/process. In practice, a combination of both approaches is often used. In that, the forms of the equations are developed from the fundamental principles, while the unknown or uncertain parameters are adjusted to fit the empirical data. This fitting entails optimization, where a common objective is to minimize the sum of squared error, which penalizes deviation of the fundamental model from the data. Here, constraints may ensure satisfaction of the important or essential characteristics of the system. In this context, the task of finding regression models can be viewed as an optimization task. Alternatively, modeling can also be achieved using neural networks, where a system's input–output relationships are available. This, too, involves optimization. In that, the network's architecture and the connection weights can be considered as variables; restrictions on the complexity of the architecture and weight values can serve as constraints; and the objective can pertain to minimization of back-propagation error (error between the network's and true system's output). Furthermore, the modeling of time-series data may eventually lead to the prediction of critical quantities of the system. From past data, the error between predicted and true values of critical quantities can be minimized to make a better prediction model.

Many practical problems demand that inputs be found for desired outputs, while the forward process of achieving output from input is known. For example, in material discovery problems, the manufacturing process parameters (inputs) should be known to achieve certain desired material properties (outputs). Such inverse problems are difficult to solve for multiple reasons, and optimization is one way to achieve this task. In these problems, the inputs serve as variables, and the objective relates to the minimization of the error between the predicted output (obtained by the forward computation or simulation based on the input) and the desired output. Depending on the problem, simplicity or the principle of *Occam's razor* is used as a constraint, so that the resulting input solution is not too complex.

In many expensive and time-consuming processes for data acquisition, a method called *design of experiments* is often used. Since it is not possible to evaluate all possible feature combinations, an optimization methodology can provide information on the next most appropriate set of experimental setups, given the experiments already performed. The variables in such an optimization problem would be a sequence of combinations of input features for the next set of experiments, and the objective function could be to maximize an 'information gain' estimated from the execution of the experiments.

Scheduling and resource allocation problems are an important class of problems related to efficient resource management, where optimization plays a key role. In such problems, resources and/or schedules serve as variables; the tardiness, cost, throughput, or other performance-related indicators are pursued as objectives; and certain resource limits are portrayed as constraints. Such problems are often formulated as linear programming problems, for which structured optimization algorithms in the operations research domain are available. However, these problems could be quite challenging, particularly when they are large-scale or when their solutions are desired in real-time.

Optimal control problems require identifying time-varying or spatial variations of control parameters so that the underlying system becomes stable as quickly as possible. In these problems, variables are usually temporal or spatial functions, rather than fixed values, and a simulation process is needed to construct the objective and constraint values.

Finally, most machine learning methods use an optimization algorithm at the core. Most often, machine learning researchers resort to the use of a standard *hill-climbing* method, despite the fact that it works only if there is a single peak—no local peaks, no plateaus, and no ridges. Hence, a more efficient approach could be to use other local search algorithms such as stochastic hill climbing, random walks, and simulated annealing or more sophisticated nonlinear optimization algorithms toward finding the true optimal solution. The discussions in this chapter may produce interesting future studies to settle the pros and cons of such approaches.

It is clear from the above discussions that optimization lies at the core of practical problem solving, across multiple important domains. When formulating an optimization problem, the following four ingredients must be identified:

1. **Parameters**: The problem parameters refer to all those quantities whose values are *fixed* before starting the optimization process.
2. **Variables**: The variables refer to all those quantities whose values are *varied* during the optimization process. These variables, also known as design/decision variables, are denoted by a n-dimensional vector X (where n is the number of variables). Since each $x_i \in X$ varies during optimization, its range can be specified by its lower bound $x_i^{(L)}$ and its upper bound $x_i^{(U)}$. If such bounds are available for all the variables, they constitute a hyperbox within which the search is restricted. However, if such bounds are not available and the *feasible* search space (defined below) is not bounded, then the optimization process may not terminate in a finite number of iterations. Since the search takes place in an n-dimensional space, efforts to reduce the number of variables are always helpful for an optimization task.
3. **Constraints**: The constraints refer to the restrictions on the search space and influence whether any given solution is permissible or not. There are two types of constraints. Each inequality constraint ($g_j(X) \leq 0$, for $j = 1, \ldots, J$) divides the search space into a feasible and an infeasible region. In doing so, each inequality constraint reduces the permissible search space to a subset of the original space. Each equality constraint ($h_k(X) = 0$, for $k = 1, \ldots, K$) imposes stricter restrictions, as only the solutions that belong to the constraint boundary are acceptable. In doing so, each equality constraint reduces the dimensionality of the search space. All the equality and inequality constraints together constitute the *feasible* search space $\mathcal{X}$ (used in Eq. 1.1), which could be defined as follows:

$$\mathcal{X} = \left\{ X \middle| \left(g_j(X) \leq 0,\ \forall j \right) \wedge \left(h_k(X) = 0,\ \forall k \right) \wedge \left(x_i^{(L)} \leq x_i \leq x_i^{(U)},\ \forall i \right) \right\}. \tag{2.1}$$

It is needless to assert that the optimal solution to the problem must satisfy the above equation. Also, in certain problems, some constraints can be declared *soft*, which means that a small violation (within ϵ_j) of these constraints is allowed, that is, $g_j(X) \leq -\epsilon_j$. Constraints that cannot be violated are known as *hard* constraints.

4. **Objectives**: The objectives refer to the performance metrics that must be minimized or maximized, and their set is denoted as F (Eq. 1.1). While the categorization of problems into single-, multi-, and many-objective problems has been introduced in Chap. 1 alongside their salient features, this chapter discusses them in more detail below.

2.1.1 Single-objective Optimization Problems

Traditionally, most optimization problems were posed as single-objective problems (SOPs), as given by Eq. 2.2.

$$\begin{array}{lll} \text{Minimize} & f(X), & \\ \text{subject to} & g_j(X) \leq 0, & j = 1, 2, \ldots, J; \\ & h_k(X) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array} \tag{2.2}$$

The variables in the above formulation are denoted by the vector X. The parameters within the description of objective and constraint functions and the variable bounds ($x_i^{(L)}$ and $x_i^{(U)}$) are parameters of the problem and are kept fixed during an optimization run. A maximization problem, in which the objective function $f(X)$ is to be maximized, can be aligned with the above formulation, by pre-multiplying the objective by -1. In other words, $Max.\ f(X)$ is equivalent to $Min.\ -f(X)$ in terms of their optimal solutions, irrespective of whether the problem is constrained or unconstrained. Similarly, if a constraint is given in the form $g_j(X) \geq 0$, it can be aligned with the above as $-g_j(X) \leq 0$. In some problems, a strict inequality constraint $g_j(X) < 0$ means that the feasible search space does not contain the boundary $g_j(X) = 0$. Although theoretically an optimal solution in the limit can be defined, practically (while working with finite-precision computing platforms) the optimal solution can only be found approximately for such constraints. For such reasons, strict inequality constraints have not been considered in this book.

A problem modeled as above may have one or more optimal solution(s) X^*. The role of an optimization algorithm (discussed in the next section) is to search the n-dimensional space and find the exact optimal solution(s) or near-optimal solution(s) with as few *solution evaluations* as possible. A solution evaluation is a complete evaluation (objective and constraint functions) of a solution X. For practical problems, the evaluation of one solution can be computationally expensive, taking a few minutes or even several hours.

For an illustration of some key concepts, consider the problem defined by Eq. 2.3. Its two-variable search space is depicted in Fig. 2.1a. In that, the feasible region is bounded by the curve (nonlinear constraint) and the straight line (linear constraint). Furthermore, the objective contours shown in this figure highlight the fact that it is a uni-modal problem, where both constraints are *inactive* at the optimal solution, implying that the same optimal solution will result even if the constraints were not considered.

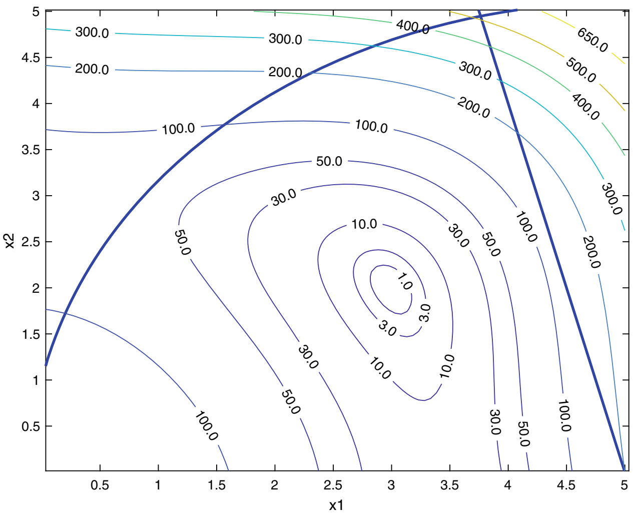
$$\begin{aligned} \text{Minimize } & f(x_1, x_2) = (x_1^2 - x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2, \\ \text{subject to } & g_1(x_1, x_2) = (x_1 - 5)^2 + x_2^2 - 26 \le 0, \\ & g_2(x_1, x_2) = 4x_1 + x_2 - 20 \le 0, \\ & 0 \le x_1 \le 5, \quad 0 \le x_2 \le 5. \end{aligned} \tag{2.3}$$

Now consider a variant of the above problem, where f, g_1, and g_2 remain the same, but the new variable bounds are given by $-5 \le x_1 \le 5$ and $-5 \le x_2 \le 5$. Figure 2.1b highlights the fact that with a change in the variable bounds, the problem becomes a multi-modal problem, with four optimal solutions, of which only two are feasible. With regard to modality, the following are not clearly defined in the literature but may be noted:

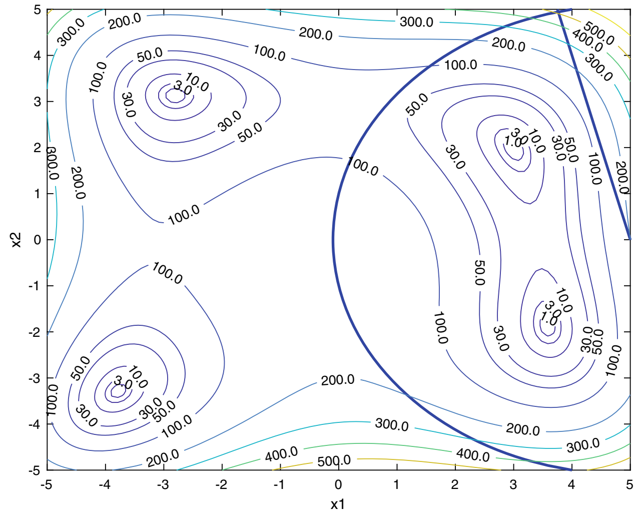
- In a multi-modal problem, not all optimal solutions may have the same objective values. Those offering the best objective value are called *global* optimal solutions, while those offering inferior objective value(s) are called *local* optimal solutions. If the user is interested in finding multiple optima, then the task is referred to as *multi-modal optimization*.
- Even for a multi-modal problem, if the user wants to find only a single optimum (global optimum or one of the global optima, as the case may be), then the task is referred to as *uni-modal optimization* or *global optimization*.
- Optimization algorithms aiming at multi-modal optimization need to have inbuilt mechanisms that can drive the search to multiple optimal solutions. In other words, the design of optimization algorithms for multi-modal optimization ought to be different from those designed for uni-modal optimization. Hence, the choice of algorithm by a user should depend on the user's goal, despite addressing the same problem.

It is implicit in the problem definition itself that every optimal solution X^* must be feasible, that is, it should satisfy all the constraints, including the variable bounds. Furthermore, X^* must satisfy several *optimality* and *constraint qualification* conditions, a discussion of which is beyond the scope of this book. Interested readers may refer to the theoretical optimization literature [1, 20].

It may be noted that the generic formulation given by Eq. 2.3 may lead to more structured problems [38] under specific conditions, as highlighted below:



(a) Uni-modal problem



(b) Multi-modal problem

Fig. 2.1 Feasible search space with single and multiple optimal solutions for a two-variable and two-constraint problem. The region between the circle and straight line is feasible. Contours of the objective function are shown

- If the objective function is a convex function and the feasible search space constitutes a convex region, then the resulting problem constitutes a *convex programming problem* (CPP). For a CPP, every local optimum is also a global optimum. Hence, a CPP may have one or more global optimal solutions.
- If the objective and constraints are all linear functions of the variables, the resulting problem is called a *linear programming problem* (LPP). LPPs are a special case of CPPs, and hence, they may have one or more global optimal solutions.
- If the objective function is quadratic and all the constraints are linear functions of the variables, the resulting problem is called a *quadratic programming problem* (QPP). For a bounded feasible space, a QPP always has a single optimal solution.

If the objective function and/or any of the constraint functions do not meet the above conditions, the resulting problem is called a *nonlinear programming* (NLP) problem. There does not exist, and arguably there cannot exist due to the NFL theorem [51], a single optimization algorithm that is most efficient in finding the optimal solution(s) for all NLP problems (Sect. 2.2.3). However, it may be possible to develop an efficient optimization algorithm for a specific class of problems.

2.1.2 Multi-objective Optimization Problems

A multi-objective optimization problem (MOP), as given by Eq. 2.4, involves a number of conflicting objective functions (meaning that optima of all objectives are not identical and produce trade-offs among objectives) that are to be minimized or maximized subject to a number of constraints and variable bounds. Note that despite objectives being non-conflicting, certain constraints can produce trade-off optimal solutions, thereby leading to a multi-objective optimization problem.

$$\begin{array}{rlr} \text{Minimize} & F(X) = \{f_1(X), f_2(X), \ldots f_M(X)\}^T, & \\ \text{subject to} & g_j(X) \le 0, \qquad j = 1, 2, \ldots, J, & \\ & h_k(X) = 0, \qquad k = 1, 2, \ldots, K, & \\ & x_i^{(L)} \le x_i \le x_i^{(U)}, \; i = 1, 2, \ldots, n. & \end{array} \tag{2.4}$$

The functions to maximize can be converted to minimization functions by multiplying by -1. For any feasible solution X in $\mathcal{X}$ (satisfies Eq. 2.1), there exists its image, given by $\mathbf{z} = F(X) = (f_1, f_2, \ldots, f_M)^T \in \mathcal{Z}$ (the set of all feasible objective values). Optimal solutions in the context of MOPs can be defined through a mathematical concept of *partial ordering* [43], implemented through the *Pareto-dominance* principle [8, 34], as highlighted below.

Definition 2.1 A solution $X^{(1)}$ is said to Pareto-dominate another solution $X^{(2)}$, if *both* the conditions are true:

1. The solution $X^{(1)}$ is not worse than $X^{(2)}$ in all objectives, that is, no component of $\mathbf{z}^{(1)}$ is worse than the corresponding component of $\mathbf{z}^{(2)}$.

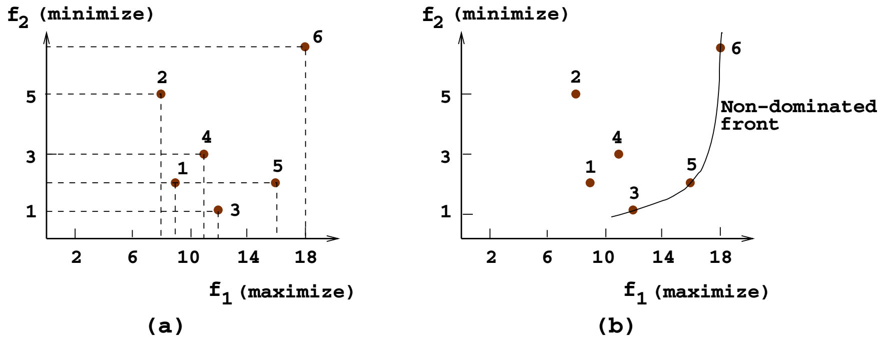


Fig. 2.2 A set of solutions and the first non-dominated front are shown in the objective space (taken from [8] and edited)

2. The solution $X^{(1)}$ is strictly better than $X^{(2)}$ in at least one objective, that is, at least one component of $\mathbf{z}^{(1)}$ is strictly better than the corresponding component of $\mathbf{z}^{(2)}$.

To appreciate how the dominance principle helps to define optimal solutions in the context of MOPs, consider Fig. 2.2, pertaining to a two-objective problem, where f_1 is to be maximized and f_2 is to be minimized. It represents the objective vectors ($\mathcal{Z}$ vectors) of six solutions $X^{(1)}$ to $X^{(6)}$, given by $\mathbf{z}^{(1)}$ to $\mathbf{z}^{(6)}$, respectively. However, these $\mathcal{Z}$ vectors have been referred to as 1 to 6 to avoid clutter in the figure. To help visualize the application of the dominance principle in all solutions, Fig. 2.2a highlights the individual objective components of each $\mathcal{Z}$ vector by dashed lines. In that, the larger the f_1 component and the smaller the f_2 component, the better. In this context, 3 can be said to dominate 1, since: (i) 3 is not worse than 1 in any of the two objective components, and (ii) 3 is strictly better than 1 in at least one objective component (in fact, 3 is strictly better than 1 in both objective components). A similar dominance check can be applied to every solution with respect to every other solution. Finally, solutions such as 3, 5, and 6 that are not dominated by any other solution are called *non-dominated* or *efficient* solutions and together constitute what is referred to as the *Pareto front* (*PF*). This front is characterized by an interesting *trade-off* property, in that, a traversal from one constituent solution to another is marked by a gain in some objective(s) and a loss in some other objective(s). Notably, the solutions in $\mathcal{X}$ space underlying the solutions in $\mathcal{Z}$ space are optimal solutions to an MOP and are said to constitute the Pareto-optimal set (*PS*). There are more mathematically elegant definitions of Pareto-optimality, which the interested readers may refer to [29, 34]. The computational complexity associated with the determination of non-dominated solutions in a given set of N solutions is $\mathcal{O}(N \log N)$ for $M = 2$ and 3. However, for $M > 3$, the complexity increases to $\mathcal{O}(N \log^{M-2} N)$ [33], implying that the procedure is not so efficient even for a moderately high number of objectives.

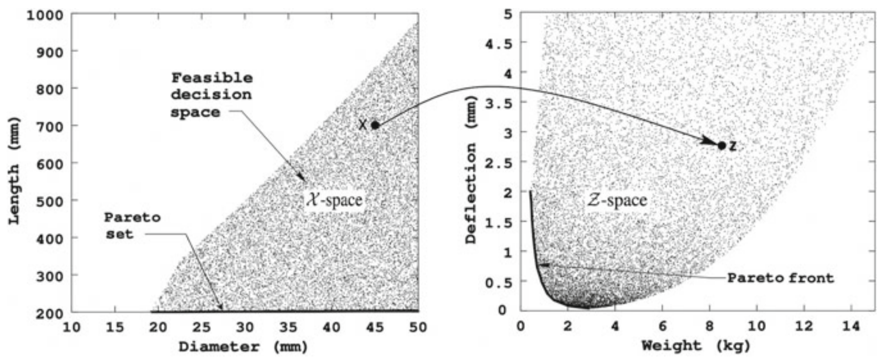


Fig. 2.3 Cantilever beam design problem: depicting the feasible variable space ($\mathcal{X}$) and the corresponding feasible objective space ($\mathcal{Z}$); Pareto set and Pareto front (taken from [8] and edited)

To familiarize the readers with the key terminology introduced so far, Equation 2.5 gives a two-objective, two-variable (d, l), and two-constraint cantilever beam design optimization problem, originally introduced in [8].

$$\begin{array}{ll}\text{Minimize} & f_1(d,l) = \rho\frac{\pi d^2}{4} l,\\ \text{Minimize} & f_2(d,l) = \delta = \dfrac{64Pl^3}{3E\pi d^4},\\ \text{subject to} & g_1(d,l) = \frac{32Pl}{\pi d^3} - S_y \le 0,\\ & g_2(d,l) = \delta - \delta_{\max} \le 0,\\ & 10 \le d \le 50, \quad 200 \le l \le 1000.\\ \text{with parameters} & \rho = 7800\ \text{kg/m}^3,\ P = 1\ \text{kN},\ E = 207\ \text{GPa},\\ & S_y = 300\ \text{MPa}, \quad \delta_{\max} = 5\ \text{mm}.\end{array} \tag{2.5}$$

The plot on the left in Fig. 2.3 shows that the feasible decision space is only a subset of the overall decision space ($\mathcal{X}$ space) bounded by $10 \le d \le 50$ and $200 \le l \le 1000$. Furthermore, the figure depicts the mapping of a feasible solution $X \in \mathcal{X}$ to a feasible solution $\mathbf{z} \in \mathcal{Z}$; the PF in $\mathcal{Z}$ space; and the PS in $\mathcal{X}$ space which corresponds to $l = 200$, $18.94 \le d \le 50$.

2.1.3 Many-objective Optimization Problems

Many-objective optimization problems (MaOPs) refer to problems with more than three objectives. Their formulation remains the same as that of MOPs (Eq. 2.4), just that $M \ge 3$. As highlighted in Chap. 1, this distinction in terminology became imperative between 2000–2005, considering that some of the most well-known Pareto-dominance/ranking-based evolutionary multi-objective optimization algorithms (EMOAs) reportedly showed poor scalability with the number of objectives.

With each increase in M, MaOPs pose additional challenges with regard to: (i) search efficiency, (ii) the need for an exponentially increasing population size and number of scalarizations, for EMOAs and point-based algorithms, respectively, and (iii) difficulty in visualization (necessitating Parallel coordinate plots, Radial visualization, etc. [34, 48]).

2.2 Optimization Algorithms

Optimization algorithms can be broadly classified into point-based and population-based algorithms, which are discussed next.

2.2.1 Point-Based Optimization Algorithms

Point-based algorithms start their search with a single solution X_t (often a randomly chosen solution) and create a single new solution Y_t using an update process: $Y_t = \mathcal{N}(X_t)$. At first, the iteration counter is $t = 0$. If Y_t is *better* than X_t, then Y_t is accepted and used to define the starting solution for the next iteration, that is, $X_{t+1} = Y_t$, otherwise $X_{t+1} = X_t$. If a termination condition is not satisfied, the iteration counter t is incremented by one and the above procedure is repeated. The superiority of a solution is determined by a composite function $\mathcal{L}(X)$, which combines objective and constraint function values in a way to prefer smaller $f(X)$ values for minimization problems and smaller overall constraint violations. A pseudo-code for the same is presented in Algorithm 2.1.

Algorithm 2.1: `Point_Optimization`($\mathcal{N}$, $X^{(0)}$, $t_{\max}$)

```
Input: A composite function $\mathcal{L}(X)$, Neighborhood operator $\mathcal{N}$, initial point $X^{(0)}$,
       maximum iterations $t_{\max}$
Output: Optimized solution $X^{(t_{\max})}$
1 $t \leftarrow 0$
2 for $t$ = 0 to ($t_{\max} - 1$) do
3     $Y_t = \mathcal{N}(X_t)$
4     if $\mathcal{L}(Y_t) < \mathcal{L}(X_t)$ then
5         $X_{t+1} \leftarrow Y_t$
6     else
7         $X_{t+1} \leftarrow X_t$
8     $t \leftarrow t + 1$
```

The main crux of an optimization algorithm is to design an appropriate update procedure $\mathcal{N}(X_t)$. It can simply be a neighborhood operator in which the new point can be created using a Gaussian probability distribution around X_t with a predefined fixed or adaptive covariance matrix [25]. In gradient-based optimization algorithms, the new solution is created along the negative of the gradient vector $\nabla f(X_t)$ [6]. In the presence of constraints, more sophisticated direct or gradient-based update methods of $\mathcal{L}$ are used [6, 39].

In addition to the choice of update procedure $\mathcal{N}(X_t)$, comparison of solutions X_t and X_{t+1} is another important aspect. It depends on whether the problem being solved is constrained or not, and whether it involves a single objective or multiple objectives, as discussed below.

2.2.1.1 Point-Based Algorithms for Single-objective Problems

For unconstrained SOPs without constraints, the comparison of X_t and X_{t+1} can simply be made based on the objective function value ($\mathcal{L}(X) = f(X)$). However, for constrained SOPs, a hierarchical comparison is proposed that first checks the feasibility of solutions and then the objective values [7]. In more sophisticated algorithms, the update operator itself factors in the constraints to ensure that the newly created X_{t+1} is always better, in the combined sense of objective and constraints.

For illustration, the constrained problem introduced earlier (Eq. 2.3) is considered here, and the search trajectory or the progress path of a point-based algorithm (`fmincon()` MATLAB software), starting with the solution $X^{(0)} = (0, 5)^T$, leading to the optimal solution $X^* = (3, 2)^T$ is shown in Fig. 2.4. Intermittent solutions are marked by blue circles.

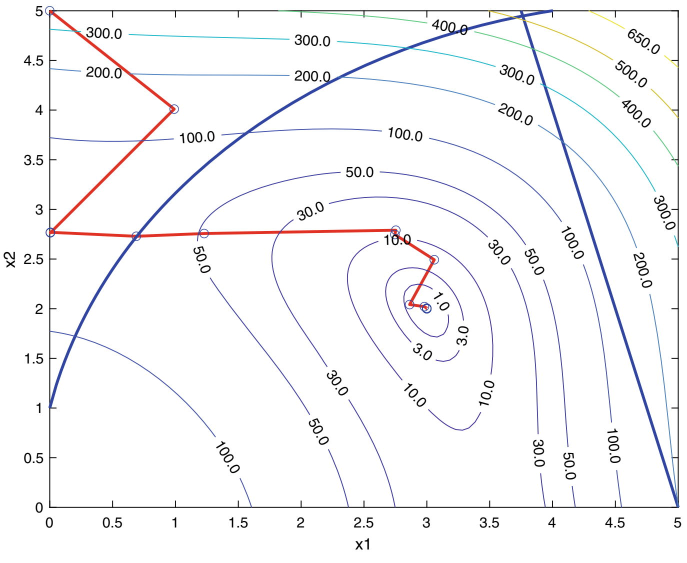


Fig. 2.4 Iteration-wise progress of a point-based optimization algorithm toward the optimum

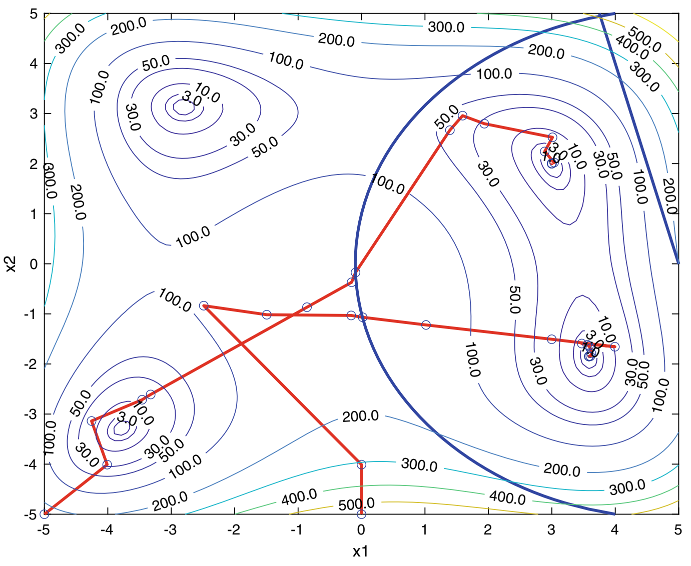


Fig. 2.5 Iteration-wise progress of a point-based optimization algorithm toward two optimal solutions starting with different initial solutions

To further illustrate the performance of `fmincon()` on an SOP with multi-modality, the earlier introduced variant of Problem 2.3 (with $-5 \leq x_1 \leq 5$ and $-5 \leq x_2 \leq 5$) is also considered here. Figure 2.5 reveals that the choice of $X^{(0)} = (-5, -5)^T$ leads to one of the two (feasible) optimal solutions. Since the goal is to find another optimal solution, `fmincon()` is invoked again but with a significantly different $X^{(0)} = (0, -5)^T$, in the hope of a different search trajectory. As evident in the figure, the new starting solution indeed follows a different path to the second (feasible) optimal solution. In essence, invoking a point-based algorithm with different starting solutions can be seen to lead to different optimal solutions in a multi-modal problem.

2.2.1.2 Point-Based Algorithms for Multi-objective Problems

To address MOPs, point-based optimization algorithms scalarize multiple objectives into a single aggregate function. One common approach is to minimize the *weighted sum* of objectives, as defined in Eq. 2.6.

$$\begin{aligned} &\text{Minimize } F(X) = \sum_{m=1}^{M} w_m f_m(X), \\ &\text{subject to } X \in \mathcal{X}. \end{aligned} \tag{2.6}$$

Ideally, this requires that each objective be normalized to have a similar range of values in the feasible search space and then use a specific weight vector $W = (w_1, w_2, \ldots, w_M)$, where $w_m \geq 0$ for each $m \in [1, M]$, and often satisfies $\sum_{m=1}^{M} w_m = 1$, to solve the SOP in Eq. 2.6. It can be proven that the resulting solution X_W^* is a member of *PS*, and the corresponding objective vector, $F(X_W^*)$, is a member of *PF*. Notably, this approach suffers from multiple challenges, as highlighted below:

- *PF*-approximation with a limited number of uniformly distributed solutions requires multiple invocations of a point-based algorithm, each time with a different weight vector. However, even a uniformly distributed set of weight vectors need not ensure a uniformly distribution of resulting solutions on the *PF*.
- A large computational effort is required due to independent applications of a point-based algorithm, particularly when no learning is derived from multiple runs to help enhance the search efficacy in any run.
- It reportedly fails to offer solutions on the non-convex parts of the *PF*.

While the first two limitations cited above remain common to the different scalarization approaches to tackle MOPs, the third limitation pertaining to non-convex *PF* can be alleviated. One such approach is the *epsilon-constraint* approach, as defined in Eq. 2.7.

$$\begin{aligned} &\text{Minimize } f_\mu(X), \\ &\text{subject to } f_m(X) \leq \epsilon_m, \ \text{ for } m = 1, 2, \ldots, M, m \neq \mu, \\ &\qquad\qquad X \in \mathcal{X}. \end{aligned} \tag{2.7}$$

In that, one particular objective is minimized at a time, say $f_\mu(X)$, while all the remaining objectives are converted into constraints using a pre-specified $\boldsymbol{\epsilon}$ vector, representing an upper bound on each of these objectives. The resulting optimal solution X_ϵ^* has been proven to be a member of *PS*, and the corresponding objective vector, $F(X_\epsilon^*)$, is a member of *PF*. It has also been proven that all Pareto-optimal solutions can be achieved using different combinations of $(\mu, \boldsymbol{\epsilon})$. Notably, the choice of the $\boldsymbol{\epsilon}$ vector must be within the ideal and nadir values of each objective, and this requires additional effort. Finding the ideal (best) value of an objective is relatively straightforward since it can be obtained by minimizing the objective independently subject to the original constraints and variable bounds. However, finding the nadir value (worst value among all Pareto-optimal solutions) of an objective is quite challenging, since it cannot be obtained without knowing the entire set of critical Pareto-optimal solutions.

There exist various other scalarization approaches, which interested readers can find in [3, 8, 34, 46]. However, one other scalarization approach which is quite commonly used for tackling MOPs is covered here. It pertains to the minimization of an *achievement scalarization function* (ASF) [50], as defined in Eq. 2.8.

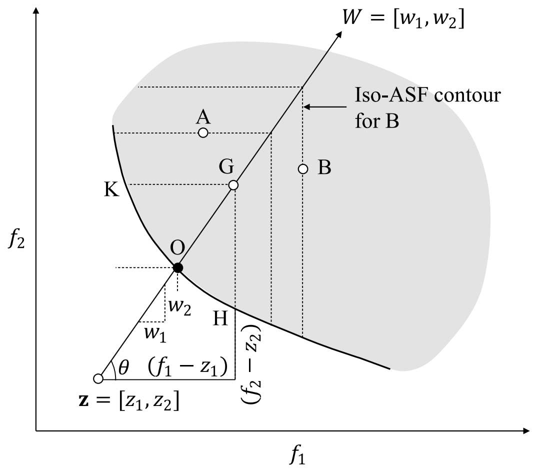


Fig. 2.6 Depicting the procedure for finding a Pareto-optimal solution using ASF

$$\begin{aligned}&\text{Minimize } \text{ASF}(X) = \max_{m=1}^{M} \frac{f_m(X) - z_m^r}{w_m},\\ &\text{subject to } X \in \mathcal{X}.\end{aligned} \tag{2.8}$$

The formulation requires two M-dimensional vectors: (i) the reference point $\mathbf{z}^r$, and (ii) the weight vector W, in the objective space ($\mathcal{Z}$ space). For any solution X in the $\mathcal{X}$ space, its $\mathcal{Z}$ space representation is referred to as $F(X)$. Given $F(X)$ and a weight vector W, $a_1 = (f_1(X) - z_1)/w_1$ and $a_2 = (f_2(X) - z_2)/w_2$ can be calculated, and the larger of the two quantities is chosen as the ASF value for the solution X. Figure 2.6 shows the contours of the ASF function for a specific $\mathbf{z}$ (identical to the ideal point) and $W = (w_1, w_2)^T$ in a two-objective space. To appreciate it, consider that the line represented by W has an angle θ with the horizontal axis, which implies $\tan(\theta) = w_2/w_1$. Furthermore, consider a point G belonging to the line given by W. Clearly, $\tan(\theta) = w_2/w_1 = (f_2 - z_2)/(f_1 - z_1)$, which implies $a_2 = (f_2 - z_2)/w_2 = a_1 = (f_1 - z_1)/w_1$. For the solutions lying on the line that joins G and H, a_1 remains constant while a_2 reduces, that is, $a_1 = ASF_{\text{G}} > a_2$. Similarly, for the solutions lying on the line that joins G and K, a_1 reduces while a_2 remains the same, that is, $a_1 < ASF_{\text{G}} = a_2$. Therefore, the iso-ASF contour for any solution can be given as a set of two lines (one horizontal and one vertical), intersecting the vector W emanating from $\mathbf{z}$. Based on this argument, Fig. 2.6 also shows, through dashed lines, the iso-ASF contours for two different points A and B. Since at least one of the $F(X)$ components for point A is larger than those of G, $ASF_{\text{A}} > ASF_{\text{G}}$. Similarly, $ASF_{\text{B}} > ASF_{\text{A}}$. Considering that the goal is to minimize the ASF value (Eq. 2.8), solution G is considered to be better than A, which is better than B. In this context, for given $\mathbf{z}$ and W, the point O will offer a minimum value ASF and therefore its underlying X while being the optimum (X^*_{ASF}) for this ASF problem would be a member of PS.

It is intuitive that by keeping the reference point $\mathbf{z}^r$ fixed and simply changing the weight vector, the entire PF can be approximated. Notably, the ASF function is non-differentiable due to the max operator. However, the smooth version given by Eq. 2.9, involving an additional variable, can be optimized so that $ASF(X^*) = ASF^{\text{smooth}}(X^*) = x^*_{n+1}$.

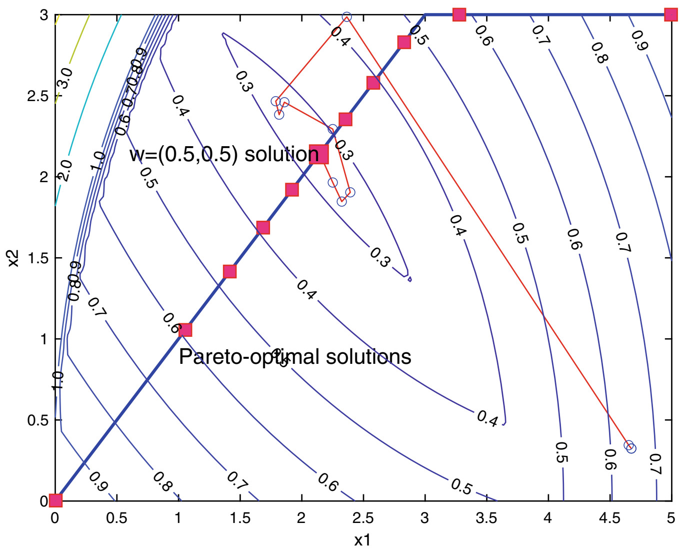
$$\begin{aligned}&\text{Minimize } ASF^{\text{smooth}}(X) = x_{n+1},\\&\text{subject to } \tfrac{f_m(X)-z^r_m}{w_m} \le x_{n+1},\ \text{for } m = 1, 2, \ldots, M;\ \text{and } X \in \mathcal{X}\end{aligned} \tag{2.9}$$

$$\begin{aligned}\text{Minimize } & f_1(x_1, x_2) = 4x_1^2 + 4x_2^2,\\ \text{Minimize } & f_2(x_1, x_2) = (x_1 - 5)^2 + (x_2 - 5)^2,\\ \text{subject to } & g_1(x_1, x_2) = (x_1 - 5)^2 + x_2^2 \le 25,\\ & g_2(x_1, x_2) = (x_1 - 8)^2 + (x_2 + 3)^2 \ge 7.7,\\ & 0 \le x_1 \le 5, \quad 0 \le x_2 \le 3.\end{aligned} \tag{2.10}$$

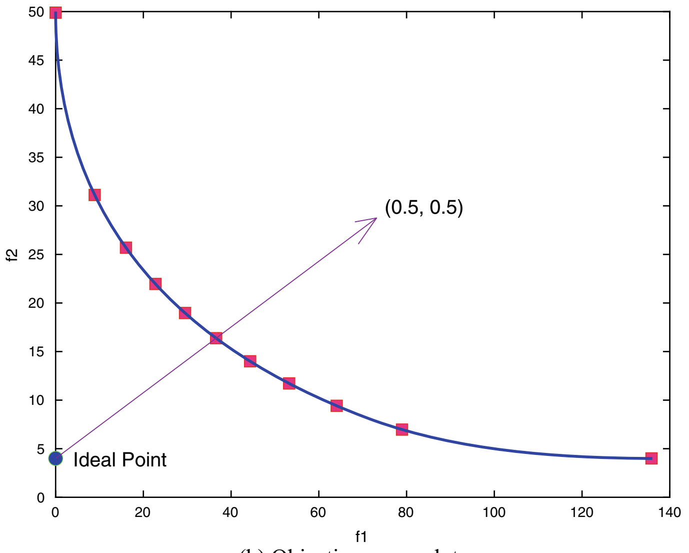
To illustrate the functioning of ASF-based scalarization approach, the problem defined in Eq. 2.10 is considered, and the SOP resulting from ASF^{smooth} has been solved through the `fmincon()` routine of MATLAB. Figure 2.7(a) shows 11 different Pareto-optimal solutions, corresponding to 11 uniformly spaced weight vectors $((1, 0),\ (0.9, 0.1),\ \ldots,\ (0, 1))$. For the particular choice of $W = (0.5, 0.5)$, it also shows the search trajectory leading to one of the Pareto-optimal solutions. Furthermore, Fig. 2.7b makes it evident that the images of 11 solutions in the $\mathcal{Z}$ space coincide with the true PF shown by a curve.

Finally, the salient features of point-based optimization algorithms may be noted:

- They are usually theoretically motivated and thus usually provide a convergence to at least local optimal solution, though they are vulnerable to remaining stuck at such solutions.
- Owing to their basis in theory (optimality conditions), they are quite rigid and not amenable to customization or incorporation of any problem-knowledge.
- For uni-modal SOPs, they happen to be computationally efficient and also require little memory.
- For MOPs or even multi-modal SOPs, they are not computationally efficient due to the need for multiple invocations for a reasonable PF-approximation.
- Since each Pareto-optimal solution is obtained one at a time, it is difficult to find a well-distributed set of solutions on the Pareto front, particularly for more than three-objective problems.



(a) Variable space plot for $W = (0.5,0.5)$. Contour plot of the ASF function is drawn with a penalty function



(b) Objective space plot

Fig. 2.7 Depicting the capability of a point-based algorithm, ASF^{smooth}, to find multiple Pareto-optimal solutions, one at a time for a different W-vector. The search trajectory corresponding to $W = (0.5, 0.5)$ is also shown. The true Pareto-optimal set and the Pareto front are highlighted by a solid line in both figures

2.2.2 Population-Based Optimization Algorithms

Unlike point-based algorithms, population-based algorithms work with a set of solutions that are improved iteratively, in pursuit of Pareto-optimal solutions, in a single run/invocation.

Algorithm 2.2: `Evolutionary_Optimization`(Operators,P_0,$t_{\max}$)

```
Input: Selection operator, Recombination operator, Mutation operator, $P_0$, maximum
       iterations $t_{\max}$
Output: Optimized solution $P_{(t_{\max})}$
1 $t \leftarrow 0$
2 Evaluate($P_t$)
3 for $t$ = 0 to ($t_{\max} - 1$) do
4     $P_{\text{mating}} = \text{Selection}(P_t)$
5     $Q_t = \text{Recombination}(P_{\text{mating}})$
6     $Q_t = \text{Mutation}(Q_t)$
7     Evaluate($Q_t$)
8     $P_{t+1} = \text{Survivor}(P_t, Q_t)$
9     $t \leftarrow t + 1$
```

A pseudo-code for a population-based evolutionary optimization (EO) algorithm is presented in Algorithm 2.2. Notably, it starts with a population of randomly created initial solutions (P_0) of size N. After evaluating the solutions, EO moves into a loop of iterations in which P_t is modified to a new population P_{t+1} by (i) selecting good solutions of P_t using a mating selection procedure to create a mating pool P_{mating}, (ii) using P_{mating} to create a new offspring population Q_t sized N using genetically motivated variation operators, such as recombination and mutation, and (iii) then selecting N best solutions from a combined parent–offspring population ($P_t \cup Q_t$) of size $2N$ in a survival selection operator. Iterations are continued until a termination criterion is met. While several variations of the above procedure can be implemented, the creation of new solutions by utilizing properties of good existing solutions is the hallmark of an EO algorithm. For example, an EO algorithm can be modified by customizing any of its following components:

- **Supply of initial population:** The knowledge of any existing good solution(s) can be used to create the initial population. Local perturbation of existing solutions can be made to fill the initial population from a few known solutions.
- **Choice of the mating selection operator:** Usually, a tournament selection operator is used, in which two population members compete, and the one with better *fitness* is chosen. Such a process with replacement results allows the selection of good solutions from the population to form the mating pool. Alternatively, a probabilistic tournament selection [27] or multi-membered tournament selection operator [24] can also be used, depending on the desired degree to which good solutions are to be emphasized.

- **Choice of variation operators:** Usually, two types of variation operators are used in EO studies, namely recombination and mutation. The recombination operator is applied over two or more parent solutions to create one or more offspring solutions. The recombination operators can be classified, on the basis of their application, into variable- and vector-wise operators. In variable-wise operators, such as simulated binary crossover (SBX) [10], the recombination is applied separately on each variable, so that the mathematical operation on one variable does not affect the operation on other variables. In contrast, in vector-wise operators, such as parent-centric crossover (PCX) [11] or differential evolution [36], the recombination is applied on the vector of the variable itself, so the same operation is applied to each constituent variable. The general form to create a new offspring solution from κ population members of P_{mating} (κ-parent recombination) is as follows:

$$Y = \text{Recombination}\,(X^{(i)}, i = 1, 2, \ldots, \kappa). \tag{2.11}$$

  One of the simplest recombination operators would be to take an average of κ parent solutions (selected from P_{mating}). However, recombination operators can be more sophisticated, which may be applied to current and past parent solutions (from previous generations), as demonstrated in [31].
  While a recombination operator involves two or more solutions to create the offspring solution(s), a mutation operator involves only a single solution. Similarly to recombination, mutation can be applied in a variable-wise manner, as in the polynomial mutation operator [13], or in a vector-wise manner, as in multivariate Gaussian operator [44]. If carefully considered, the mutation complements the tasks that recombination operators cannot perform. When certain variables lose diversity among all population members while the population has not yet converged to the optimum, the recombination operators cannot improve their diversity significantly. The only way to improve diversity is to apply mutation operators to modify the values of such variables. There can be other ways to apply mutation, such as using a local search operation or creating problem-specific mutation operators [19].
- **Choice of survival Operator:** The goal of the survival operator is to choose the best set of solutions from the combined current and newly created offspring populations. This operator also ensures that the previously obtained elite solutions survive from generation to generation. The survival operator can bring in other selection criteria, such as age-related selection and selection of a diverse population.
- **Choice of termination condition:** Most EO studies are terminated after a predefined number of iterations have elapsed or a predefined number of solution evaluations have been completed. But other termination criterion, such as search-stabilization [40, 42], achievement of a target solution quality, or satisfaction of certain optimality conditions [9], can also be used.

In general, EO algorithms offer a few advantages, as highlighted here:

- Being population-based, such algorithms are naturally suited for multi-modal SOPs and MOPs.
- Since population members can be compared against each other in a relative sense at each iteration, they allow for an iteration-independent evaluation scheme, if desired, in certain problems.
- Any normalization procedure can also be easily applied within a population of solutions, instead of finding normalization constants for the entire search space.
- The members of the population can be classified according to their objective and constraint values, and their effective recombination (or blending) can be executed to obtain new solutions. This process can enable *implicit parallel* search [8], which point-based algorithms cannot offer.

In addition to a number of EO algorithms, such as, binary and real-coded genetic algorithms [8, 23, 28], which closely follow Algorithm 2.2, there exists a number of other population-based optimization algorithms, such as differential evolution [47] and particle swarm optimization [31], which follow a slightly different procedure, but making an implicit algorithmic similarity [16, 35] with Algorithm 2.2.

2.2.2.1 Population-Based Algorithms for Single-objective Problems

At first glance, the use of population-based EO algorithms to solve uni-modal SOPs may seem like an overkill. However, even for such problems, their usage promises several advantages. First, a population can quickly lead to the optimal region through its implicit parallel search process compared to serially traversing the search space with a single solution. Second, the population provides a better global perspective for the algorithm in finding better optimal solutions. Third, the presence of a population of solutions in each iteration allows for any objective normalization, a relative comparison of solutions, or other derived properties of the search space.

For multi-modal SOPs, EO algorithms are naturally suited, owing to their ability to find and store multiple optimal solutions in a single application. However, in such problems, the selection operator should not compare two solutions that have more than a threshold distance (*niching* distance) in the $\mathcal{X}$ space. This way, two population members lying on distinct optimal basins will not be compared against each other, and both will co-survive in the population. Such an approach is expected to be more computationally effective than finding a single optimal solution at a time using a point-based algorithm.

As mentioned earlier, both mating and survival selection operators favor solutions with better *fitness*. Fitness in unconstrained problems can be directly indicated by the objective function value. However, in the case of constrained problems, the constraint violations also need to be factored in, besides the objective function value. To determine the extent of constraint violations by a solution X, the constraints are first normalized and an aggregate constraint violation value is calculated according to Eq. 2.12.

$$\mathrm{CV}(X) = \sum_{j=1}^{J} \langle \bar{g}_j(X) \rangle + \sum_{k=1}^{K} |\bar{h}_k(X)|. \tag{2.12}$$

In that, the bracket operator $\langle\ \rangle$ is such that $\langle \alpha \rangle = \alpha$, if $\alpha > 0$; zero, otherwise. Thus, for a solution $\bar{X}$ violating the j-th constraint $g_j(\bar{X}) \leq 0$, $\langle g_j(\bar{X}) \rangle$ takes a positive value, making $\mathrm{CV}(X)$ non-zero. To balance the contribution of each constraint, each constraint violation is normalized by dividing it by the largest violation of that constraint in the population (P), which can be expressed as follows [45]:

$$\langle \bar{g}_j(X) \rangle = \frac{\langle g_j(X) \rangle}{\max\limits_{X \in P} g_j(X)}, \text{ and } |\bar{h}_k(X)| = \frac{|h_k(X)|}{\max\limits_{X \in P} |h_k(X)|}. \tag{2.13}$$

Notably, $\mathrm{CV}(X) = 0$ shall imply that X is a feasible solution that satisfies all equality and inequality constraints. Any X for which $\mathrm{CV}(X) > 0$ is considered infeasible. In this context, the relative fitness of two solutions, say X and Y, can be assessed as follows [7]:

- If X is feasible, that is, $\mathrm{CV}(X) = 0$, and Y is infeasible, that is, $\mathrm{CV}(Y) > 0$, then feasibility is given preference, and X is chosen over Y.
- If both X and Y are infeasible, and $\mathrm{CV}(X) < \mathrm{CV}(Y)$, then lower constraint violation is given preference, and X is chosen over Y.
- If both X and Y are feasible, then much like the unconstrained case, fitness is indicated by the objective function value, and the solution with the better objective value is chosen. That is, if both X and Y are feasible and $f(X) < f(Y)$, then X is chosen over Y for minimization problems.

To demonstrate the working of a population-based EO algorithm, both the uni-modal SOP in Eq. 2.3 and its multi-modal variant emanating from adapted variable bounds ($-5 \leq x_1 \leq 5$ and $-5 \leq x_2 \leq 5$) are considered. In that:

- For the uni-modal problem: The real-parameter genetic algorithm (RGA) with binary tournament selection, SBX operator (distribution index of 2), and polynomial mutation (distribution index of 50) is chosen as the EO algorithm. Working with a population size of 20, the performance of RGA up to 50 iterations is presented in Fig. 2.8. It can be seen that the initial population contained both feasible and infeasible solutions; however, the population quickly became feasible and clustered around the optimal solution in merely 10 iterations. Then, after a few more iterations, the entire population converged very close to the optimal solution, visually showing no difference among the solutions.
- For the multi-modal problem: The same RGA is used, but with a niching-based selection operator (with a niching parameter of 0.1). Its performance captured in Fig. 2.9 reveals that it is able to quickly make the population members feasible and find sub-populations around two or more optimal solutions simultaneously, in a single run.

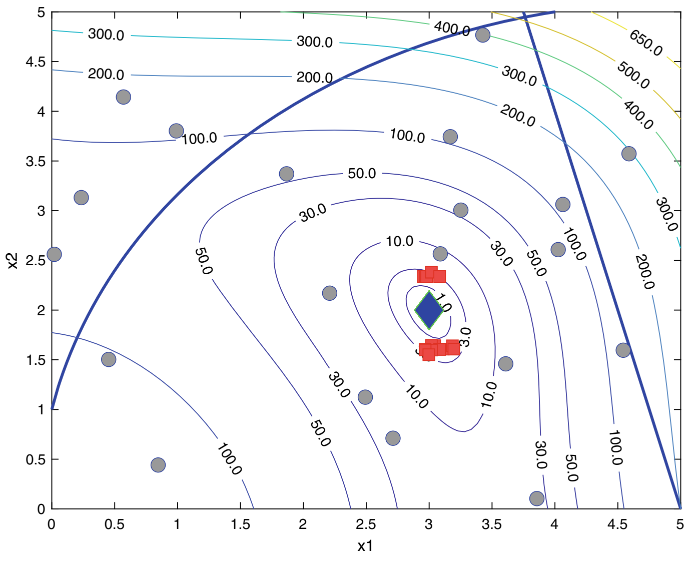


Fig. 2.8 Populations at iteration zero (blue circles), 10 (red squares), and 50 (green diagonals) are shown for the uni-modal problem, indicating the population-based optimization algorithm can converge to the uni-modal optimum with iterations

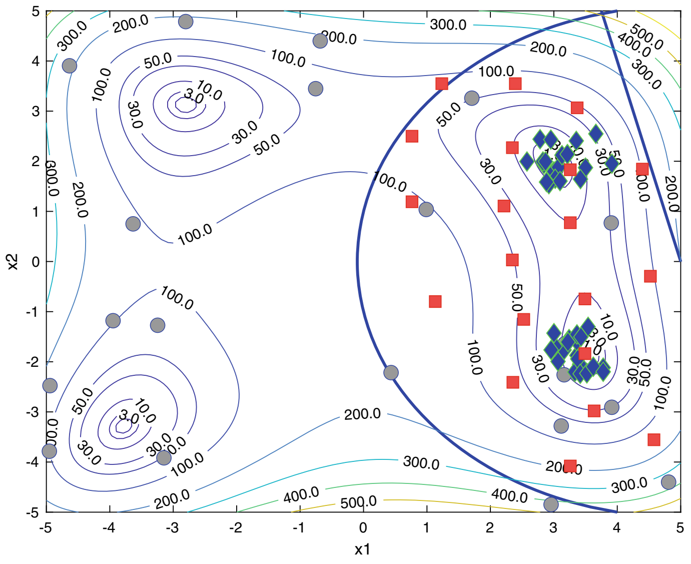


Fig. 2.9 Populations at iteration zero (blue circles), 20 (red squares), and 50 (green diagonals) are shown for the multi-modal problem, indicating the population-based EO can converge to the multiple optimal solutions simultaneously with iterations

2.2.2.2 Population-Based Algorithms for Multi-objective Problems

It is intuitive that population-based EO algorithms are naturally suited to handle MOPs, since they are capable of arriving at a PF-approximation in a single algorithmic run. EO algorithms that are designed for handling MOPs are referred to as EMOAs. It has been highlighted earlier in Chap. 1 that EMOAs iteratively evolve a randomly initialized population (a set of solutions), using the principles of natural evolution, namely variation and selection, toward a good PF-approximation. As shown in Fig. 2.10, a good PF-approximation requires (i) *convergence*: close proximity of the population to the true PF, and (ii) *diversity*: complete coverage of the true PF by the population and uniform distribution of population members across that spread, as much as possible. It has also been highlighted in Chap. 1 that most existing EMOAs are Pareto-dominance-based, indicator-based, or decomposition-based. While a detailed discussion on each of these EMOA classes is beyond the purview of this book, some fundamental concepts pertaining to Pareto-dominance-based EMOAs are presented here. In general, such EMOAs pursue convergence by utilizing the notion of Pareto-dominance, and diversity by employing a density criterion that seeks to favor less crowded solutions.

More details are presented here, in reference to one of the most popular EMOAs, namely NSGA-II [17]. For convergence, NSGA-II applies the notion of Pareto-dominance to categorize the population members into different non-domination ranks and sorts these categories in descending order of importance, so that the (mating and survival) selection operators can attach higher fitness to population members occupying better non-domination ranks. If the selection operators need to distinguish between population members occupying the same non-domination rank, then NSGA-II employs crowding distance as the density criterion and attaches relatively higher fitness to population members that are less crowded. The crowding distance for any solution reflects on its distance from its neighboring solutions, collectively. In other words, it indicates the size of the void that would be created if that solution were to be eliminated. With this background, reference may be made to NSGA-II's pseudo-

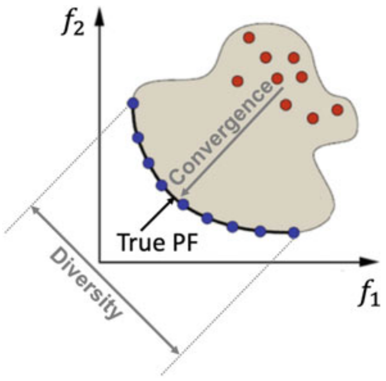
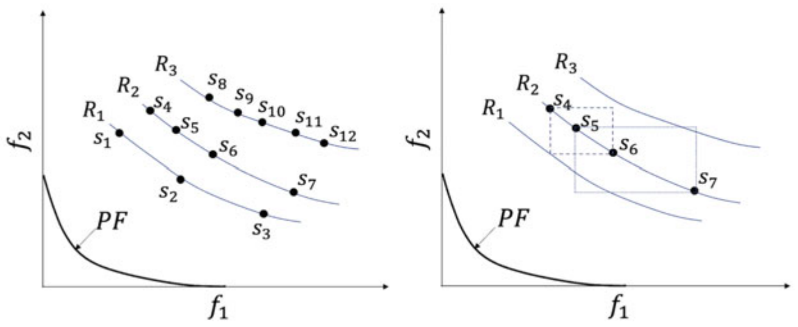


Fig. 2.10 A symbolic depiction of the two goals of EMOAs: convergence (proximity to the true PF) and diversity (complete spread and uniform distribution within that spread)

Algorithm 2.3: Iteration t of NSGA-II(P_t,R_t, CD$_t$)

```
Input: Population $P_t$, non-domination rank of population members $R_t$, crowding distance
       of population members CD$_t$
Output: Population $P_{t+1}$, rank $R_{t+1}$, crowding distance CD$_{t+1}$
1  $N = |P_t|$
2  $P_{t+1} = \emptyset$
3  $Q_t$ = Mutation (Recombination (Selection ($P_t$, $R_t$, CD$_t$)))
4  Evaluate($Q_t$)
5  $S_t = P_t \cup Q_t$
6  $[\mathcal{Z}_1, \mathcal{Z}_2, \ldots]$ = Non-Dominated-Sorting($S_t$)
7  $k = 1$
8  while $|P_{t+1}| \leq N$ do
9      $P_{t+1} = P_{t+1} \cup \mathcal{Z}_k$
10     $R_{t+1}^{\mathcal{Z}_k} = k$
11     Evaluate CD($\mathcal{Z}_k$)
12     $k = k + 1$
13 $P_{t+1} = P_{t+1}/\mathcal{Z}_{k-1}$
14 $\mathcal{Z}_{k-1}^{\text{div}}$ = Sorted ($\mathcal{Z}_{k-1}(1 : (N - |P_{t+1}|)$, 'Descending CD')
15 $P_{t+1} = P_{t+1} \cup \mathcal{Z}_{k-1}^{\text{div}}$
```

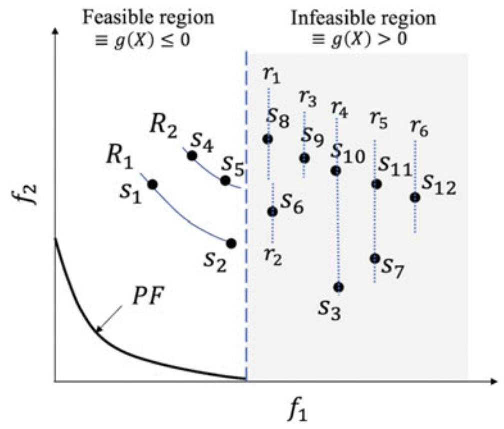
code given in Algorithm 2.3. It can be seen that environmental selection involves sorting the combined population ($2N$ members) into increasing domination ranks. Then, starting with the selection of the best-rank members (rank-one), the poorer-rank members (rank-two onward) are selected until the total number of selected solutions does not exceed the population size N. Then, members of the last rank that take the number of selected solutions more than N are evaluated for their crowding distance values. The ones having the highest crowding distance value are chosen to fill up the population size. In doing so, the extreme members are always included, since they are assigned infinite crowding distance (under considerations of achieving a larger spread). For ease of interpretation, Fig. 2.11 illustrates the NSGA-II selection procedure, with reference to a combined population, sized 12, of which the six fittest members are to be selected for the next generation. In that, Fig. 2.11a shows that these 12 members are categorized into three non-domination ranks. S_1, S_2, and S_3 qualify for rank-one (R_1), since they do not get dominated by any other population member. Clearly, they represent the best/elite members of the population. By temporarily discounting these three solutions, S_4 to S_7 qualify as rank-two (R_2) solutions, and the remaining as rank-three (R_3) solutions. Then, for the next generation's population, all the R_1 solutions qualify, and R_2 becomes the last rank including whose solutions exceed the desired population size of 6. Hence, only three solutions from R_2 can be selected. In that, S_4 and S_7 being extreme solutions get selected, and among S_5 and S_6, the one with higher crowding distance needs to be selected. The bounding boxes around S_5 and S_6 in Fig. 2.11b, indicative of their respective crowding distances, help select S_6. Its larger crowding distance suggests that eliminating it would leave a larger void, and retaining it promises a more uniform distribution of solutions. Hence $P_{t+1} = \{S_1, S_2, S_3, S_4, S_6, S_7\}$. Interestingly, the same procedure holds for



(a) Non-domination ranking (b) Crowding distance for S_5 and S_6

Fig. 2.11 Depicting NSGA-II's selection procedure: convergence based on non-domination ranking, and diversity based on crowding distance

Fig. 2.12 Constrained dominance principle implying an altered non-domination-ranking



mating selection. In that, if one of the two competing solutions belongs to a higher rank, it is selected. If both belong to the same rank, then the one with the larger crowding distance is selected.

As in the case of SOPs, the selection procedure discussed above also needs to be adapted for constrained MOPs. Toward this, the Pareto-dominance principle is replaced by *constrained dominance* principle [17]. A solution X is said to constrained-dominate Y, if *any one* of the following is true:

1. Both X and Y are feasible; X Pareto-dominates Y.
2. X is feasible; Y is infeasible.
3. Both X and Y are infeasible; X has a smaller constraint violation than Y.

Under the constrained dominance principle, the non-domination ranking of the solutions in Fig. 2.11a gets altered, and the revised ranking is shown in Fig. 2.12. In that, the $\mathcal{Z}$ space is shown to be split into feasible and infeasible regions through a constraint boundary (depicted by the vertical dashed line), such that any dis-

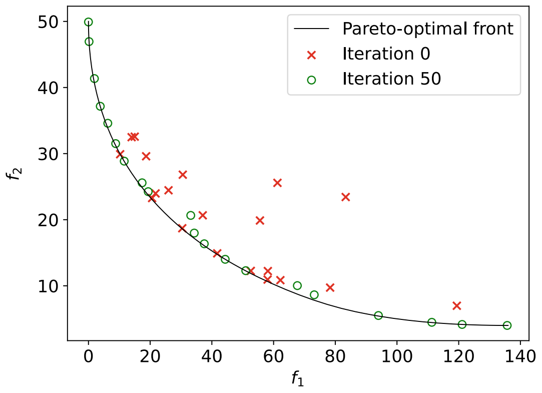


Fig. 2.13 NSGA-II's progress from generation zero to 50 with 20 members

tance to the right of this boundary marks the degree of infeasibility. Since feasible solutions are prioritized and ranked using Pareto-dominance, $\{S_1, S_2\} \in R_1$ and $\{S_4, S_5\} \in R_2$. All remaining solutions, which are infeasible, are ranked according to their respective violations of the constraints, as shown by r_1 to r_6. Therefore, $P_{t+1} = \{S_1, S_2, S_4, S_5, S_6, S_8\}$.

It is well-known that NSGA-II is fast and works extremely well for two- and three-objective problems. For a sample illustration, its performance in the two-objective problem given by Eq. 2.10 is shown in Fig. 2.13. It can be seen that, starting with 20 random initial solutions, NSGA-II could offer a widely distributed set of solutions close to the true *PF* in just 50 generations.

2.2.2.3 Population-Based Algorithms for Many-objective Problems

As highlighted in Chap. 1, 2000–2005 marked an unsettling phase for EMO researchers, because with an increase in the number of objectives beyond three, the *otherwise very effective* notion of Pareto-dominance ceased to induce selection pressure for convergence, and the density criterion favoring less crowded solutions turned counterproductive. In that:

- The efficacy of Pareto-dominance was marred by *dominance resistance* [37], plausibly due to its inability to account for the critical human decision-making (HDM) elements [22]: (i) the *number* of improved objectives, (ii) the *extent* of improvements, and (iii) *objectives' relative preferences*.
- The density-based diversity preservation worsened the performance by diversifying poorly converged solutions, an undesired feature, referred to as *active diversity promotion* [37].

Recognizing that a free-formed evolution of widely spread and well-converged Pareto-optimal solutions is quite challenging, the focus shifted to the development of reference vector (RV)-based evolutionary multi- and many-objective optimization algorithms, namely RV-EMâOAs. As depicted for a two-objective instance in

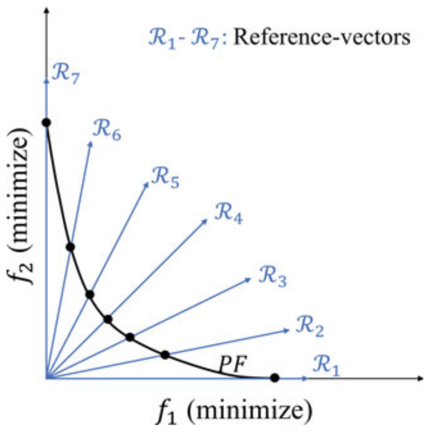


Fig. 2.14 Depicting the $\mathcal{Z}$ space decomposition by RV-EMâOAs

Fig. 2.14, RV-EMâOAs decompose the $\mathcal{Z}$ space through a uniformly distributed set of RVs, so that a Pareto-optimal solution can be found along each RV. Clearly, the role of RV generation in the positive orthant of an M-dimensional $\mathcal{Z}$ space, with uniformity, is critical. In addition to structured methods [5], recent Riesz-energy-based methods are used to create an arbitrary number of RVs [12]. Since RV generation methods do not account for feasible/infeasible regions or the shape of the true PF, one Pareto-optimal solution per RV may not be possible, which implies that the number of Pareto-optimal solutions could be lesser than the number of RVs. Recent computationally efficient methods re-adjust RVs to produce exactly the same number of Pareto-optimal solutions as the number of RVs [18].

One of the most popular RV-EMâOAs is NSGA-III [14]. The pseudo-code for any generation t of NSGA-III is presented in Algorithm 2.4. Unlike NSGA-II, there is no mating selection operator in NSGA-III since the population size is chosen identical to the number of RVs. In addition, environmental selection involves an RV-based selection process. The non-dominated ranking and selection of a few best non-dominated front members to P_{t+1} is identical to that in NSGA-II. However, for the last non-dominated rank members (front $\mathcal{Z}_{k-1}$ in Algorithm 2.4) which could not be accepted in total for P_{t+1}, an RV-based selection process is applied to pick the remaining N_{rem} members from $\mathcal{Z}_{k-1}$, rather than using the crowding distance operator. In the RV-based process, first, the ideal and nadir points are updated based on the extreme objective values of the solutions in $\mathcal{Z}_{k-1}$. Then, these points are used to normalize the solutions in $\mathcal{Z}_{k-1}$ (in the $\mathcal{Z}$ space). Subsequently, each solution in $\mathcal{Z}_{k-1}$ is associated with a specific RV based on its minimum orthogonal distance from the RVs (in normalized $\mathcal{Z}$ space). Finally, for each RV, the closest member in terms of orthogonal distance is selected in the niching operator to fill the P_{t+1} population. Since there are at most N RVs, this process will end with the choice of a maximum of N members for P_{t+1}. If certain RVs have empty members, then the second closest member for each RV is chosen until N members are selected, and so forth. Since the constrained dominance principle is used to sort the non-dominated ranks, NSGA-III works well for constrained problems as well [30].

Algorithm 2.4: Iteration t of `NSGA-III`(P_t,$\mathcal{R}$, Z^{ideal}, Z^{nadir})

```
Input: Population $P_t$, Reference set $\mathcal{R}$, Ideal point $Z^{\text{ideal}}$, Nadir point $Z^{\text{nadir}}$
Output: Population $P_{t+1}$, Updated ideal point $Z^{\text{ideal}}$, Updated nadir point $Z^{\text{nadir}}$
1  $N = |P_t|$
2  $P_{t+1} = \emptyset$
3  $Q_t$ = Mutation (Recombination ($P_t$))
4  Evaluate($Q_t$)
5  $S_t = P_t \cup Q_t$
6  $[\mathcal{Z}_1, \mathcal{Z}_2, \ldots]$ = Non-Dominated-Sorting($S_t$)
7  $k = 1$
8  while $|P_{t+1}| \leq N$ do
9      $P_{t+1} = P_{t+1} \cup \mathcal{Z}_k$
10     $k = k + 1$
11 $P_{t+1} = P_{t+1}/\mathcal{Z}_{k-1}$
12 $\left(Z^{\text{ideal}}, Z^{\text{nadir}}\right)$ = Update $\left(Z^{\text{ideal}}, Z^{\text{nadir}}, \mathcal{Z}_{k-1}\right)$
13 $\mathcal{Z}_{k-1}^{\text{norm}}$ = Normalize $\left(\mathcal{Z}_{k-1}, Z^{\text{ideal}}, Z^{\text{nadir}}\right)$
14 $\mathcal{Z}_{k-1}^{\text{asso}}$ = Associate $\left(\mathcal{Z}_{k-1}^{\text{norm}}, \mathcal{R}\right)$
   /* niching operator fills population $P_{t+1}$ */
15 $r = 1$
16 while $|P_{t+1}| < N$ do
17     if $r > |\mathcal{R}|$ then
18         $r = 1$
19     $\mathbf{z}$ = orthogonally_closest $\left(\mathcal{Z}_{k-1}^{\text{asso}}, \mathcal{R}_r\right)$ /* $\mathbf{z}$ may be empty */
20     $r = r + 1$
21     $\mathcal{Z}_{k-1}^{\text{asso}} = \mathcal{Z}_{k-1}^{\text{asso}}/\mathbf{z}$
22     $P_{t+1} = P_{t+1} \cup \mathbf{z}$
```

It is important to note that NSGA-III remedied the challenges associated with density-based diversity preservation by use of an RV-based architecture, but did not address the key challenge of dominance resistance, and continued to rely on the use of Pareto-dominance-based *ranking*, which is known to be ineffective for MaOPs. In this context, the recently proposed LHFiD by the authors [41] is noteworthy. While it takes care of diversity by employing an RV-based architecture, it resolves the root cause of dominance resistance by replacing Pareto-dominance with *hf*-dominance [41]. A solution X is said to *hf-dominate* another solution Y, denoted as $X \succ_{hf} Y$, if *both* the following conditions are met:

- X is better in more than or equal number of objectives than Y, implying, $n_b^X \geq n_b^Y$.
- For X, the weighted gain in objectives exceeds the weighted loss in objectives, compared to Y. If p_m is the relative preference for the mth objective, and $\Delta f_m = f_m^X - f_m^Y$, then for a minimization problem, the above can be described by $\Delta F_{X,Y} = \sum_{m=1}^{M} p_m \cdot \Delta f_m < 0$.

It is critical to note that *hf*-dominance is able to account for the *number* of improved objectives (n_b) and the *size* of improvements (Δf_m), without involving a new parameter, since these could be determined directly from the objective vectors of the solutions being compared. If the decision maker preferences for the objectives (p_m) are available, *hf*-dominance also provides the scope to account for them.

Based on extensive experiments, LHFiD has been shown to perform significantly better than other RV-EMâOAs based on different architectures, including: (i) RV-based NSGA-III, θ-DEA [52], and RPD-NSGA-II [21], (ii) decomposition-based MOEA/D-LWS [49], and (iii) dominance-based multiGPO [53]. The performance of LHFiD could largely be attributed to its underlying *hf*-dominance, which, for the

Algorithm 2.5: Generation t of LHFiD(P_t, $\mathcal{R}$, Z^{ideal}, Z^{nadir})

```
   Input: Population $P_t$, RV set $\mathcal{R}$, ideal point $Z^{\text{ideal}}$, nadir point $Z^{\text{nadir}}$
   Output: Population $P_{t+1}$, updated ideal point $Z^{\text{ideal}}$, updated nadir point $Z^{\text{nadir}}$
 1 begin
 2     $Q_t \leftarrow$ Mutation(Recombination($P_t$))
 3     Evaluate $Q_t$
 4     $S_t \leftarrow P_t \cup Q_t$
 5     $Z^{\text{ideal}} \leftarrow$ Update($Z^{\text{ideal}}$, $S_t$)
 6     if $Z^N \neq \emptyset$ then
 7         $\tilde{S}_t \leftarrow$ Normalize($S_t$, $Z^{\text{ideal}}$, $Z^{\text{nadir}}$)
 8     else
 9         $\tilde{S}_t \leftarrow$ Translate($S_t$, $Z^{\text{ideal}}$)
10     $S \leftarrow \emptyset$, $S^L \leftarrow \emptyset$, $\mathcal{R}^L \leftarrow \mathcal{R}$
11     Associate each solution $\tilde{S}_t$ with its nearest RV based on orthogonal distance
12     foreach $\mathcal{R}_j \in \mathcal{R}$ do
13         $\mathcal{C} \leftarrow$ All solutions associated with $\mathcal{R}_j$
14         Remove Pareto-dominated solutions in $\mathcal{C}$
15         $\alpha \leftarrow$ Solution in $\mathcal{C}$ with minimum orthogonal distance
16         if $\mathcal{R}_j$ is not an objective-axis vector then
17             $\beta \leftarrow \emptyset$
18             for $s \in \mathcal{C}$ do
19                 if $s$ hf-dominates $\alpha$ then
20                     $\beta \leftarrow \beta \cup s$
21             if count($\beta$) $\geq$ 1 then
22                 $\alpha \leftarrow$ Solution in $\beta$ with minimum orthogonal distance
23         $S \leftarrow S \cup \alpha$;   $S_t \leftarrow S_t \setminus \alpha$;   $\mathcal{R}^L \leftarrow \mathcal{R}^L \setminus \mathcal{R}_j$
24     foreach $r \in \mathcal{R}^L$ do
25         $\alpha \leftarrow$ Solution in $S_t$ nearest to $r$
26         $S^L \leftarrow S^L \cup \alpha$
27     $P_{t+1} \leftarrow S \cup S^L$
28     if $P_{t+1}$ is mildly stabilized then
29         $Z^N \leftarrow$ Update Nadir($P_{t+1}$, $Z^{\text{ideal}}$)
30     if $P_{t+1}$ is strictly stabilized then
31         Terminate LHFiD
```

first time, accounts for the number of objectives in which a solution is better; the size by which a solution is better; and objectives' relative preferences—explicitly and simultaneously, and without requiring tuning of any additional parameter.

The pseudo-code for any generation t of LHFiD is presented in Algorithm 2.5. Following the offspring (Q_t) creation, they are evaluated and merged with the parent population. It is imperative to note that, unlike other RV-EMaOAs, LHFiD does not compute and update the nadir point in each generation. Hence, if nadir point has been computed at least once, the population members are normalized using ideal and nadir points, else they are translated using just the ideal point. The normalized/translated solutions are then clustered around each RV based on orthogonal distance (similar to NSGA-III). From each cluster, exactly one solution is selected, based on Pareto-dominance and *hf*-dominance, and survived to P_{t+1}. If the total cluster count is N, the above step allows the survival of N solutions. Otherwise, if the cluster count is less than N, the nearest solutions to each of the empty RVs survive to P_{t+1}, leading to a surviving population of size N. Furthermore, LHFiD tracks the stability of the population using a stabilization tracking algorithm [40]. In that: (a) if mild stabilization is detected, the nadir point is updated; and (b) if strict stabilization is detected, the LHFiD run is terminated.

2.2.3 No-Free-Lunch (NFL) Theorem

The discussion on optimization algorithms cannot be complete without referring to the *No-Free-Lunch (NFL)* theorem [51]. In its simplest form, the NFL theorem can be described as follows. Imagine that someone is trying to establish which optimization algorithms between A_1 and A_2 are best in terms of a performance metric for solving *all* possible optimization problems (set $\mathcal{P}$). The experiments are set up to apply A_1 and A_2 to each problem p in $\mathcal{P}$ with a specific fixed resource (on a specific computer for a specific number of solution evaluations, etc.) and the performance indicator $I_1(p)$ is computed. After all the problems have been solved, an aggregate performance indicator I_1 is calculated from all $I_1(p)$ with $p \in \mathcal{P}$ values. The same experiments are repeated for algorithm A_2 using the same resource used for algorithm A_1 and an aggregate performance indicator value of I_2 is obtained. Their theoretical results show that

$$I_1 = I_2. \tag{2.14}$$

The theorem states that no algorithm is better than another algorithm in terms of solving *all* possible optimization problems. It is also argued that the NFL theorem breaks down if the scope of investigation is restricted to a subset of problems. It implies that if the problem of interest belongs to a specific problem class, then there can exist a specific super algorithm (A^{best}) that would be better than all other algorithms.

The theorem keeps optimization researchers modest in their claims and compels researchers to state the problem class for which an algorithm has been developed, as it is now well established that a single optimization algorithm cannot be most efficient for solving each and every problem that one can encounter. The NFL theorem also facilitates an important deduction that there is a need to customize an optimization algorithm for specific problem classes. Customization may involve the exploitation of specific characteristics and properties of the problem to design an efficient algorithm. One specific instance of a population-based customized optimization algorithm relates to solving a billion-variable integer linear programming (ILP) problem to near-optimality [15], while standard point-based ILP algorithms cannot solve more than 2000-variable version of the same problem class. Based on the above, customizability can be seen to be a desirable feature and strength for any algorithm. For instance, the fact that population-based algorithms rely on simple variation and selection operators and do not rigidly rely on mathematical computations such as gradient makes them more amenable to customization. This in turn could be seen as the strength of such algorithms, since available problem-specific knowledge could be incorporated for tuning of operators, enhancing their efficiency. The NFL theorem has also been extended for multi-objective optimization and a similar conclusion has been reached [4].

Notably, the NFL theorem does not deny the possibility that there can exist a specific super algorithm which performs better than all other algorithms for a specific problem class. This is manifested by the fact that point-based optimization algorithms with provable efficacy exist for certain structured optimization problems. For example, an LPP with linear objective and constraint functions of real-parameter variables can be solved to optimality with known computational complexity. The same is possible for QPPs with a quadratic objective function having a positive definite Hessian matrix and linear constraint functions. Some algorithms with proof exist for certain convex programming problems, but the generality of provable algorithms almost ends there. Furthermore, if even some of the problem characteristics (nature of objectives, constraints, or variables) for the respective problem classes cited above are altered, then the corresponding algorithms may no longer perform with the same efficacy. Therefore, the real challenge lies in developing flexible and customizable algorithms for generic problem scenarios. This may require a combination of point-based and population-based algorithms (for instance, point-based *local search* applied to solutions obtained from EO algorithms) with possible incorporation of problem-knowledge-driven heuristics.

2.3 Summary

This chapter started by highlighting the importance of optimization in different application domains. The optimization problems were then categorized according to the number of objectives involved and the corresponding solution features were highlighted. This was followed by the introduction of point-based and population-based optimization algorithms, and their suitability for uni-modal SOPs, multi-modal SOPs, MOPs, and MaOPs was discussed. Finally, the NFL theorem was presented, which offers researchers a sense of perspective, stating that no single algorithm can be better than other algorithms for *all* possible optimization problems. Hence, it is prudent to have a dedicated focus on solving specific problem classes at a time, rather than hoping to find a single optimization algorithm that is best suited for all possible problems. Similarly, in the context of generic (unstructured) problem scenarios, it is rather imperative to develop flexible and customizable algorithms. This chapter also includes an Appendix, which shares details on the generation of RVs that enable an important class of algorithms, namely RV-EMâOAs, that are intensively used in subsequent chapters.

Appendix

2.4 Generation of Reference Vectors (RVs)

It was discussed earlier in Sect. 2.2.2.3 that some EMâOAs use a uniform set of RVs, referred to as RV-EMâOAs, in their search method. In this appendix, some methods to create a uniformly distributed set of RVs, scalable for any number of objectives, have been discussed. But before discussing these methods in detail, the philosophy for the creation of RVs has been highlighted, in terms of covering the positive orthant of an M-dimensional $\mathcal{Z}$ space.

The purpose of creating a uniform set of RVs is to aid in the selection of uniformly distributed solutions in the $\mathcal{Z}$ space. This necessitates a mapping of solutions' objective vectors to these RVs, using some distance metric in the M-dimensional space. In any real-world problem, the objective values can be either positive or negative and can also be differently scaled. Thus, it is intuitive to bring the RVs and the objective vectors (of the population) to the same scale. Toward this, the objective values of all solutions are normalized, to bring them into the hypercube bounded by zeros and ones. One such way is to have minimum ($f_i^{\min}$) and maximum ($f_i^{\max}$) values computed for the non-dominated solutions in the population, and then compute the normalized objective values as below.

$$\bar{f}_i(X) = \frac{f_i(X) - f_i^{\min}}{f_i^{\max} - f_i^{\min}}. \tag{2.15}$$

For an EMâOA, $f_i^{\min}$ and $f_i^{\max}$ can be obtained from the current population, which provides an edge to population-based optimization algorithms. However, some sophisticated methods for computing $f_i^{\min}$ and $f_i^{\max}$ have also been proposed to deal with different scenarios of non-dominated solutions that may arise during the evolutionary process [14].

Normalization of objectives brings the non-dominated solutions within a positive and unit hypercube: $\bar{f}_i(X) \in [0, 1]$, for all objectives. The RVs can now be constructed in the same hypercube, spanned uniformly. Consider the three-dimensional hypercube (representing a three-objective case), bounded by $[0, 0, 0]$ (origin) and

[1, 1, 1] (diagonally opposite to origin), as shown in Fig. 2.15. To span the space from the origin (representing the ideal point) or from the diagonally opposite corner of the hypercube (representing the nadir point), one idea would be to consider the unit simplex shown in the figure and then find a uniformly distributed set of points on it, as shown in the figure (with 91 points). Thereafter, each point can be joined with either the ideal point (origin) or the nadir point (diagonally opposite to origin) to construct the RVs. These RVs will span the entire hypercube uniformly. If a single solution, close to each of these RVs and as close to the origin (or ideal point) as possible, can be found by an EMâO, then a reasonably good distribution on PF can be achieved. If the PF is linear in shape, the Pareto-optimal solutions will lie on the unit simplex itself; however, for PFs with nonlinear shapes, the solutions will deviate from the unit simplex. Unless the PF deviates by a large amount, the RVs derived from uniformly distributed points on the unit simplex will be sufficient to reasonably represent the PF.

Two prominent methods of creating a uniformly distributed set of RVs have been discussed in the following subsections. The first method follows a structured approach that can only produce a specific number of RVs (H), given a dimension (M). However, the second method follows a generic approach, where any number of RVs can be created in a given M-dimensional $\mathcal{Z}$ space.

2.4.1 Das–Dennis Method

The Das–Dennis method [5] involves a parameter p, indicating the number of gaps between RVs along one of the objective axes. For given objectives M and number of gaps p, the total number of equi-distant points on the unit simplex is given by

$$H(p, M) = \binom{M + p - 1}{p}. \tag{2.16}$$

The smallest distance between any two points is $\sqrt{2}/p$. Owing to this strict structure, an arbitrary number of points cannot be created using this method. Figure 2.15 shows the result with $M = 3$ and $p = 12$, to create 91 points. If $p = 13$ were used, it would create 105 points. Notably, no other number between 91 and 105 can be achieved using the Das–Dennis method.

There are some other properties of this method, as will be described next. In addition to the inability to construct an arbitrary number of points, there is another problem with the Das–Dennis approach. As M increases, the number of total points on the unit simplex increases rapidly, as shown for $M = 10$ on the right vertical axis in Fig. 2.16. Although a sublinear plot on the semilog scale indicates weaker than exponential behavior, an extremely large number of points are generated for very reasonable values of gaps p. Since the population size of an EMâOA is usually the

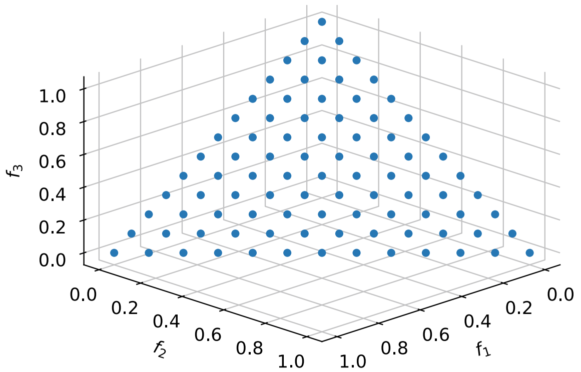


Fig. 2.15 Depiction of $\binom{3+12-1}{12} = 91$ points in a three-objective space, created using Das–Dennis method

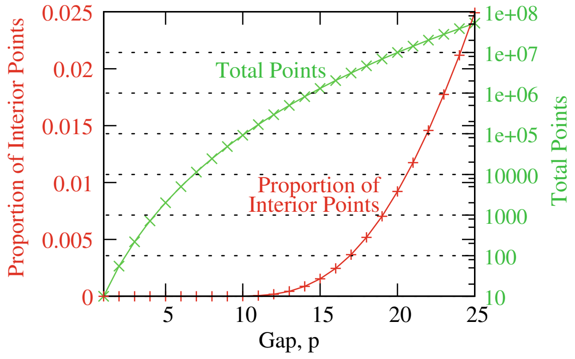


Fig. 2.16 Proportion of interior points compared to the total number of points created by the Das–Dennis method with $M = 10$ (taken from [2])

same as the number of points and consequently the number of RVs, this requires an extremely large population size.

Another crucial problem with large M is that the majority of points with reasonable values of p lie on the boundary of the unit simplex and only a few points lie in the interior region. Calculations reveal that $p < M$ leads to only boundary points and no interior points; and $p = M$ leads to all but one boundary point, where that one point lies in the center of the unit simplex. With $p > M$, points begin to appear in the interior region, but the count of such interior points is given by $n_I = \binom{p-1}{p-M}$, which is only a tiny fraction of the total number of points.

$$\rho_I = \frac{n_I}{n} = \frac{p!\,(p-1)!}{(p-M)!\,(M+p-1)!}. \tag{2.17}$$

Figure 2.16 shows that for $M = 10$, the proportion of interior points increases with p, but the proportion (left vertical axis) is very small.

To provide an example, with $p = 15$, a total of 1, 307, 504 points are created, of which only 0.15% (only 2, 002) points are in the interior region. The rest of the points lie on the boundary of the unit simplex.

2.4.2 Riesz-Energy-Based Unit Simplex Method

A recently proposed Riesz-energy based unit simplex method alleviates the rigid structure of the Das–Dennis method.

The motivation behind the use of an energy concept to obtain a well-distributed set of points comes from nature. Multi-body and interacting physical systems eventually settle on a state that corresponds to the minimum overall potential energy. For example, given two bodies, the potential energy is inversely proportional to the distance between them. The minimum potential energy solution corresponds to a diverse distribution of multiple bodies in a physical space with dimension s. When dealing with a high-dimensional space, a generalization of potential energy, called *Riesz s-Energy* [26], is used, which is defined between two points ($\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$) as follows:

$$U(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \frac{1}{\left\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\right\|^{s}}. \tag{2.18}$$

In the context of creating well-distributed points, it is not clear how the dimension s should depend on the number of objectives (M), but based on the motivation set in [2], $s = M^2$ is used here. For multiple (N) points, the overall s-energy can be given as follows:

$$U_T(\mathbf{z}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{1}{\left\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\right\|^{s}}, \quad \mathbf{z} \in \mathbb{R}^{n\times M}. \tag{2.19}$$

The basic concept behind this energy method is to find the $\mathbf{z}$-matrix of size $N \times M$ that minimizes U_T subject to every $\mathbf{z}^{(i)}$ vector to lie on the unit simplex, that is, $\sum_{m=1}^{M} z_m^{(i)} = 1$. In that, a gradient-based optimization method (Adam [32]) has been used. Due to the large magnitude of U_T, the logarithm of U_T is determined and then the partial derivative of $F_E = \log U_T$ with respect to $z_m^{(i)}$ is calculated as follows:

$$\frac{\partial F_E}{\partial z_m^{(i)}} = -\frac{2s}{U_T}\left[\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\left(z_m^{(i)} - z_m^{(j)}\right)}{\left\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\right\|^{s+1}}\right]. \tag{2.20}$$

To ensure that all points remain on the unit simplex, gradients are projected onto the unit simplex. Figure 2.17 shows the result for $M = 3$ with 92 points.

The general outline of the *s-Energy* method is as follows:

1. An initial point-set $\mathbf{z}^{(i)}$ ($i = 1, \ldots, n$) is generated on the unit simplex using the *Reduction* method.
2. Until the maximum number of iterations is reached or convergence criteria are met:

   a. The gradient is calculated with respect to $\mathbf{z}^{(i)}$:
   $\nabla_{\mathbf{z}^{(i)}} F_E = \left(\frac{\partial F_E}{\partial z_1^{(i)}}, \frac{\partial F_E}{\partial z_2^{(i)}}, \ldots, \frac{\partial F_E}{\partial z_M^{(i)}}\right)$ for all points $i = 1, 2, \ldots, n$.

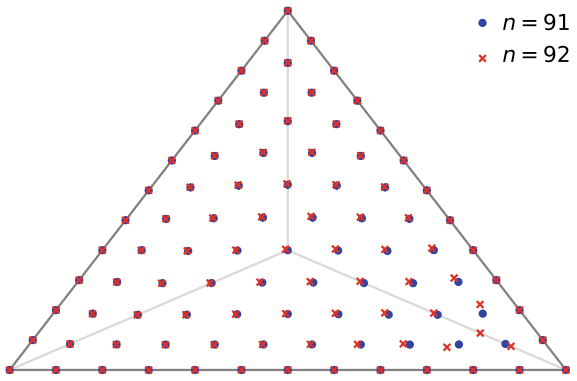


Fig. 2.17 Distribution of points obtained by *s-Energy* with 91 (circle) and 92 (cross) points. Notice how an additional point placed on the bottom-right corner adjusts the neighboring points to make a good distribution (taken from [2])

b. The calculated gradient is projected onto the unit simplex: $\nabla^{\text{proj}}_{\mathbf{z}^{(i)}} F_E = \nabla_{\mathbf{z}^{(i)}} F_E - \frac{1}{M} \times \left(\sum_{m=1}^{M} \frac{\partial F_E}{\partial z_m^{(i)}} \right) \hat{u}$, where $\hat{u}$ is a M-dimensional vector of the ones. This guarantees $\sum_{m=1}^{M} z_m^{(i)} = 1, \forall i$.
c. Each $z_m^{(i)}$ (for n points and for M dimensions) is updated using the gradient-based optimizer *Adam*. Whenever a point $\mathbf{z}^{(i)}$ is outside the unit simplex, it is projected back to the unit simplex to ensure it satisfies $z_m^{(i)} \geq 0, \forall i, m$.

Since the final distribution of points depends on the initial configuration, the *Reduction* method is used to generate an initial set of points. In that, the termination criterion is defined based on the average movement of all points (below 10^{-5}), and a restart of *Adam* optimizer is performed, in case no improvement has been made for 50 iterations. In addition, the maximum number of iterations is set to 3000.

Earlier, the result of the Das–Dennis method was presented with $p = 12$ for a three-objective scenario, which created 91 points. Figure 2.17 shows the distribution of 91 points in the unit simplex using the *s-Energy* method, marked with circles. Next, a set of 92 points is created from scratch using the *s-Energy* method, which is not possible with the Das–Dennis method. These points are marked with crosses. It is interesting to observe from the figure that only a few points toward the lower right corner of the simplex get readjusted to accommodate the extra (92nd) point. The choice of adjustment in the lower right corner is arbitrary and probably is an outcome of the starting distribution of points (random). The major portion of the overall distribution remains nearly similar to the configuration obtained for $n = 91$ points. Visually, the resulting set of 92 points is uniform, even though (to the best of authors' knowledge) it is known that no perfectly well-spaced distribution of points with 92 or any arbitrary number of points exists.

References

1. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms. Wiley, Singapore (2004)
2. Blank, J., Deb, K., Dhebar, Y., Bandaru, S., Seada, H.: Generating well-spaced points on a unit simplex for evolutionary many-objective optimization. IEEE Trans. Evol. Comput. **25**(1), 48–60 (2021). https://doi.org/10.1109/TEVC.2020.2992387
3. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making Theory and Methodology. North-Holland, New York (1983)
4. Corne, D.W., Knowles, J.D.: Estimation-free leftovers theorems for multiobjective optimisation problems. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (eds.) Evolutionary Multi-Criterion Optimization, pp. 327–341. Springer, Berlin (2003). https://doi.org/10.1007/3-540-36970-8_23
5. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998)
6. Deb, K.: Optimization for Engineering Design: Algorithms and Examples. Prentice-Hall, New Delhi (1995)
7. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2–4), 311–338 (2000)
8. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley, Chichester (2001)
9. Deb, K., Abouhawwash, M.: An optimality theory-based proximity measure for set-based multiobjective optimization. IEEE Trans. Evol. Comput. **20**(4), 515–528 (2016). https://doi.org/10.1109/TEVC.2015.2483590
10. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Syst. **9**(2), 115–148 (1995)
11. Deb, K., Anand, A., Joshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evolut. Comput. J. **10**(4), 371–395 (2002)
12. Deb, K., Bandaru, S., Seada, H.: Generating uniformly distributed points on a unit simplex for evolutionary many-objective optimization. In: Proceedings of the Tenth International Conference on Evolutionary Multi-Criterion Optimization (EMO-19), LNCS, vol. 11411, pp. 179–190. Springer, Heidelberg (2019)
13. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Comput. Sci. Inf. **26**(4), 30–45 (1996)
14. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
15. Deb, K., Myburgh, C.: A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. Eur. J. Oper. Res. **261**(2), 460–474 (2017). https://doi.org/10.1016/j.ejor.2017.02.015
16. Deb, K., Padhye, N.: Development of efficient particle swarm optimizers by using concepts from evolutionary algorithms. In: Proceedings of Genetic and Evolutionary Algorithms Conference (GECCO-2010), pp. 55–62 (2010)
17. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
18. Deb, K., do Val Lopes, C.L., Martins, F.V.C., Wanner, E.F.: Identifying pareto fronts reliably using a multi-stage reference-vector-based framework. IEEE Trans. Evolut. Comput. (in press)
19. Deep, K., Mebrathu, H.: Combined mutation operators of genetic algorithm for the travelling salesman problem. Int. J. Combinat. Optim. Probl. Inf. **2**, 2–24 (2011)
20. Dhara, A., Dutta, J.: Optimality Conditions in Convex Optimization: A Finite-Dimensional View. CRC Press, Boca Ratan (2011)

21. Elarbi, M., Bechikh, S., Gupta, A., Ben Said, L., Ong, Y.: A new decomposition-based NSGA-II for many-objective optimization. IEEE Trans. Syst., Man, Cybern.: Syst. **48**(7), 1191–1210 (2018). https://doi.org/10.1109/TSMC.2017.2654301
22. Farina, M., Amato, P.: A fuzzy definition of "optimality" for many-criteria optimization problems. IEEE Trans. Syst., Man, Cybern. - Part A: Syst. Humans **34**(3), 315–326 (2004). https://doi.org/10.1109/TSMCA.2004.824873
23. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, USA (1989)
24. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins, G.J.E. (ed.) Foundations of Genetic Algorithms, vol. 1, pp. 69–93. Elsevier (1991). https://doi.org/10.1016/B978-0-08-050684-5.50008-2
25. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolut. Comput. J. **9**(2), 159–195 (2000)
26. Hardin, D., Saff, E.: Minimal Riesz energy point configurations for rectifiable d-dimensional manifolds. Adv. Math. **193**(1), 174–204 (2005). https://doi.org/10.1016/j.aim.2004.05.006
27. Hingee, K., Hutter, M.: Equivalence of probabilistic tournament and polynomial ranking selection. In: 2008 IEEE Congress on Evolutionary Computation, pp. 564–571 (2008). https://doi.org/10.1109/CEC.2008.4630852
28. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Ann Arbor (1975)
29. Jahn, J.: Vector Optimization. Springer, Berlin (2004)
30. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. IEEE Trans. Evol. Comput. **18**(4), 602–622 (2014). https://doi.org/10.1109/TEVC.2013.2281534
31. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Press (1995)
32. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y., (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings (2015). http://arxiv.org/abs/1412.6980
33. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. J. Assoc. Comput. Mach. **22**(4), 469–476 (1975)
34. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
35. Padhye, N., Bhardawaj, P., Deb, K.: Improving differential evolution through a unified approach. J. Global Optim. **55**, 771–799 (2013)
36. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005)
37. Purshouse, R.C., Fleming, P.J.: On the evolutionary optimization of many conflicting objectives. IEEE Trans. Evol. Comput. **11**(6), 770–784 (2007)
38. Rao, S.S.: Engineering Optimization Theory and Practice. Wiley, USA (2019)
39. Reklaitis, G.V., Ravindran, A., Ragsdell, K.M.: Engineering Optimization Methods and Applications. Wiley, New York (1983)
40. Saxena, D.K., Kapoor, S.: On timing the nadir-point estimation and/or termination of reference-based multi- and many-objective evolutionary algorithms. In: Deb, K., Goodman, E., Coello Coello, C.A., Klamroth, K., Miettinen, K., Mostaghim, S., Reed, P., (eds.) Evolutionary Multi-Criterion Optimization, pp. 191–202. Springer International Publishing, Cham (2019)
41. Saxena, D.K., Mittal, S., Kapoor, S., Deb, K.: A localized high-fidelity-dominance based many-objective evolutionary algorithm. IEEE Trans. Evolut. Comput. 1 (2022). https://doi.org/10.1109/TEVC.2022.3188064
42. Saxena, D.K., Sinha, A., Duro, J.A., Zhang, Q.: Entropy-based termination criterion for multi-objective evolutionary algorithms. IEEE Trans. Evol. Comput. **20**(4), 485–498 (2016). https://doi.org/10.1109/TEVC.2015.2480780
43. Schröder, B.S.W.: Ordered Sets: An Introduction. Birkhäuser, Boston (2003)
44. Schwefel, H.P.: Collective phenomena in evolutionary systems. In: Checkland, P., Kiss, I. (eds.) Problems of Constancy and Change - the Complementarity of Systems Approaches to

Complexity, pp. 1025–1033. International Society for General System Research, Budapest (1987)
45. Si, C., Shen, J., Wang, L.: On the constraint normalization: A further empirical study. In: 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 967–972 (2018). https://doi.org/10.1109/YAC.2018.8406511
46. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation and Application. Wiley, New York (1986)
47. Storn, R., Price, K.: Differential evolution - a fast and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
48. Talukder, A.K.A., Deb, K.: Paletteviz: a visualization method for functional understanding of high-dimensional Pareto-optimal data-sets to aid multi-criteria decision making. IEEE Comput. Intell. Mag. **15**(2), 36–48 (2020). https://doi.org/10.1109/MCI.2020.2976184
49. Wang, R., Zhou, Z., Ishibuchi, H., Liao, T., Zhang, T.: Localized weighted sum method for many-objective optimization. IEEE Trans. Evol. Comput. **22**(1), 3–18 (2018). https://doi.org/10.1109/TEVC.2016.2611642
50. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T., (eds.) Multiple Criteria Decision Making Theory and Application, pp. 468–486. Springer, Berlin (1980). https://doi.org/10.1007/978-3-642-48782-8_32
51. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893
52. Yuan, Y., Xu, H., Wang, B., Yao, X.: A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. **20**(1), 16–37 (2016). https://doi.org/10.1109/TEVC.2015.2420112
53. Zhu, S., Xu, L., Goodman, E.D., Lu, Z.: A new many-objective evolutionary algorithm based on generalized pareto dominance. IEEE Trans. Cybern. **52**(8), 7776–7790 (2022). https://doi.org/10.1109/TCYB.2021.3051078

Chapter 3
Foundational Studies on ML-Based Enhancements

Many efficient evolutionary multi- and many-objective optimization algorithms, jointly referred to as EMâOAs, have been proposed in the last three decades. However, while solving complex real-world problems, EMâOAs that rely only on natural variation and selection operators may not produce an efficient search [14, 33, 45]. Therefore, it may be desirable or essential to enhance the capabilities of EMâOAs by introducing synergistic concepts from probability, statistics, machine learning (ML), etc. This chapter highlights some of the key studies that have laid the foundations for ML-based enhancements for EMâOAs and inspired further research that has been shared in subsequent chapters.

3.1 *Innovization*-Based Enhancements

The term *innovization*, rooted in the merging of innovation and optimization, refers to a two-step procedure involving: (1) invocation of an EMâOA to obtain a set of trade-off solutions for a given problem; and (2) unveiling new, innovative, and important design principles relating to the decision variables and objectives that are common to a subset or the complete set of trade-off solutions. Think of a design optimization problem in which the objectives are to minimize the size of an electric induction motor and simultaneously maximize the power delivered by the motor. Clearly, different combinations of design variables, including armature radius, wire diameter, and number of wiring turns, lead to different trade-off solutions. Possibly, a solution representing a small-sized motor (solution A in Fig. 3.1) may only deliver a few horsepower, just enough to run a pump to lift water to a two-story building. Similarly, a solution representing a high-power motor based on the same technology (solution B in Fig. 3.1), which produces the few hundred horsepower needed to run a compressor in an industrial air conditioning unit, is likely to have substantially larger size and weight. The application of an EMâOA to this problem leads to these two extreme solutions and a number of other intermediate solutions (as in Fig. 3.1)

D. K. Saxena et al., *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization*, Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-2096-9_3

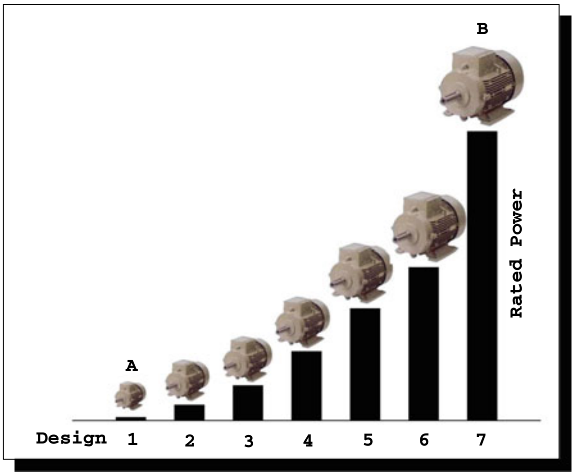


Fig. 3.1 A hypothetical set of trade-off designs showing a conflict between motor size and power delivered in a range of induction motors. Despite the differences, there are similarities in their designs (taken from [16])

with different trade-offs between size and power. Such intermediate solutions may represent motors that can be used in an overhead crane to lift and maneuver a load, motors delivering 50 to 70 horsepower that can be used to run a machining center in a factory, and motors that deliver a few hundred horsepower which can be used in an industrial exhaust fan. Now, if all such motors are lined up as per the worsening order in one of the objectives (say, increasing size), then they would automatically represent an improving order in the other objective (increasing power). Obtaining such a wide variety of solutions in a single EMâOA run is in itself a significant matter, discussed and demonstrated in various EMâO studies [9, 11].

To further enhance the value derived from the computational effort spent on an EMâOA run, innovization seeks to determine important innovative design principles hidden in trade-off solutions, or the recipe for optimality, through a post-optimality analysis, as discussed below. The basis for the postulation that optimal trade-off solutions may encapsulate important design principles (relationships between the decision variables and objective functions) lies in the fact that optimal solutions (X^*) are not arbitrary solutions in the variable space ($\mathcal{X}$ space), rather they are special solutions which satisfy the so-called Fritz John necessary conditions [19, 29, 41], in addition to them being feasible, certain constraint qualification conditions.

$$\sum_{i=1}^{M} \lambda_i \nabla f_i(X^*) + \sum_{j=1}^{J} \lambda_i \nabla g_j(X^*) = 0, \text{ (n-dim vector equation)}, \tag{3.1}$$

$$\mu_j g_j(X^*) = 0, \text{ for all } j = 1, 2, \ldots, J, \tag{3.2}$$

$$\mu_j \geq 0, \text{ for all } j = 1, 2, \ldots, J. \tag{3.3}$$

The fact that these conditions bind the variables to the gradients of the objective and constraint functions makes it natural for them to encapsulate and exhibit these relationships. Depending on the underlying problem, such relationships may hold across the trade-off solutions on the entire Pareto front (PF), or different relationships may characterize different parts of the PF. In either case, such relationships shedding light on the physics of the underlying problem may not be intuitive to the designer, and once revealed, could be utilized subsequently as a set of innovative rules. Given that these innovative relationships are derived through the outcome of a carefully performed optimization task, it can be interpreted as a process of obtaining *innov*ative solution principles through optim*ization*, abbreviated as *innovization* [15]. In the specific context of the induction motor example, it would be interesting to see if all optimal solutions have an identical wire diameter or have an armature diameter proportional to or in some relation to the delivered power. If such a relationship among the design variables and objective values exists and can be captured, it would be of great importance to a designer. With such a recipe, the designer can later design a new motor for a new application without resorting to solving a completely new optimization problem. Moreover, the crucial relationship among design variables and objectives will also provide vital information about the theory of design of a motor, which can bring out limitations and scopes of the existing procedure and spur new and innovative ideas of designing an electric motor.

The manual implementation of the innovization task, in reference to two engineering design problems, is described in the Appendix. For each problem, it involves the following:

1. Discovery of Pareto-optimal solutions using an EMâOA followed by local search (for improved convergence of solutions).
2. Manual analysis of Pareto-optimal solutions to extract innovated principles of a certain structure.

3.1.1 Automated Innovization Procedure

Early studies on innovization used a manual process of plotting variables in pairs to find patterns and relationships between them, making the task tedious. In 2010, a machine learning procedure was proposed [1], in which the trade-off solutions obtained from an EMâOA run were used to automatically extract rules of the pre-specified structure, based on a sophisticated optimization problem formulation. The choice of pre-specified rule structure was motivated by two factors: (i) rules must be easily interpretable, and (ii) rules must be able to reveal direct and inversely proportional properties of variables, as often desired in practical problems. Taking into account this, pre-specified rule structures were restricted to *power laws*, as shown below:

$$\psi_k(X) = \prod_{i=1}^{n} x_i^{b_{ik}} = c_k. \tag{3.4}$$

In that formulation, ψ_k refers to the kth rule, and $b_{ik} \in \mathbb{R}$ is the power of x_i in the kth rule. Notably, the powers b_{ik} and constants c_k are found by solving a nonlinear optimization problem of minimizing the error between the rule-predicted values and the actual variable values in the trade-off solutions obtained from an EMâOA run. In some studies, instead of considering all n variables in a rule, only a few (two or three) variables were allowed. For illustration, the sample rules possible in a two-variable scenario (x_1 and x_2), under the above power law, are as follows:

- $x_1^2 x_2^{-1} = 2$, indicating $x_2 = 0.5x_1^2$, implying a positive correlation between the two variables in the entire Pareto-optimal set (PS).
- $x_1^2 x_2^{0.5} = 1.2$, indicating $x_2 = 1.44/x_1^4$, implying a negative correlation between the variables.

Another interesting possibility is that while the left part of the rule (ϕ_k) may be the same for the entire PS, the right value (c_k) may be different for different subsets of PS. In other words, $x_1^2 x_2^{-1} = 2$ and $x_1^2 x_2^{-1} = 1$ could simultaneously hold for different subsets of the Pareto-optimal solutions. Considering the above, some key features of the automated innovization (AutoInn) procedure can be summarized as below:

- AutoInn is a data-driven approach: The trade-off solutions obtained from an EMâOA run serve as the input data.
- AutoInn is capable of simultaneously finding a family of rules for the same predefined structure, implying multiple c_k (right-hand sides) for the same structure represented by ϕ_k (left-hand side); in such a case, each c_k may correspond to a subset of Pareto-optimal solutions.
- AutoInn is capable of finding multiple rules based on different predefined structures, implying different ϕ_k (left-hand side) based on different combinations of variables. This is challenging in the sense that several combinations of variables are possible.
- It is assumed that the data is noisy, as the trade-off solutions obtained may not be true representatives of the true PS. This may happen if the EMâOA is terminated prematurely or the problem search space is too difficult.

To illustrate the case of finding a family of rules with the same predefined structure (say $x_1^{b_1} x_2^{b_2} = c$), the set of N trade-off solutions obtained from an EMâOA run serve as the input data. Then, a new single-objective optimization problem (SOP) is created, with power-law coefficients (b_1 and b_2) as variables. In that SOP, each solution (b_1-b_2 pair) would lead to N values of c. The objective value for each solution is computed by applying a clustering procedure on the c values; and calculating the sum of clusters found in the c values, the count of c values left unclustered, and the coefficient of variation ($\text{cov}_v^l = \mu_l/\sigma_l$) in each cluster found (depicted in Eq. 3.5). Additionally, a constraint is applied to the variables (b_1 and b_2), to avoid their near-zero values, since $b_1 = b_2 = 0$ can always be a solution to this SOP, leading to $c = 1$. A single-objective genetic algorithm is then used to solve this SOP, to arrive at an optimal b_1-b_2 pair. When the same clustering procedure is applied one last time on the c values obtained using the optimal b_1-b_2 pair, it reveals a single rule or family of

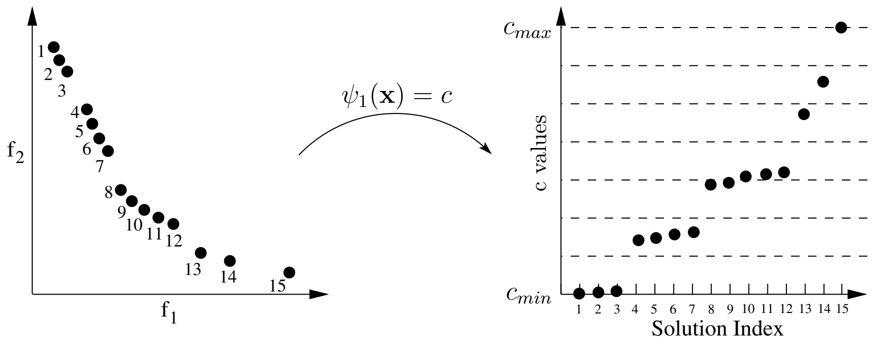


Fig. 3.2 Distribution of c_k-values, indicating three clusters and three unclustered data points

rules depending on whether one or more clusters were found. This procedure, when applied to a sample dataset of 15 trade-off solutions as shown on the left side of Fig. 3.2, leads to 15 values of c as shown on the right side of Fig. 3.2. As is evident, three clusters are identified, each representing a different rule (different values of c) within the same family of rules (same values of b_1 and b_2). In this background, the results on a few engineering design problems are presented in the appendix.

$$\begin{aligned} &\text{Minimize} \quad \left(\#\text{Clusters} + \text{Unclustered\_points} + \textstyle\sum_l \text{cov}_v^l\right), \\ &\text{Subject to} \quad b_{\min} \le |b_{ik}| \le 1, \quad \text{for all } i \text{ and } k. \end{aligned} \tag{3.5}$$

3.1.2 *Innovized Repair (IR) Operator*

Manual and automated innovation procedures find variable relationships preserved within variable vectors of the *PS*. They can also be used to influence subsequent iterations of an EMâOA. The existing *innovized repair* (IR) operator [22, 23] is discussed here in brief.

The main idea is that innovized principles (of power-law structure) can be extracted from non-dominated solutions in intermittent generations of an EMâOA run. Unlike the AutoInn procedure, here the power-law rules are generated using log-linear modeling, followed by multivariate linear regression [23]. The inter-variable relationships (in terms of rules) thus generated can then be used to *repair* the offspring created by the genetic operators (referred to as *natural offspring*) in an EMâOA generation. For instance:

- If a rule involving a single variable x_i happens to be $x_i = c$, then the variable x_i of a natural offspring can be repaired, by picking a random number from a bounded uniform distribution given as follows:

$$\hat{x}_i = \mathcal{U}(\mu_i - \sigma_i, \mu_i + \sigma_i), \tag{3.6}$$

where μ_i and σ_i are the mean and standard deviation of the variable x_i in the parent population.

- If a rule involving two or more variables is given by $x_1^{b_1} x_3^{b_3} = c$, then the dependent variables x_1 or x_3 of an offspring can be repaired (assuming that x_3 or x_1 are independent, respectively) according to the equation below.

$$\hat{x}_1 = \left(\hat{c}/x_3^{\hat{b}_3}\right)^{1/\hat{b}_1}, \qquad \hat{x}_3 = \left(\hat{c}/x_1^{\hat{b}_1}\right)^{1/\hat{b}_3}. \tag{3.7}$$

It has also been observed that the best strategy is to randomly pick one of the variables as the independent variable. For rules with more than two variables, a similar logic can be followed. Such repairs are made to all the natural offspring in the current EMâOA generation, one by one, and the repaired offspring thus created replace their respective original natural offspring. Note the following important aspects of repairing an offspring [23]:

1. Following each variable repair, if the repaired variable value lies outside its original bounds as defined in the given multi- or many-objective optimization problem (MOP/MaOP), then the repaired value is replaced by its nearest bound.
2. It is plausible that a variable may simultaneously be a part of multiple rules. In such scenarios, while repairing the offspring, a particular variable is only allowed to be repaired once using only one of the rules obtained. For example, given two rules $x_1x_2 = 2$ and $x_1x_3 = 3$, if the first rule is used to repair x_1 in a particular offspring, then variable x_1 is not considered again for repair, leaving x_3 to be repaired using the second rule.
3. In scenarios where multiple rules of different lengths may exist, a preference is given to shorter rules for repair, as the corresponding repair is found to be relatively more effective.

However, before any meaningful rules can be extracted, adequate convergence or stability of the non-dominated dataset may be required. In EMâOA-IR, an EMâOA integrated with the IR operator, no learning or repair is executed until 33% of $t_{\max}$ have passed, where $t_{\max}$ is the pre-specified number of generations after which an algorithmic run of EMâOA-IR is terminated. In each subsequent generation, the learned mathematical relationships are applied to repair offspring solutions created by the genetic operators. In this background, the results on a few engineering design problems are presented in the appendix of this chapter.

3.2 Surrogate Modeling

ML methods have often been used in the EMâO domain for surrogate modeling. The overall idea is to learn the variable–objective relationships locally, using an ML method (called a surrogate model), and evolve the solutions on the basis of approximate function values (through the surrogate model). Repeated use of this procedure during an optimization run may require fewer actual function evaluations in order to converge. Despite the high computational complexity of building surrogate models, this approach could be useful in real-world problems where the actual function evaluation can be computationally very expensive [3, 8, 18].

Although the overarching idea of ML-based surrogate modeling in EMâO is highlighted above, it is incomplete without taking into account the evaluation of constraint functions, which could also be computationally expensive. Given this, the function and constraint evaluations are collectively referred to as *solution evaluations* in this book. There are several approaches, where each objective function and constraint function is modeled *independently* from the solutions already evaluated, using different ML methods, such as Gaussian random field models [21]; Gaussian regression models [20]; Kriging [30, 48]; radial basis functions [44]; random forests [49]; response surfaces [34]; and support vector regression [28].

In a slight departure, [13] discussed the possibility of building a single *collective* surrogate model for all objective functions using scalarizing functions, such as weighted sum or Tchebychev [41]; and separately building a single surrogate model for all constraint functions using a combined normalized constraint violation function [12]. In a balancing act, [27] proposed a framework that could switch between independent and collective surrogate models in an adaptive manner.

In addition to facilitating approximate solution evaluations, surrogate models have also been used to perform other tasks in EMâO, including local search [31, 56] and efficient offspring creation [32, 37]. Next, the surrogate-model-based local search is discussed, followed by the surrogate-based efficient offspring creation.

Performing a local search in intermediate generations of an EMâOA run requires additional solution evaluations beyond the usual evaluations of offspring in each generation. To reduce this additional computational burden, some studies use a surrogate model to perform the local search using approximate solution evaluations [35, 36]. In these studies, a structural hill climber (for local search) had been combined with the BOA [46], where the search neighborhoods are defined by the inter-variable dependencies learned by the probabilistic model of BOA. Despite these efforts, the choice between using a surrogate model for approximate solution evaluations and using actual solution evaluations was inconclusive, as different dynamics were observed for problems with different characteristics [36].

Some methods have also used surrogate models for efficient offspring creation. In that work, [37] relies on the generation of multiple offspring from the same set of parents (in each generation) using different mating strategies. This, in general, may necessitate extra solution (offspring) evaluations compared to the conventional scenario where only one/two offspring are generated from a set of parents. To limit the additional computational cost, the fitness of the latter is approximated using a surrogate model instead of the potentially expensive actual solution evaluations. This challenge of extra solution evaluations is also manifested in another similar method [32], where in addition to the parents in any generation, some extra solutions are created that could serve as potential parents. Again, the evaluation of the fitness of such solutions is based on surrogate models rather than potentially expensive actual evaluations.

3.3 Model-Based Offspring Sampling

EMâOAs with model-based offspring sampling began with the development of Estimation of Distribution Algorithms (EDAs) in 1996 [43]. EDAs were designed to directly extract the statistical information from the global search space from the current search and build a *probabilistic model* of *elite* solutions, using ML methods such as Bayesian networks or decision trees. This model would then be used to create new offspring solutions (through sampling) for subsequent generation(s). EDAs, such as BOA [46], MONEDA [38], RM-MEDA [52], EDA-VNS [17] and HMOBEDA [39], have shown a potentially distinctive advantage of exploiting inter-variable dependencies in creating new offspring solutions. One of the significant extensions to EDAs includes FEG-EDA [51], that (i) maps the original search space to a modified search space, (ii) samples new offspring solutions in the modified search space, and (iii) reverts them back to the original search space, in an attempt to capture the inter-variable dependencies better. Reportedly, there are two major issues with model-based sampling, including the need to choose: (a) the selection strategy for elite solutions and (b) the building strategy for the probability distribution model [6].

Given that sampling through these probabilistic models embeds the captured inter-variable dependencies into the offspring solutions, their advantage is tangible when there are inter-variable dependencies present in the *PF* solutions, for a given MOP/MaOP. However, there are other MOPs/MaOPs where independent mating through EMâO's natural variation operators produces a more efficient search [42]. In this background, several EMâOAs have been proposed that create offspring partially through natural variation operators and partially through model-based sampling, including M-MOEA [55], Hybrid [54], IM-MOEA [7], and GMOEA [25].

3.4 Efficient Mutation

In EMâOAs, the mutation is one of the natural variation operators, which is used primarily as a mechanism to maintain diversity in the population [26]. Although the mutation operator alone may not produce an effective search, it plays a crucial role along with a suitable recombination operator, making the overall search efficient [24]. Traditionally, mutation operators modify one or more variables in a given solution with *probability*, while the extent of that modification is controlled by an *index*. To avoid the fixation of these parameters a priori, which can deter an efficient search, some studies have chosen to integrate reinforcement learning (RL) into EMâOAs. One such study is NSGA-RL [4], where suitable mutation parameters are learned on-the-fly using RL, individually for each variable, toward a more efficient search. Another such study is RL-NSGA-II [47], where the RL algorithm attempts to learn and decide which variable(s) of a given solution should be modified, while controlling the extent of that modification through a randomized input.

In addition to the above, some studies have used the learning automaton (LA), a variant of RL, to guide the mutation in a more comprehensive manner [10, 53]. In these instances, [10] learns two probabilities individually for each variable, corresponding to the direction of modification of that variable (toward the lower or upper bound) and the extent of that modification. Alternatively, [53] takes advantage of the reference vector (RV)-based structure of the underlying RV-EMâOA, by learning the choice of mutation operator (of three), individually for each RV.

Unlike the above approaches that attempted to learn and adapt the mutation operation to a more efficient search, a recent study [50] attempted to learn and adapt the search space itself, using PCA. In that work, the parent solutions are first mapped to a transformed (reduced) search space built through the application of PCA, and the new offspring solutions are generated through mutation in the transformed search space. Finally, the new offspring solutions are mapped back to the original search space prior to their evaluation. Notably, the PCA-assisted mutation is applied only after a fixed proportion of the maximum generations allowed ($t_{\max}$).

3.5 Summary

In this chapter, some studies related to innovization and online innovization have been discussed. Innovization refers to the task of revealing innovative and important design rules hidden in the trade-off (Pareto-optimal) solutions, also being referred to as the recipe for optimality. In that, manual and automated innovization procedures have been discussed. While the former entails a series of steps that a user can manually execute, the latter involves solving an SOP, to arrive at the design rules. While innovization refers to a post-optimal analysis, online innovization includes revealing the design rules in any intermediate generation of an EMâOA run and using those rules within the same run to repair the offspring solutions. To this end, the innovized repair operator has been discussed. Furthermore, some other ML-based enhancements in EMâOAs have been discussed, including the use of surrogate models, model-based offspring sampling, and efficient mutation operators. This chapter also carries an Appendix, which shares some solved examples on manual innovization, automated innovization, and the innovized repair operator.

Appendices for in this Chapter

The following sections discuss examples of the manual innovization task, the automated innovization task, and the innovized repair operator.

3.6 Examples of Manual Innovization Task

The basic working of the manual innovization procedure has been illustrated here through a three-variable, two-objective truss design problem, which was originally studied using the ϵ-constraint approach [5, 41] and later using an evolutionary approach [11]. In that, the truss (Fig. 3.3) must carry a certain load without incurring an elastic failure. The two conflicting design objectives are to (i) minimize the total volume of the truss members and (ii) minimize the maximum stress developed in both members (AC and BC) due to the application of the load 100 kN. Furthermore, the three decision variables are cross-sectional area of AC and BC (x_1 and x_2, respectively), measured in square meters, and the vertical distance between A (or B) and C (y), measured in meters. The optimization problem formulation is given as follows:

$$\begin{aligned} \text{Minimize} \quad & f_1(\vec{x}, y) = x_1\sqrt{16+y^2} + x_2\sqrt{1+y^2}, \\ \text{Minimize} \quad & f_2(\vec{x}, y) = \max(\sigma_{AC}, \sigma_{BC}), \\ \text{Subject to} \quad & \max(\sigma_{AC}, \sigma_{BC}) \le S_{\max}, \\ & 0 \le x_1, x_2 \le A_{\max}, \\ & 1 \le y \le 3. \end{aligned} \tag{3.8}$$

Using the dimensions and loading specified in Fig. 3.3, it can be observed that member AC is subjected to $20\sqrt{16+y^2}/y$ kN load and member BC is subjected to $80\sqrt{1+y^2}/y$ kN load. The stress values are calculated as follows:

$$\sigma_{AC} = \frac{20\sqrt{16+y^2}}{yx_1}, \quad \sigma_{BC} = \frac{80\sqrt{1+y^2}}{yx_2}. \tag{3.9}$$

Here, the stress values and the cross-sectional areas are limited to $S_{\max} = 1(10^5)$ kPa and $A_{\max} = 0.01$ m^2, respectively. All three variables are treated as real-valued. Simulated binary crossover (SBX) with $\eta_c = 10$ and polynomial mutation operator with

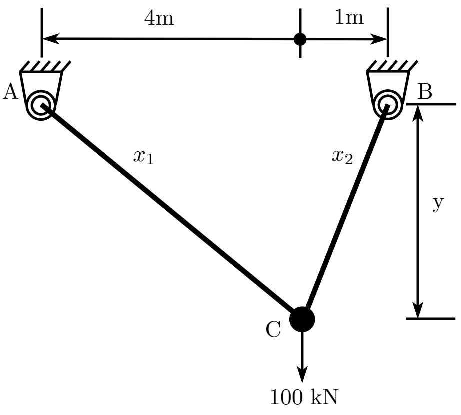


Fig. 3.3 The design of two-bar truss (taken from [16])

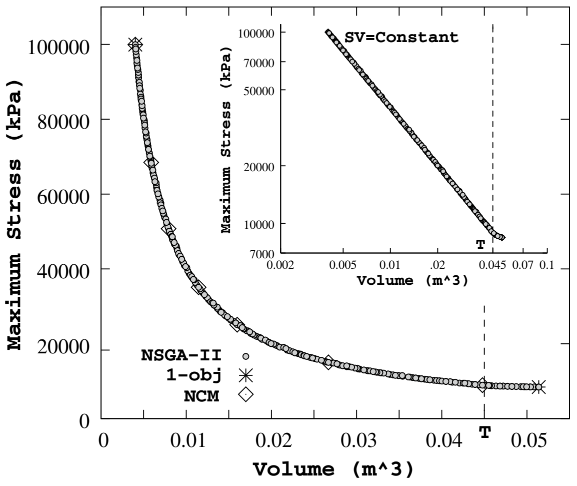


Fig. 3.4 NSGA-II solutions obtained for the two-bar truss problem (taken from [16])

$\eta_m = 50$ have been used. All constraints are handled using the constraint-tournament approach [11]. Figure 3.4 shows the final set of non-dominated solutions obtained by an algorithmic run of NSGA-II. Although the trade-off between the two objectives is evident in Fig. 3.4, these solutions are further analyzed, using two different studies, to gain more confidence in the Pareto-optimality of these solutions. First, a single-objective RGA is used to find the optimum of individual objective functions, subject to the same constraints and variable bounds. Figure 3.4 marks these two solutions (one per objective) as 1-obj solutions. It is evident that the front obtained using NSGA-II extends to these two extreme solutions. Next, the normal constraint method (NCM) [40] is used with different starting points from a line that joins the two extreme solutions. The solutions thus obtained, one at the end of each NCM procedure, are also shown in Fig. 3.4. Since these solutions lie on the front obtained using NSGA-II, it is confirmed that the non-dominated solutions obtained using NSGA-II are close to the true PF.

3.6.1 Theoretical Innovized Principles and Manual Innovization Results

Before applying the manual innovization procedure to the solutions obtained using NSGA-II, an exact analysis of this problem is presented to identify the true PF, and the underlying innovized principles (theoretical), if any. The problem, although simple mathematically, is a typical optimization problem that has two resource terms in each objective, involving variables x_1 and x_2, and interlinking them with the third variable y. For such problems, the optimum occurs when identical resource allocation is made between the two terms in both objective and constraint functions, as shown below.

$$x_1\sqrt{16+y^2} = x_2\sqrt{1+y^2} \implies \frac{20\sqrt{16+y^2}}{yx_1} = \frac{80\sqrt{1+y^2}}{yx_2}. \tag{3.10}$$

Thus, every optimal solution is expected to satisfy both of the above equations, resulting in the following innovated rules:

$$\frac{x_1}{x_2} = 0.5, \text{ and } y = 2. \tag{3.11}$$

Substituting $y = 2$ into the expression for the first objective (volume) leads to $x_2 = V/2\sqrt{5}$ m^2, where V is the volume of the structure (in m^3). Similarly, substituting these values into the objective functions $V = f_1$ and $S = f_2$ leads to $SV = 400$—an inverse relationship between the objectives. Thus, the solutions in the true *PF* are given in terms of volume V, as follows:

$$x_1^* = \frac{V^*}{4\sqrt{5}} \text{ m}^2, \quad x_2^* = \frac{V^*}{2\sqrt{5}} \text{ m}^2, \quad y^* = 2 \text{ m}, \quad S^* = 400/V^* \text{ kPa}.$$

When the variable x_2 reaches its upper bound, that is, at the transition point T shown in Fig. 3.5, $V_T = 0.04472$ m^3 and $S_T = 8944.26$ kPa, since x_2 cannot be increased any further. The inset plot (drawn with a logarithmic scale of both axes) in Fig. 3.4 shows this interesting aspect of the front obtained. There are two distinct behaviors around the transition point T marked in the figure: (i) one that stretches from the smallest volume solution to a volume of about 0.04478 m^3 (point T), and (ii) another that stretches from this transition point to the smallest stress solution.

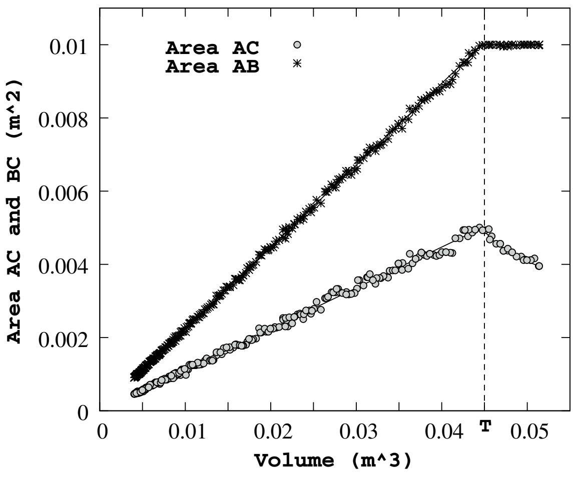


Fig. 3.5 Variation of x_1 and x_2 for the truss design problem (taken from [16])

Table 3.1 Two extreme solutions and an interesting intermediate solution (T) for the two-bar truss design problem are presented (taken from [16])

Solution	x_1 (m^2)	x_2 (m^2)	y (m)	f_1 (m^3)	f_2 (kPa)
Min. volume	4.60(10^{-4})	9.05(10^{-4})	1.935	0.004013	99,937.031
Intermediate (T)	49.30(10^{-4})	99.89(10^{-4})	2.035	0.044779	8,945.610
Min. of max. stress	39.54(10^{-4})	100.00(10^{-4})	3.000	0.051391	8,432.740

The extreme solutions and this intermediate solution, obtained by NSGA-II, are tabulated in Table 3.1.

An investigation of the values of the decision variables reveals the following:

1. The inset plot in Fig. 3.4 reveals that for optimal structures, the maximum stress (S) developed is inversely proportional to the volume (V) of the structure, that is, $SV =$ constant, as predicted above. When a straight line is fitted through the logarithm of the two objective values, $SV = 402.2$, a relationship is found between these solutions obtained using NSGA-II. The obtained relationship is close to the theoretical relationship computed above (from the true PF).
2. The inset plot also reveals that the transition occurs at $V = 0.044779$ m^3, which is also close to the exact theoretical value computed above.
3. To achieve an optimal solution with a lower maximum stress (and larger volume), both cross-sectional areas (AC and BC) should increase linearly with volume, as shown in Fig. 3.5. The figure also plots the mathematical relationships (x_1 and x_2 versus V) obtained earlier with solid lines, which can barely be seen, as the solutions obtained using NSGA-II fall on top of these lines.
4. A further investigation reveals that the ratio between these two cross-sectional areas is almost 1:2, and the vertical distance (y) takes a value close to 2 for all solutions.
5. Figure 3.6 reveals that the stress values on both members (AC and BC) are identical for any Pareto-optimal solution (Fig. 3.7).

The innovized rules illustrated above are some interesting properties of the original optimization problem that may not be intuitive to the designer. However, these principles can be explained from the mathematical formulation described above. Thus, although these optimality conditions can be derived mathematically from the problem formulation given in Eq. 3.8 in this simple problem, such optimality conditions may often be tedious and difficult to achieve exactly for large and complex problems. The application of a numerical optimization procedure and then investigating the obtained optimal solutions have the potential to reveal such important innovative design principles.

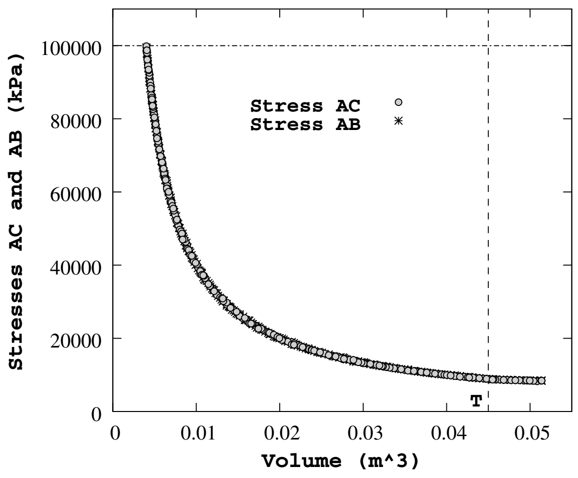


Fig. 3.6 Variation of stresses in AC and BC of the two-bar truss problem (taken from [16])

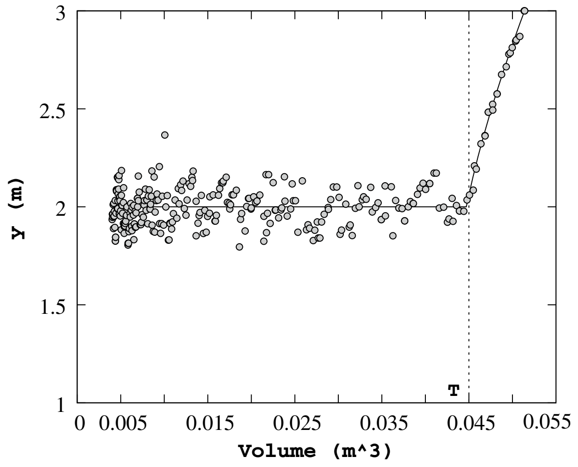


Fig. 3.7 Variation of y for the two-bar truss design problem (taken from [16])

3.7 Examples of Automated Innnovization Task

The same two-bar truss problem, discussed above, is chosen to illustrate the working of the AutoInn procedure. For the solutions obtained using NSGA-II, the AutoInn procedure finds four rules common to 87% to 92% of the non-dominated dataset:

$$SV = 400.770, \quad \frac{x_1}{V} = 0.111, \quad \frac{x_2}{V} = 0.224, \quad \frac{x_2}{x_1} = 1.984. \tag{3.12}$$

Figure 3.8 shows the relevant non-dominated solutions obtained using NSGA-II. Some unclustered solutions are marked as red points. Figure 3.9 shows the distribution of c_k values for one of the rules obtained $V^{-0.997}x_1^{1.000} = c$. It is clear that

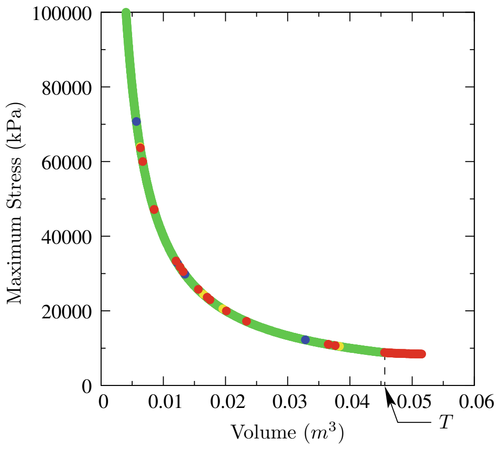


Fig. 3.8 Pareto front for the two-bar truss design problem. Red points are a few unclustered points for the $V^{-0.997}x_1^{1.000} = c$ rule

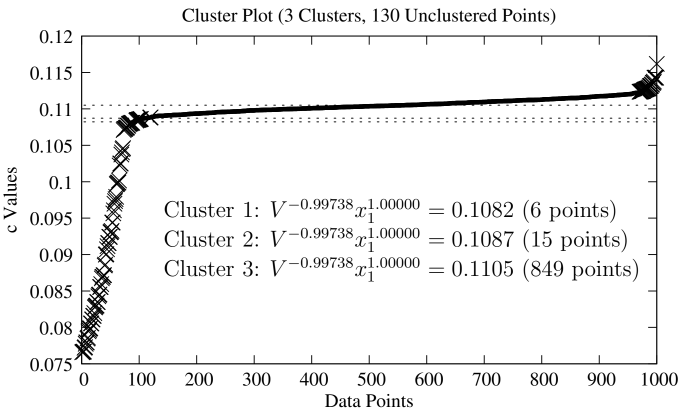


Fig. 3.9 c_k distribution for the rule $V^{-0.997}x_1^{1.000} = c$ is shown, found to 87% of the non-dominated dataset. Unclustered non-dominated data are shown with a 'X' (taken from [2])

the values V and x_1 of the majority (87%) non-dominated solutions satisfy the rule. The clustering algorithm inbuilt in the AutoInn procedure found three clusters with slightly different c_k-values. But the non-dominated solutions that do not satisfy the rule have very different c_k values. For ease of understanding, the c_k-values are sorted from low to high in the figure shown.

The respective distributions of the c_k values for two other rules are shown in Figs. 3.10 and 3.11.

Although the AutoInn procedure finds multiple clusters, the respective c_k values are close to each other, and the difference in the c values from the unclustered points is significant.

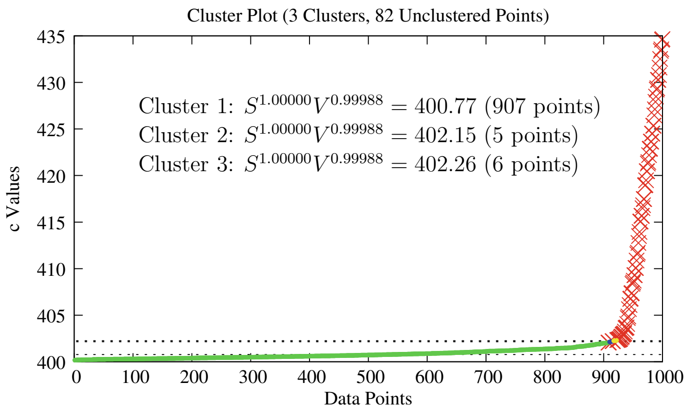


Fig. 3.10 c-value distribution for $S^{1.0000}V^{0.9999} = c$ found to 92% of the non-dominated dataset

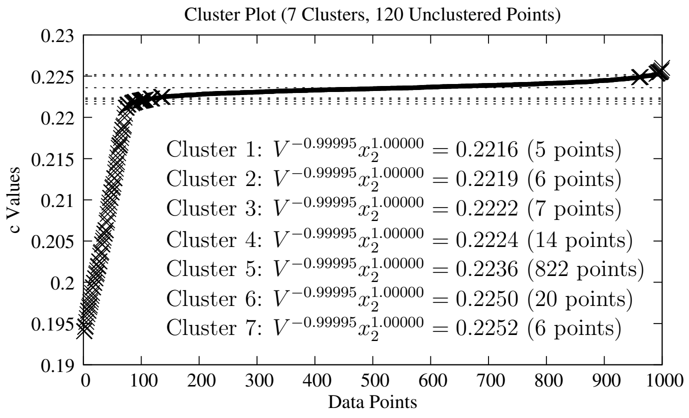


Fig. 3.11 Cluster plot for $V^{-0.9999}x_2^{1.0000} = c$ found to 88% of the non-dominated dataset (taken from [2])

3.8 Examples of Innovized Repair Operator

Fig. 3.12 shows the median generational distance (GD) and inverse generational distance (IGD) metrics [11] for the two-bar truss design problem. Notably, GD is an indicator of convergence, and IGD is a combined indicator of convergence and diversity. The plots in Fig. 3.12 reveal that NSGA-II-IR with repair preference given to short rules (SN repair strategy) performs much better than the no repair strategy (NI, i.e., base NSGA-II), in terms of GD (smaller the better). However, in terms of the IGD metric, the NI strategy performs marginally better. This is expected, as the NSGA-II with SN repair strategy is expected to focus more on improving the convergence than on maintaining the diversity.

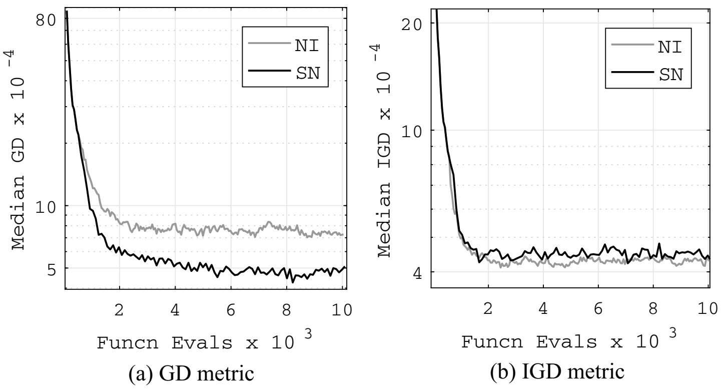


Fig. 3.12 Median GD and IGD results for the two-bar truss design problem over 30 runs

The rules extracted at the end of the NSGA-II run with the SN strategy, provided below, closely match the theoretical property of variables stated in Eq. 3.11:

$$x_1^{-1.008}x_2 = 2.0750, \quad x_3 = 1.9449 \pm 0.0674. \tag{3.13}$$

References

1. Bandaru, S., Deb, K.: Automated discovery of vital knowledge from Pareto-optimal solutions: first results from engineering design. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010). https://doi.org/10.1109/CEC.2010.5586501
2. Bandaru, S., Deb, K.: Towards automating the discovery of certain innovative design principles through a clustering based optimization technique. Eng. Optim. **43**(9), 911–941 (2011)
3. Bhattacharjee, K.S., Isaacs, A., Ray, T.: Multi-objective optimization using an evolutionary algorithm embedded with multiple spatially distributed surrogates. In: Multi-objective Optimization, pp. 135–155. World Scientific (2017). https://doi.org/10.1142/9789813148239_0005
4. Bora, T.C., Mariani, V.C., dos Santos Coelho, L.: Multi-objective optimization of the environmental-economic dispatch with reinforcement learning based on non-dominated sorting genetic algorithm. Appl Thermal Eng **146**, 688–700 (2019). https://doi.org/10.1016/j.applthermaleng.2018.10.020
5. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making Theory and Methodology. North-Holland, New York (1983)
6. Chen, Y., Zhang, Y., Abraham, A.: Estimation of distribution algorithm for optimization of neural networks for intrusion detection system. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.), Artificial Intelligence and Soft Computing—ICAISC 2006. ICAISC 2006. Lecture Notes in Computer Science, vol. 4029. Springer, Berlin, Heidelberg (2006)
7. Cheng, R., Jin, Y., Narukawa, K., Sendhoff, B.: A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling. IEEE Trans. Evol. Comput. **19**(6), 838–856 (2015). https://doi.org/10.1109/TEVC.2015.2395073
8. Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Trans. Evol. Comput. **22**(1), 129–142 (2018)

9. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-objective Problems. Springer, New York (2007)
10. Dai, C., Wang, Y., Ye, M., Xue, X., Liu, H.: An orthogonal evolutionary algorithm with learning automata for multiobjective optimization. IEEE Trans. Cybern. **46**(12), 3306–3319 (2016). https://doi.org/10.1109/TCYB.2015.2503433
11. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
12. Deb, K., Datta, R.: Hybrid evolutionary multi-objective optimization and analysis of machining operations. Eng. Optim. **44**(6), 685–706 (2012). https://doi.org/10.1080/0305215X.2011.604316
13. Deb, K., Hussein, R., Roy, P.C., Toscano-Pulido, G.: A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization. IEEE Trans. Evol. Comput. **23**(1), 104–116 (2019). https://doi.org/10.1109/TEVC.2018.2828091
14. Deb, K., Myburgh, C.: A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. European J. Oper. Res. **261**(2), 460–474 (2017). https://doi.org/10.1016/j.ejor.2017.02.015
15. Deb, K., Srinivasan, A.: Innovization: innovating design principles through optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 1629–1636. Association for Computing Machinery (ACM), New York, NY, USA (2006)
16. Deb, K., Srinivasan, A.: Innovization: discovery of innovative design principles through multi-objective evolutionary optimization. In: Knowles, J., Corne, D., Deb, K. (eds.) Multiobjective Problem Solving from Nature: From Concepts to Applications, pp. 243–262. Springer, Berlin (2008)
17. Du, Y., Li, J.Q., Luo, C., Meng, L.L.: A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations. Swarm Evol. Comput. **62**, 100–861 (2021). https://doi.org/10.1016/j.swevo.2021.100861
18. Dutta, S., Gandomi, A.H.: Surrogate model-driven evolutionary algorithms: theory and applications. In: Evolution in Action: Past, Present and Future: A Festschrift in Honor of Erik D. Goodman, pp. 435–451. Springer International Publishing, Cham (2020)
19. Ehrgott, M.: Multicriteria Optimization. Springer, Berlin, Heidelberg (2005)
20. El-Beltagy, M.A., Nair, P.B., Keane, A.J.: Metamodelling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999), pp. 196–203. San Mateo, CA: Morgan Kaufman (1999)
21. Emmerich, M., Giannakoglou, K.C., Naujoks, B.: Single and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Trans. Evol. Comput. **10**(4), 421–439 (2006)
22. Gaur, A., Deb, K.: Adaptive use of *innovization* principles for a faster convergence of evolutionary multi-objective optimization algorithms. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO '16 Companion, pp. 75–76. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2908961.2909019
23. Gaur, A., Deb, K.: Effect of size and order of variables in rules for multi-objective repair-based *innovization* procedure. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2177–2184 (2017). https://doi.org/10.1109/CEC.2017.7969568
24. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, USA (1989)
25. He, C., Huang, S., Cheng, R., Tan, K.C., Jin, Y.: Evolutionary multiobjective optimization driven by generative adversarial networks (GANs). IEEE Trans. Cybern. **51**(6), 3129–3142 (2021). https://doi.org/10.1109/TCYB.2020.2985081
26. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Ann Arbor, MI (1975)
27. Hussein, R., Roy, P., Deb, K.: Switching between metamodeling frameworks for efficient multi-objective optimization. In: IEEE Symposium Series on Computational Intelligence (SSCI-2018), pp. 1–8. IEEE Press, Piscatway, NJ (2018)

28. Inapakurthi, R.K., Mitra, K.: Optimal surrogate building using SVR for an industrial grinding process. Mater. Manuf. Proc. **37**(15), 1701–1707 (2022). https://doi.org/10.1080/10426914.2022.2039699
29. Jahn, J.: Vector Optimization. Springer, Berlin, Germany (2004)
30. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**(4), 345–383 (2001)
31. Koçer, H.G., Uymaz, S.A.: A novel local search method for LSGO with golden ratio and dynamic search step. Soft Comput. **25**, 2115–2130 (2021). https://doi.org/10.1007/s00500-020-05284-x
32. Li, F., Gao, L., Shen, W., Cai, X., Huang, S.: A surrogate-assisted offspring generation method for expensive multi-objective optimization problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2020). https://doi.org/10.1109/CEC48606.2020.9185691
33. Li, L., Chen, H. et al.: C.L.: A robust hybrid approach based on estimation of distribution algorithm and support vector machine for hunting candidate disease genes. Sci. World J. **2013**(393570), 7 (2013). https://doi.org/10.1155/2013/393570
34. Lian, Y., Liou, M.S.: Multiobjective optimization using coupled response surface model and evolutionary algorithm. AIAA J. **43**(6) (2005)
35. Lima, C., Pelikan, M., Lobo, F., Goldberg, D.: Loopy substructural local search for the bayesian optimization algorithm. In: Stützle T., Birattari M., Hoos H.H. (eds.), Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. SLS 2009. Lecture Notes in Computer Science, vol. 5752. Springer, Berlin, Heidelberg (2009)
36. Lima, C., Pelikan, M., Sastry, K., Butz, M., Goldberg, D., Lobo, F.: Substructural neighborhoods for local search in the bayesian optimization algorithm. In: Runarsson T.P., Beyer HG., Burke E., Merelo-Guervós J.J., Whitley L.D., Yao X. (eds) Parallel Problem Solving from Nature—PPSN IX. Lecture Notes in Computer Science, vol. 4193. Springer, Berlin, Heidelberg (2006)
37. Mallipeddi, R., Lee, M.: Surrogate model assisted ensemble differential evolution algorithm. In: 2012 IEEE Congress on Evolutionary Computation, pp. 1–8 (2012). https://doi.org/10.1109/CEC.2012.6256479
38. Martí, L., García, J., Berlanga, A., Molina, J.M.: Introducing MONEDA: scalable multiobjective optimization with a neural estimation of distribution algorithm. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08, pp. 689–696. Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1389095.1389230
39. Martins, M.S.R., Yafrani, M.E., Delgado, M., Lüders, R., Santana, R., Siqueira, H.V., Akcay, H.G., Ahiod, B.: Analysis of bayesian network learning techniques for a hybrid multi-objective bayesian estimation of distribution algorithm: a case study on MNK landscape. J. Heuristics **27**, 549–573 (2021). https://doi.org/10.1007/s10732-021-09469-x
40. Messac, A., Mattson, C.A.: Normal constraint method with guarantee of even representation of complete Pareto frontier. AIAA J. **42**(10), 2101–2111 (2004). https://doi.org/10.2514/1.8977
41. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
42. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: A learning-based *innovized* progress operator for faster convergence in evolutionary multi-objective optimization. ACM Trans. Evol. Learn. Optim. **2**(1) (2021). https://doi.org/10.1145/3474059
43. Miuhlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. binary parameters. In: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, pp. 178–187. London, UK (1996)
44. Mullur, A.A., Messac, A.: Metamodeling using extended radial basis functions: a comparative approach. Eng. Comput. **21**(203) (2006). https://doi.org/10.1007/s00366-005-0005-7
45. Pelikan, M., Goldberg, D., Lobo, F.: A survey of optimization by building and using probabilistic models. Comput. Optim. Appl. **21**(1), 5–20 (2002). https://doi.org/10.1023/A:1013500812258
46. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The bayesian optimization algorithm. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation—Volume 1, GECCO'99, pp. 525–532. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)

47. Ren, Q., Luo, F., Ding, W., Lu, H.: An improved NSGAII algorithm based on site-directed mutagenesis method for multi-objective optimization. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 176–181 (2019). https://doi.org/10.1109/SSCI44817.2019.9002847
48. Sinha, A., Bedi, S., Deb, K.: Bilevel optimization based on kriging approximations of lower level optimal value function. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8 (2018). https://doi.org/10.1109/CEC.2018.8477763
49. Wang, H., Jin, Y.: A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. IEEE Trans. Cybern. **50**(2), 536–549 (2020). https://doi.org/10.1109/TCYB.2018.2869674
50. Wang, R., Dong, N.J., Gong, D.W., Zhou, Z.B., Cheng, S., Wu, G.H., Wang, L.: PCA-assisted reproduction for continuous multi-objective optimization with complicated Pareto optimal set. Swarm Evol. Comput. **60**, 100–795 (2021). https://doi.org/10.1016/j.swevo.2020.100795
51. Xu, Q., Zhang, C., Zhang, L.: A fast elitism Gaussian estimation of distribution algorithm and application for PID optimization. Sci. World J. **2014**(597278), 14 (2014). https://doi.org/10.1155/2014/597278
52. Zhang, Q., Zhou, A., Jin, Y.: RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. IEEE Trans. Evol. Comput. **12**(1), 41–63 (2008). https://doi.org/10.1109/TEVC.2007.894202
53. Zhao, H., Zhang, C.: An online-learning-based evolutionary many-objective algorithm. Inf. Sci. **509**, 1–21 (2020). https://doi.org/10.1016/j.ins.2019.08.069
54. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: 2006 IEEE International Conference on Evolutionary Computation, pp. 892–899 (2006). https://doi.org/10.1109/CEC.2006.1688406
55. Zhou, A., Zhang, Q., Jin, Y., Tsang, E., Okabe, T.: A model-based evolutionary algorithm for bi-objective optimization. In: 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2568–2575 (2005). https://doi.org/10.1109/CEC.2005.1555016
56. Zhou, Z., Wang, Z., Pang, T., Wei, J., Chen, Z.: A competition-cooperation evolutionary algorithm with bidirectional multi-population local search and local hypervolume-based strategy for multi-objective optimization. In: 2021 IEEE Congress on Evolutionary Computation (CEC), pp. 153–160 (2021). https://doi.org/10.1109/CEC45853.2021.9504689

Chapter 4
Learning to Understand the Problem Structure

This chapter focuses on an important aspect of *learning* the preference structure of the objectives, inherent in multi- and many-objective optimization problem formulations. This involves identifying the *non-essential* (*redundant*) objectives, and also determining the relative importance of the *essential* objectives. Such an approach to knowledge discovery is based on the following rationale. Modeling an optimization problem, analytically or through experiments, involves a lot of time and physical resources, possibly from multiple disciplines, in conjunction or isolation from each other. Often, it can be intriguing for analysts or decision makers (DMs) to know if the developed model represents the underlying problem in a minimal form or is marked by redundancy. Any redundancy among objectives, if revealed, could shed insightful light on the physics of the underlying problem, in addition to reducing its complexity and promising greater search efficiency for evolutionary multi- and many-objective optimization algorithms (EMâOAs). Furthermore, the revelation of the relative preferences among the essential objectives that are inherent in the problem models could also be significantly useful, as highlighted below.

In the context of many-objective problems, the challenges associated with the complete Pareto front (PF)-approximation by the EMâOAs are well known. Even where a complete PF-approximation is possible, selecting the best point of interest catering to many objectives is a non-trivial task. Taking into account these factors, the notion of guiding an EMâOA to only a few solutions preferred by the DMs seems quite appealing [12, 25, 38]. This requires the participation of DMs in one of the following forms. They may interact only at the beginning of an EMâOA run, providing preference information through—one or more reference point(s) [13, 40]; one or more reference directions [7]; and one or more light beam specifics [8], following which an EMâOA would target convergence of the population near the specific solutions on the PF. Alternatively, the DMs may be engaged multiple times along an EMâOA run [12] toward a more seamless optimization-cum-decision-making process. However, the generality and utility of this approach could be impaired, since the preferences of the DMs in the case of many-objective problems can *lack*

D. K. Saxena et al., *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization*, Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-2096-9_4

in—*objectivity* (a rational basis); *repeatability* (the same preferences, if the same solutions are repeated); *consistency* (broadly aligned preferences where sought progressively along an EMâOA run); and *coherence* (non-conflicting preferences amidst multiple DMs). This, in turn, could be attributed to *cognitive limitations* of human beings [27, 30, 39] in handling several factors at once (5 ± 2), and the fact that real-world decision-making: (i) is often based on *habitual domains* [43] of the DMs, consisting of personal experience, memory, thoughts, thinking paradigms, psychological states, and *perceived state of nature* rather than the true state of nature [18], and (ii) involves many people, often with conflicting objectives, and the whole process is redolent with feedback [1]. Such bottlenecks could potentially be resolved if the objectives' relative preferences inherent in the problem model were made available to the DMs. Ideally, regardless of the form and number of times the DMs may be engaged, their preferences should be consistent with those embedded in the problem model. If, in a practical situation, that is not the case, then at least the DMs must have a justification for the digression.

Approaches to identify the redundant objectives, referred to as *objective reduction* approaches were proposed as early as 2006. These approaches were based on preserving the *correlation structure* [11] and the *dominance structure* [2], of the non-dominated solution set obtained from an EMâOA. Both these approaches were further developed overtime and more comprehensive studies were available [3, 16, 33, 34]. Several other correlation-based approaches were developed parallelly [23] and subsequently [15, 17, 29, 41]. While most of these approaches are based on analyzing the set of non-dominated solutions obtained from an EMâOA run, some specialized EMâOAs for objective reduction have also been proposed [26, 37]. Considering the scope of this book, this chapter presents an ML-based framework [16, 34] that operates on the objective vectors of the non-dominated solutions obtained from an EMâOA; *learns* the preference structure of the objective functions by preserving the correlation structure of the solutions; and provides decision support in terms of:

1. Revelation of an *essential* objective set: an essential objective set ($\mathcal{F}_{\mathcal{S}}$) is defined as the smallest set of *conflicting* objectives that can generate the same *PF* as the original objective set ($\mathcal{F}_0$). Here, an essential objective (standalone) can be defined as one that conflicts with at least one other objective and is not correlated with any other objective. Notably: (i) if the cardinality of $\mathcal{F}_0$ is M, then the cardinality of $\mathcal{F}_{\mathcal{S}}$ shall be less than or equal to M, say $= m \leq M$, and (ii) revelation of $\mathcal{F}_{\mathcal{S}}$, implicitly reveals the *redundant* objective set, say $\mathcal{F}_{\mathcal{R}} = \mathcal{F}_0 \setminus \mathcal{F}_{\mathcal{S}}$, each member of which is correlated with at least one other objective.
2. Preference ranking of all the objective functions: the preference ranking of the objectives is based on their preference weights, namely w_is, such that $w_i \geq 0$ and $\sum_{i=1}^{M} w_i = 1$. In this case, w_i represents the normalized error (variance lost) that would be incurred if f_i were to be eliminated.
3. δ-MOSS (δ-minimum objective subset) analysis: here, for an allowable degree of error δ ($0 \leq \delta \leq 1$) specified by the DM, the task is to determine the δ-minimum

objective subset,[1] namely $\mathcal{F}_{\{\delta\}}$—the smallest subset of objectives that ensures that the error associated with omission of the remaining objectives does not exceed δ.

4. k-EMOSS (minimum objective subset of size k with minimum error) analysis: here, for an allowable set size k specified by the DM, the task is to determine the k-minimum set, namely $\mathcal{F}_{\{k\}}$—the subset of k objectives which ensures that the error associated with omission of the remaining objectives is the minimum, compared to any other possible combination of k objectives.

4.1 Foundation: Machine Learning Based Identification of Preference Structure of Objectives

Machine learning (ML)-based *dimensionality* reduction arises from the premise that the *intrinsic structure* of a *garbled* high-dimensional dataset can be revealed by transforming it so that the effect of *noise* and *redundancy* (dependencies) is minimized.

Principal Component Analysis (PCA) [36] achieves this goal by projecting the given data X so that the correlation structure is preserved most faithfully, in that PCA projects X on the eigenvectors of the correlation matrix of X. Notably, PCA is based on removing the *second order dependencies* in the given data. Hence, it is prone to errors, if applied to datasets with multi-modal Gaussian or non-Gaussian distributions [36], as in Fig. 4.1a. Several nonlinear dimensionality reduction methods exist, such as kernel PCA [35] and Graph-based methods [32], for removing the *higher order dependencies*. The former nonlinearly transforms the data by using a standard kernel function and then applies PCA in the transformed/kernel space. However, its success depends on the *a priori* chosen kernel. The latter eliminates this limitation by deriving *data-dependent* kernels. Maximum Variance Unfolding (MVU) [42] is a graph-based method that computes the low-dimensional representation by explicitly attempting to *unfold* the high-dimensional data manifold, as in Fig. 4.1. The unfolding is achieved by maximizing the Euclidean distances between the data points while locally preserving the distances and angles between *nearby* points. Mathematically, this can be posed as a semidefinite programming problem (SDP) [42], the output of which is the kernel matrix (say, K) representing the kernel space to which PCA can be applied.

The suitability of such ML approaches for objective reduction and revelation of the objectives' preference structure has been illustrated through an example, namely DTLZ5$(3, 5)$. It is an instance of the DTLZ5(I, M) problem [34], where M denotes the original number of objectives, and I denotes the *dimensionality*[2] of the PF.

[1] For a given $0 \leq \delta \leq 0$, there may be multiple subsets of objectives which ensure that the error associated with the the omission of the remaining objectives does not exceed δ. Each such subset is referred to as a δ-minimal objective subset. However, the δ-minimal objective subset having the smallest size is referred to as the δ-minimum objective subset.

[2] Here, dimensionality refers to the number of objectives that are essential to characterize the complete PF.

Hence, DTLZ5(3, 5) is a five-objective problem, with a three-dimensional PF, in that f_1–f_2–f_3 are perfectly correlated and an essential objective set is given by $\mathcal{F}_\mathcal{S} = \{f_3, f_4, f_5\}$. These characteristics are evident in Fig. 4.2a, for the solutions sampled on the true PF ($\mathcal{N}_{\mathcal{NP}}$). Therefore, an approach such as PCA, applied to $\mathcal{N}_{\mathcal{NP}}$, is expected to reveal the correlation between f_1–f_2–f_3, and the conflict between f_3–f_4–f_5. However, the challenge for the efficacy of PCA is bigger in realistic scenarios where an EMâOA may not offer an exact PF-approximation. For example, the non-dominated solutions obtained by a single run of NSGA-II [10] on DTLZ5(3, 5), denoted as $\mathcal{N}_{\mathcal{NS}}$ in Fig. 4.2b, reveal that NSGA-II has: (i) failed to converge to the true PF, since the maximum values for each objective are much larger than those possible on the true PF (Fig. 4.2a), and (ii) failed to capture the correlation structure of the true PF, since f_1–f_2–f_3 erroneously report partial conflict. More insights can be gained from Fig. 4.2c, in which $\mathcal{N}_{\mathcal{NS}}$ is projected in the $\mathcal{F}_\mathcal{S}$ subspace. In this *essential* subspace, solution A seems to dominate solution B. However, B qualifies as a member of $\mathcal{N}_{\mathcal{NS}}$, because with respect to the original set of objectives, it is better than A in one of the redundant objectives, namely f_1 or f_2. This illustrates that the presence of redundant objectives hinders the search efficiency of NSGA-II, given which the dominance relations characterizing $\mathcal{N}_{\mathcal{NS}}$ do not conform with those of $\mathcal{N}_{\mathcal{NP}}$. Theoretically, if NSGA-II were allowed to run for infinitely many generations, spurious solutions such as B could *die out*, and $\mathcal{N}_{\mathcal{NS}}$ may conform with $\mathcal{N}_{\mathcal{NP}}$, however, this is not possible when NSGA-II is practically run for a finite number of generations. In view of this, the objective reduction problem can be viewed as a machine learning problem. In that problem:

- The *intrinsic structure* of PF refers to its intrinsic dimensionality (m) and the essential components ($\mathcal{F}_\mathcal{S}$).
- The *garbled* high-dimensional data refers to the non-dominated solutions obtained from an EMâOA, which typically provide a poor PF-approximation and, therefore, are characterized by dominance relations that may be different from those of the true PF. In that situation, objectives that are perfectly correlated on PF may show a partial conflict in the EMâOA solutions obtained.

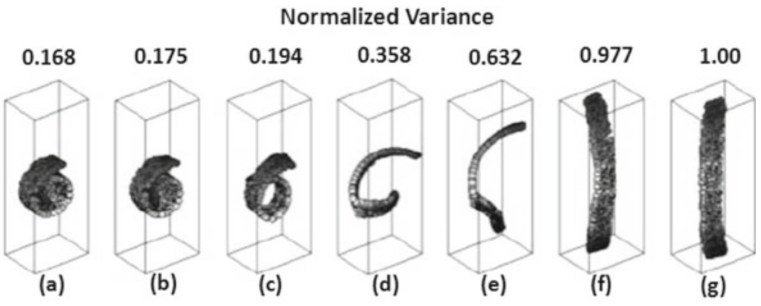
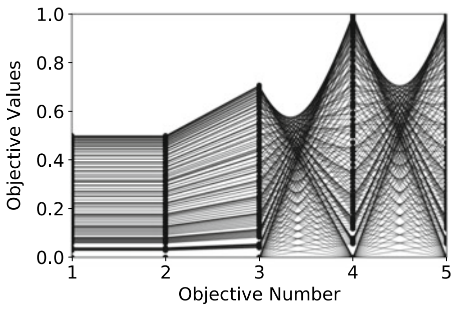
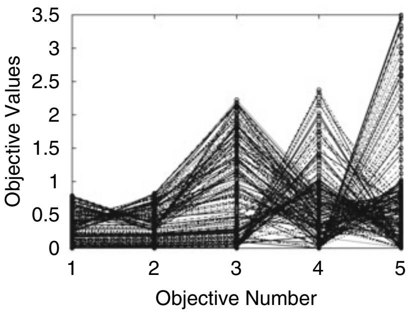


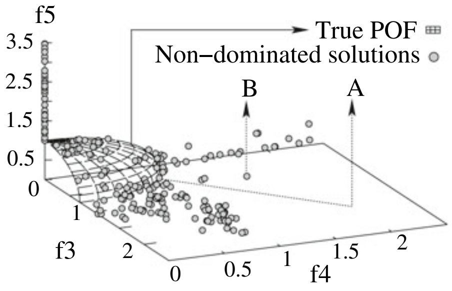
Fig. 4.1 Maximum variance unfolding (taken from [34])



(a) $\mathcal{N}_{\mathcal{NP}}$: Parallel coordinate plot



(b) $\mathcal{N}_{\mathcal{NS}}$: Parallel coordinate plot



(c) $\mathcal{N}_{\mathcal{NS}}$ versus true PF

Fig. 4.2 DTLZ5(3, 5): $\mathcal{N}_{\mathcal{NP}}$ denotes the solutions directly sampled on the true PF, while $\mathcal{N}_{\mathcal{NS}}$ denotes the PF-approximation obtained by NSGA-II (run for 2000 generations, with a population size of 200) (taken from [34])

- *Unnoised signal* refers to exact optimal solutions, that is, non-dominated solutions which are characterized by the same dominance relations as those which define the true PF (the fraction of $\mathcal{N}_{\mathcal{NS}}$ lying on the true PF, Fig. 4.2c).
- *Noised signal* refers to non-optimal solutions, characterized by dominance relations that are different from those that define the true PF (the fraction of $\mathcal{N}_{\mathcal{NS}}$ departing from the true PF, Fig. 4.2c).
- *Noise* refers to the difference in the characteristics of the *unnoised* and *noised* signal, for example, the difference in the dimensionality of *unnoised* signal (m) and that of the *noised* signal (upto M).
- *Redundancy* refers to the presence of objectives that are not conflicting (or correlated) with some other objectives and may contribute to *garbled data*.

Following the modeling of objective reduction as a machine learning problem (as above), the procedure for the determination of an essential objective set is as follows. Given am M-objective problem, the task of identifying the smallest set of m ($m \leq M$) conflicting objectives which preserves the correlation structure of the given non-dominated solution set, is achieved by eliminating objectives that are non-conflicting along the significant eigenvectors of the correlation matrix. Once an essential objective set is identified, any further elimination of an objective will distort the correlation structure, and the corresponding error is assessed by the objective's contribution to the significant eigenvectors. Here, two broad scenarios are possible with regard to the fidelity of the correlation structure of the available non-dominated solution set, in that it *may* or *may not conform* with that of the true PF. If it does, then the essential objective set determined as above would naturally represent the essential objective set for the underlying problem (PF). If it does not, then the available non-dominated dataset would need to be treated as *noisy*, necessitating *denoising* (discussed later), before the inferences drawn for the available non-dominated solution set can be extended for the underlying problem (PF). Considering that for an unknown problem, the conformance between the correlation structure of the true PF and that of the non-dominated solutions offered by an EMâOA cannot be verified, it becomes important that any approach to learning the objectives' preference structure includes some denoising mechanism(s).

4.2 Learning Preference Structure of Objectives

This section presents an ML-based framework to extract the objectives' preference structure, and provide decision support in terms of revelation of the essential objective set; preference ranking of all objectives; δ-MOSS; and k-EMOSS analysis. As depicted in Fig. 4.3, this framework is iterative in nature. In each iteration, it allows the use of linear and nonlinear objective reduction algorithms, namely, L-PCA and

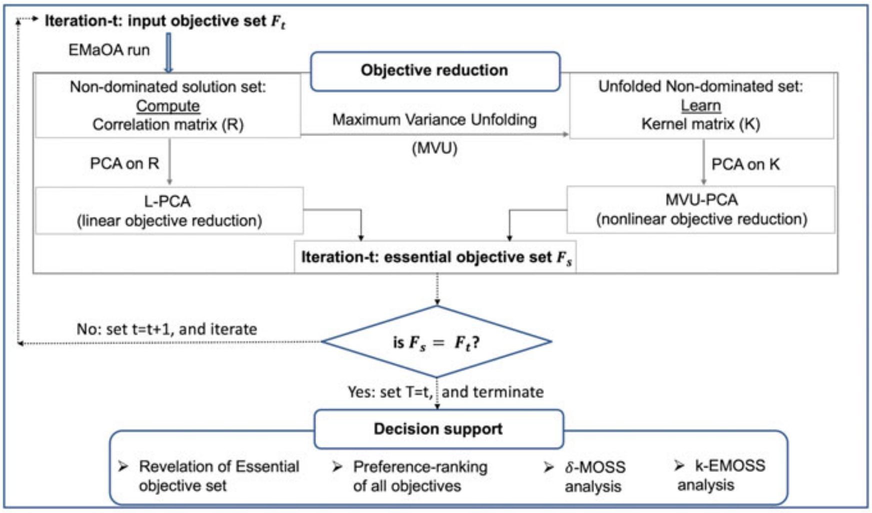

Iteration-t: input objective set F_t
EMaOA run
Objective reduction
Non-dominated solution set: Compute Correlation matrix (R)
Maximum Variance Unfolding (MVU)
Unfolded Non-dominated set: Learn Kernel matrix (K)
PCA on R
PCA on K
L-PCA (linear objective reduction)
MVU-PCA (nonlinear objective reduction)
Iteration-t: essential objective set F_s
No: set t=t+1, and iterate
is $F_s = F_t$?
Yes: set T=t, and terminate
Decision support
Revelation of Essential objective set
Preference-ranking of all objectives
δ-MOSS analysis
k-EMOSS analysis

Fig. 4.3 A schematic for the ML-based framework to extract the preference structure of objectives (taken from [16])

Framework 4.1: An ML-based framework to extract the preference structure of objectives.

```
Input: $t$; a non-dominated solution set obtained from an EMâOA corresponding to $\mathcal{F}_t = \{f_1, f_2, \ldots, f_{M_t}\}$; and an empty error list $\mathcal{L}$ (sized $M$)
1  begin
2      Obtain a set of non-dominated solutions by running an EMâOA corresponding to $\mathcal{F}_t$, for $N_g$ generations with a population size of $N$.
3      For L-PCA, compute $R$ (Eq. 4.1), and its eigenvalues and eigenvectors. For NL-MVU-PCA,, compute both $R$ and $K$ (Eq. 4.2), and the eigenvalues and eigenvectors for $K$.
4      Perform the Eigenvalue Analysis to identify the set of potentially conflicting objectives $\mathcal{F}_e \subseteq \mathcal{F}_t$ (Sect. 4.2.3).
5      Perform the Reduced Correlation Matrix Analysis to identify the identically correlated subsets in $\mathcal{F}_e$ (Sect. 4.2.4). If there is no such subset, $\mathcal{F}_s = \mathcal{F}_e$.
6      Apply the selection scheme to identify the most significant objective in each correlated subset, to arrive at $\mathcal{F}_s$, such that $\mathcal{F}_s \subseteq \mathcal{F}_e \subseteq \mathcal{F}_t$ (Sect. 4.2.5).
7      if $\mathcal{F}_s \neq \mathcal{F}_t$ then
8          For each redundant objective ($f_i \in \mathcal{F}_r$), compute $\mathcal{E}_{t,i}$ (Eq. 4.7)
9          For each essential objective ($f_j \in \mathcal{F}_s$), compute $\mathcal{E}_{t,j}$ (Eq. 4.7)
10         For all the redundant objectives combined, compute $\mathcal{E}_t^r$ (Eq. 4.8)
11         For all the essential objectives combined, compute $\mathcal{E}_t^s$ (Eq. 4.9)
12         For each redundant objective, compute $\eta_{t,i}^r$ (Eq. 4.10)
13         For each redundant objective, update $\mathcal{L}$ by storing $\eta_{t,i}^r$ at the location-$i$
14         Set $t = t + 1$, $\mathcal{F}_t = \mathcal{F}_s$, and go to Step 2
15     if $\mathcal{F}_s = \mathcal{F}_t$ then
16         Set $T = t$
17         For each essential objective ($f_j \in \mathcal{F}_s$), compute $\mathcal{E}_{t,j}$ (Eq. 4.7)
18         For all the essential objectives combined, compute $\mathcal{E}_t^s$ (Eq. 4.9)
19         For each essential objective, compute $\eta_{t,j}^s$ (Eq. 4.10)
20         For each essential objective, update $\mathcal{L}$ by storing $\eta_{t,j}^s$ at the location-$j$
21         Generate $\mathcal{SL}$, by sorting $\mathcal{L}$ in ascending order of the normalized error values therein, along with the corresponding objective indices
22         Stop and offer the decision support
23             Declare the final essential objective set $\mathcal{F}_{\mathcal{S}} = \mathcal{F}_s$
24             Declare the final redundant objective set $\mathcal{F}_{\mathcal{R}} = \mathcal{F}_0 \setminus \mathcal{F}_{\mathcal{S}}$
25             Report the total error $\mathcal{E}^R$ (Eq. 4.8) incurred for reduction of $\mathcal{F}_0$ to $\mathcal{F}_{\mathcal{S}}$
26             Perform the $\delta$-MOSS analysis using $\mathcal{SL}$ (Sect. 4.2.7.2)
27             Perform the $k$-EMOSS analysis using $\mathcal{SL}$ (Sect. 4.2.7.3)
```

NL-MVU-PCA, respectively [34]. While L-PCA is based on the decomposition of the eigenvalue of the correlation matrix (R), NL-MVU-PCA is based on the decomposition of the eigenvalue of the kernel matrix (K) that is learned using the MVU principle. Such iterations are allowed as long as their initial objective set $\mathcal{F}_t$ and the final objective set $\mathcal{F}_s$ are different; otherwise they are terminated and the decision support is offered.

The Framework 4.1 formalizes the constitutive steps for an iteration t with $M_t \leq M$ objectives, which are discussed below. However, before delving into the details, it must be realized that an *essential* objective is one that conflicts with at least one other objective and is not correlated with any other objective. Given this, the governing principle for identifying the essential objectives is as follows:

- First, the set of potentially conflicting objectives is identified, say, $\mathcal{F}_e \subseteq \mathcal{F}_t$. For any objective to qualify for $\mathcal{F}_e$, it must exhibit conflict with at least one other objective, along a significant direction of principal variance in the given set of solutions (this involves eigenvalue and eigenvector analysis, discussed later).
- Then the set of conflicting objectives, say $\mathcal{F}_s \subseteq \mathcal{F}_e$ is determined by: (i) identifying subsets of correlated objectives in $\mathcal{F}_e$ (if any) and (ii) retaining only one representative objective per such subset, in addition to including those objectives which did not belong to any correlated subset. To appreciate the above criteria, consider $\mathcal{F}_e = \{f_i, f_j, f_p, f_q\}$. Say that f_i and f_j qualified for $\mathcal{F}_e$ due to their conflict along the most significant direction of principal variance, while f_p and f_q qualified due to their conflict along the second most significant direction of principal variance. Furthermore, consider that f_i and f_p are correlated (this is possible since they correspond to different directions of principal variance). Therefore, if only one of them is retained, say f_i, it can explain the conflict with both f_j and f_q. In such a case, the set of conflicting objectives can be given by $\mathcal{F}_s = \{f_i, f_j, f_q\}$. Notably, f_j and f_q qualify as members of $\mathcal{F}_s$, since they were members of $\mathcal{F}_e$ and were not part of any correlated subset. It is critical to note that correlation is a set-based property, therefore, for f_i and f_p to be considered as correlated, the following criteria must be met: (i) the nature of the relationship of f_i and f_p with all other objectives must be similar, and (ii) the strength of their pairwise correlation must be sufficiently strong.

Given the above overarching principle, the details of the constitutive steps in the Framework 4.1 can be found below.

4.2.1 Construction of a Positive Semi-definite Matrix

As a first step, the correlation matrix (R) and the kernel matrix (K), which will be used for linear and nonlinear objective reduction, respectively, are constructed. Toward that end, the non-dominated solutions are obtained by running an EMâOA with the objective set $\mathcal{F}_t = \{f_1, \ldots, f_{M_t}\}$, corresponding to a population size of N. Let the objective vector in the non-dominated set, corresponding to the ith objective (f_i), be denoted by $\bar{f_i} \in \mathbb{R}^N$ and its mean and standard deviation by $\mu_{\bar{f_i}}$ and $\sigma_{\bar{f_i}}$, respectively. Furthermore, let $\hat{f_i} = (\bar{f_i} - \mu_{\bar{f_i}})/\sigma_{\bar{f_i}}$. Then, the input data X, an $M_t \times N$ matrix, can be composed as $X = [\hat{f_1}\ \hat{f_2} \ldots \hat{f}_{M_t}]^T$.

Given an X, the correlation matrix R is defined by Eq. 4.1. Notably, L-PCA is based on the decorrelation of R, that is, the removal of second-order dependencies in X. Therefore, the scope of L-PCA is largely limited to *linear* objective reduction.

$$R = \tfrac{1}{M_t} X X^T. \tag{4.1}$$

For the removal of higher order dependencies in X, NL-MVU-PCA relies on the decorrelation of the kernel matrix K. The determination of K involves solving the SDP formulation given by Eq. 4.2. Notably, it requires the correlation matrix R as an input. This explains why even in the case of NL-MVU-PCA, the Framework 4.1 calls for computation of both R and K.

$$\begin{aligned}
\text{Maximize } & trace(K) = \textstyle\sum_{ij} \tfrac{(K_{ii} - 2K_{ij} + K_{jj})}{2M_t}, \\
\text{Subject to: } & (a) K_{ii} - 2K_{ij} + K_{jj} = R_{ii} - 2R_{ij} + R_{jj}, \forall\, \eta_{ij} = 1, \\
& (b) \textstyle\sum_{ij} K_{ij} = 0, \\
& (c) K \text{ is positive-semidefinite}, \\
\text{where : } & R_{ij} \text{ is the } (i, j)\text{th element of the correlation matrix } R, \\
& \eta_{ij} = \begin{cases} 1, \text{ if } \dot{f}_j \text{ is among the } q - \text{nearest neighbor of } \dot{f}_i, \\ 0, \text{ otherwise.} \end{cases}
\end{aligned} \tag{4.2}$$

The neighborhood relation η_{ij} in Eq. 4.2 is governed by the parameter q. For each input feature ($\bar{f}_i \in \mathbb{R}^N$), q represents the number of neighbors with respect to which the local *isometry* is to be retained during unfolding. In other words, $\eta_{ij} \in \{0, 1\}$ indicates whether there is an edge between $\bar{f}_i$ and $\bar{f}_j$ in the graph formed by pairwise connecting all q-nearest neighbors. While a high value of q ensures better retention of isometry, it delays the unfolding process. On the contrary, a low value of q offers fast unfolding, but at the risk of distorting the isometry. Given this trade-off, the proper selection of q is crucial. From the perspective of accuracy (preservation of local isometry), higher values of q should be preferred, as long as the matrix K is not completely defined by the constraints, and the objective of the SDP formulation also has a role to play. It was established in [34] that even the most constrained case of $q = M_t - 1$, fulfills the above criterion. Therefore, $q = M_t - 1$ was recommended and used. It may also be noted that the SDP problem is convex, for which polynomial time off-the-shelf solvers, such as the SeDuMi [24] toolbox in MATLAB are available. In [34], SeDuMi had been used.

4.2.2 Computation of Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors of R and K (each, an $M_t \times M_t$ matrix) are determined, in the case of L-PCA and NL-MVU-PCA, respectively. Let these be given by: $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_{M_t}$ and $V_1, V_2, \ldots V_{M_t}$, respectively. As a foundation for subsequent processing, the following may be noted.

(a) If $e_j = \lambda_j / \sum_{j=1}^{M_t} \lambda_i$, then $\sum_{j=1}^{M_t} e_j = 1$.
(b) The ith component of jth principal component, say f_{ij}, reflects on the contribution of f_i toward $V_j \in \mathbb{R}^{M_t}$.
(c) From the orthonormality of the eigenvectors, it follows that $|V_j| = \sum_{i=1}^{M_t} f_{ij}^2 = 1 \; \forall \; j = 1, \ldots, M_t$.
(d) The contribution of f_i across all V_j's can be given by $c_i = \sum_{j=1}^{M_t} e_j f_{ij}^2$, such that $\sum_{i=1}^{M_t} c_i = 1$.

4.2.3 Eigenvalue Analysis

This step aims to identify the conflicting objectives (in $\mathcal{F}_0$) along the *significant* principal components (V_js), as follows. First, the count of *significant* V_js is determined. These correspond to the smallest number (N_v) such that $\sum_{j=1}^{N_v} e_j \geq \theta$, where the variance threshold $\theta \in [0, 1]$ is an algorithm parameter. $\theta = 0.997$ had been used in [34]. Subsequently, each *significant* V_j is interpreted as follows:

- An objective f_i with the highest contribution to V_j by magnitude is chosen, along with all other objectives with contributions of the opposite sign to V_j.
- If all the objectives have the same-sign contributions (all positive/negative), then the objectives with the top two contributions by magnitude are picked.

Let the collection of all the above-picked objectives constitute a set $\mathcal{F}_e \subseteq \mathcal{F}_t$.

4.2.4 Reduced Correlation Matrix (RCM) Analysis

This step aims to identify the subsets of *identically* correlated objectives within the set $\mathcal{F}_e$ determined by eigenvalue analysis. Such subsets may exist because the objectives chosen with reference to different (significant) V_js could well be correlated. If such subsets exist, then the most significant objective in each subset can be retained while the rest could be discarded, allowing for further reduction of $\mathcal{F}_e$. The above can be achieved through the following steps:

1. Construction of a Reduced Correlation Matrix (RCM): RCM is composed of the columns of R that correspond to the objectives in $\mathcal{F}_e$. Equivalently, the RCM is the same as R, except that the columns corresponding to the objectives in $\mathcal{F}_t \setminus \mathcal{F}_e$ are not included.
2. Identification of *potentially correlated subset* for each objective in $\mathcal{F}_e$: toward that goal, for each $f_i \in \mathcal{F}_e$, all $f_j \in \mathcal{F}_e$ which satisfy Eq. 4.3 constitute a *potentially* correlated subset $\hat{\mathcal{S}}_i$. Notably, Eq. 4.3 manifests the fact that the correlation between the objectives is a set-based property. Hence, unless the nature of the relationship of each of f_i and f_j with *all* other objectives is not identical, they cannot be considered as potentially correlated.

$$\text{sign}(R_{ik}) = \text{sign}(R_{jk}),\ k = 1, 2, \ldots, M_t. \tag{4.3}$$

3. Identification of *correlated subset* for each objective in $\mathcal{F}_e$: those $f_j \in \mathcal{F}_e$ which, in addition to Eq. 4.3, also satisfy Eq. 4.4, constitute an *identically* correlated subset $\mathcal{S}_i$. Notably, Eq. 4.4 accounts for the fact that the non-dominated set based on which R is computed may be *garbled*, and hence the strengths of correlation evident from R may not conform with the true correlations on the PF. In a sense, Eq. 4.4 serves to have a *denoising* effect, while interpreting a matrix based on *garbled* data, to minimize the chances of inaccurate identification of $\mathcal{S}$.

$$R_{ij} \geq \text{the correlation threshold } (T_{cor}). \tag{4.4}$$

The following may be observed with respect to the determination of T_{cor}. There are several guidelines for interpreting the strength of correlation, one of which is the Cohen scale [4]. It interprets a correlation strength of 0.1 to 0.3 as *weak*, 0.3 to 0.5 as *moderate* and 0.5 to 1.0 as *strong*. However, in the current context, a rigid interpretation of the correlation strength will not be helpful because T_{cor} needs to adapt to conflicting goals depending on the problem being solved. In view of this, low and high values T_{cor} are desired in solving problems with high and negligible redundancy, respectively. Toward this end, a dynamic calculation of T_{cor} is recommended based on the spectrum of the eigenvalues. This is based on the understanding that with an increase in the degree of redundancy in a problem: (i) the first eigenvalue e_1 becomes predominantly higher, and (ii) fewer principal components are required to account for a certain variance threshold. For a given M_t-objective problem, let $M_t^{2\sigma}$ denote the number of principal components required to account for 95.4% of the variance (analogical to Gaussian distributions, without restricting the scope to these), then T_{cor} can be computed as per Eq. 4.5.

$$T_{cor} = 1.0 - e_1(1.0 - M_t^{2\sigma}/M_t). \tag{4.5}$$

Notably, for a highly redundant problem where e_1 will be very high and $M_t^{2\sigma}$ will be small compared to M_t, T_{cor} will have a small value. Therefore, when the eigenvalue spectrum indicates higher redundancy, a lower T_{cor} improves the chances that potentially correlated objectives pass the correlation strength test (Eq. 4.4). In contrast, for a problem with low redundancy, e_1 will be low and $M_t^{2\sigma}$ will be comparable to M_t, T_{cor} will have a very high value. Hence, when the eigenvalue spectrum indicates a lower redundancy, a higher T_{cor} reduces the chances that potentially correlated objectives pass the correlation strength test (Eq. 4.4).

4.2.5 Selection Scheme for Final Reduction

Once the subsets of identically correlated objectives in $\mathcal{F}_e$ are identified by the RCM analysis, the goal is to identify and retain the most significant objective in each subset ($\mathcal{S}$) and eliminate the remaining ones. This is achieved by: (i) attributing a selection score for each objective, based on its contribution to the *significant* principal components,[3] as given by Eq. 4.6, and (ii) retaining the objective with highest sc_i in each $\mathcal{S}$ and eliminating the rest.

$$sc_i = \textstyle\sum_{j=1}^{N_v} e_j |f_{ij}|. \tag{4.6}$$

Doing so helps reduce $\mathcal{F}_e$ to $\mathcal{F}_s$, which becomes an *essential* objective set for the current iteration of the framework. Given this, $\mathcal{F}_r = \mathcal{F}_t \setminus \mathcal{F}_s$ becomes the redundant objective set for the current iteration of the framework.

4.2.6 Computation of Error

In Sect. 4.2.2, it was highlighted that the contribution of an objective f_i across all V_j's can be given by $c_i = \sum_{j=1}^{M_t} e_j f_{ij}^2$, such that $\sum_{i=1}^{M_t} c_i = 1$. Hence, if f_i were to be eliminated, then the variance left unaccounted for would be given by c_i, and would account for the error corresponding to the omission of f_i.

Given the above, the error in iteration t, corresponding to an essential objective $f_j \in \mathcal{F}_s$, denoted by $\mathcal{E}_{t,j}$, is defined as nothing but the corresponding c_j (Eq. 4.7). However, the error in iteration t, corresponding to a redundant objective $f_i \in \mathcal{F}_r$, denoted by $\mathcal{E}_{t,i}$ is defined as the corresponding c_i scaled down by a factor, since part of the variance c_i is already accounted for, by virtue of f_i's correlation with some essential objective $f_j \in \mathcal{F}_s$.

$$\left.\begin{aligned}
\mathcal{E}_{t,i} = {} & c_i(1.0 - \max\{\delta_{ij}.R_{ij}\}) \text{ for } f_j \in \mathcal{F}_s,\\
\text{where: } & c_i = \textstyle\sum_{k=1}^{M_t} e_k f_{ik}^2,\\
& \delta_{ij} = \begin{cases} 1, & \text{if } f_i \text{ and } f_j \text{ are identically correlated,}\\ 0, & \text{otherwise,}\end{cases}\\
& R_{ij} = \text{Correlation between } f_i \text{ and } f_j,
\end{aligned}\right\} \text{for } f_i \in \mathcal{F}_r,$$
$$\left.\mathcal{E}_{t,j} = c_j, \text{ where: } c_j = \textstyle\sum_{k=1}^{M_t} e_k f_{jk}^2\right\} \text{for } f_j \in \mathcal{F}_s. \tag{4.7}$$

Notably, defining $\mathcal{E}_{t,i}$ for $f_i \in \mathcal{F}_r$, and $\mathcal{E}_{t,j}$ for $f_j \in \mathcal{F}_s$, iteration-wise, enables computation of all other measures necessary for the decision support. For example,

[3] This contribution can be given by $sc_i = \sum_{j=1}^{N_v} e_j f_{ij}^2$ (Sect. 4.2.2). However, e_j and f_{ij} being less than one, may lead to indiscriminately low values for sc_i, hence, adapted in Eq. 4.6.

1. Once the framework terminates, the user/DM may be interested in knowing the total error incurred owing to the omission of the redundant objectives ($\mathcal{F}_r$) across all the iterations combined, implying the error corresponding to the final redundant objective set given by $\mathcal{F}_{\mathcal{R}} = \mathcal{F}_0 \setminus \mathcal{F}_{\mathcal{S}}$. Toward that end: (i) first, the total error for all redundant objectives in iteration t, denoted by $\mathcal{E}_t^r$ can be calculated (Eq. 4.8), and (ii) then the total error for all redundant objectives starting from the initial iteration until the penultimate iteration can be calculated (Eq. 4.8). The final iteration T is excluded from $\mathcal{E}^R$, since, by definition, it does not report any redundant objectives. Furthermore, it may be noted that the error $\mathcal{E}_0^r$ for the first iteration is the variance that will not be taken into account if the objectives in the corresponding $\mathcal{F}_r$ were to be eliminated. Hence, the error $\mathcal{E}_1^r$ in the second iteration will only be with respect to $1 - \mathcal{E}_0^r$ or the variance of the original problem. This argument is generalized in Eq. 4.8.

$$\mathcal{E}_t^r = \sum_{i:f_i \in \mathcal{F}_{\mathcal{R}}} \mathcal{E}_{t,i}, \quad \mathcal{E}^R = \mathcal{E}_0^r + \sum_{t=1}^{T-1} \mathcal{E}_t^r (1 - \mathcal{E}_{t-1}^r). \tag{4.8}$$

2. For the δ-MOSS and k-EMOSS analysis, it becomes necessary to generate an M-dimensional list, say $\mathcal{SL}$, consisting of *normalized errors in ascending order* along with the corresponding objective indices. This pre-necessitates that:

   - The *normalized* error is available for each objective in each iteration t. Toward it: (a) the errors for the essential objectives in any iteration t need to be computed, which Eq. 4.9 provides, and (b) the normalized errors in iteration can be computed as per Eq. 4.10. To this end: (a) for $t < T$, only the normalized errors for redundant objectives are provided for, since the essential objectives identified in the intermediate iterations are not final and subject to reduction in subsequent iterations; and (b) for $t = T$, only the normalized error for essential objectives is provided for, since the final iteration by its very definition involves only the essential objectives.

$$\mathcal{E}_t^s = \sum_{j:f_j \in \mathcal{F}_s} \mathcal{E}_{t,j}. \tag{4.9}$$

$$\text{If } t < T : \ \eta_{t,i}^r = \frac{\mathcal{E}_{t,i}}{\mathcal{E}_t^r + \mathcal{E}_t^s}, \ \text{else if } t = T : \ \eta_{t,j}^s = \frac{\mathcal{E}_{t,j}}{\mathcal{E}_t^s}. \tag{4.10}$$

   - An M-dimensional list, $\mathcal{L}$, is generated over all iterations t ($0 \le t \le T$), so that $\mathcal{L}_k$ stores the normalized error for f_k. Toward that goal, an empty M-dimensional error list, $\mathcal{L}$, is initialized, and then: (a) in each intermittent iteration t ($0 \le t < T$), $\mathcal{L}$ is updated by normalized errors for the redundant objectives, in that $\eta_{t,i}^r$ corresponding to $f_i \in \mathcal{F}_{\mathcal{R}}$ is assigned to $\mathcal{L}_i$; and (b) in the final iteration $t = T$, $\mathcal{L}$ is updated by the normalized errors for the essential objectives, in that $\eta_{t,j}^r$ corresponding to $f_j \in \mathcal{F}_{\mathcal{S}}$ is assigned to $\mathcal{L}_j$. Hence, upon termination, a fully populated, M-dimensional, normalized error list $\mathcal{L}$ will be available, which could then be sorted in ascending order of error values along with the corresponding objective indices ($\mathcal{SL}$).

For ease of reference and processing of $\mathcal{SL}$, let its composition (independent of reference to $\eta^r_{t,i}$ or $\eta^s_{t,j}$) be in *ascending order* of the values of η, and let the set of objectives based on the corresponding indices be given by Eq. 4.11, where $\lhd$ denotes *less-important-than* relation.

$$\mathcal{SL} : \{\eta_{r_1} \leq \eta_{r_2} \leq \ldots \leq \eta_{r_M}\} \Rightarrow \{f_{r_1} \lhd f_{r_2} \lhd \ldots \lhd f_{r_M}\}. \tag{4.11}$$

4.2.7 Decision Support Based on the Learned Preference Structure

Termination of the Framework 4.1 facilitates decision support in terms of the following.

4.2.7.1 Revelation of an Essential or Redundant Objective Set

Given an M-objective problem, an essential objective set, $\mathcal{F}_\mathcal{S}$, may consist of $m \leq M$ objectives. By definition, the m objectives in $\mathcal{F}_\mathcal{S}$ should lead to the same PF as the M objectives in $\mathcal{F}_0$. Furthermore, the first $M - m$ locations in the normalized error list $\mathcal{SL}$ should ideally comprise of zeros, and the redundant objectives could be indiscriminately assigned to these locations. As per the Framework 4.1, consideration of $\mathcal{F}_\mathcal{S}$ is associated with an error $\mathcal{E}^R$ (Eq. 4.8). If $\mathcal{E}^R \neq 0$, then it may apparently contradict the definition of $\mathcal{F}_\mathcal{S}$, and raise doubts on the credibility of the objectives' preference ranking derived from $\mathcal{SL}$. Such apparent contradictions should be interpreted from two perspectives discussed as follows:

1. The quality of obtained PF-approximation: if an EMâOA run were to offer an ideal PF-approximation, then the error reported by the Framework 4.1 corresponding to each $f_i \in \mathcal{F}_\mathcal{R}$ would be zero, since there would exist some $f_j \in \mathcal{F}_s$ such that $R_{ij} = 1$ (Eq. 4.7). However, practically, in most cases, the obtained PF-approximation may be *noisy*, which implies that even if f_i and f_j are perfectly correlated on true PF, the obtained solutions may yield $R_{ij} < 1$. This explains why the Framework 4.1 may report $\mathcal{E}^R \neq 0$ corresponding to $\mathcal{F}_\mathcal{S}$, and also why different $f_i \in \mathcal{F}_\mathcal{R}$ may exhibit a preference ranking in SL.
2. The nature of conflict/correlation among the objectives: it is possible that some objectives in a given problem may have partial conflict/correlation among themselves. If the degree of correlation is stronger than the degree of conflict, then they may be inferred as correlated—subject to some error. This again explains the apparent contradictions with respect to non-zero $\mathcal{E}^R$, and the preference ranking of the objectives.

4.2.7.2 δ-MOSS Analysis

As introduced before, the δ-MOSS analysis seeks to find the smallest subset of objectives (δ-minimal objective subset or $\mathcal{F}_{\{\delta\}}$) which ensures that the error associated with omission of the remaining objectives does not exceed δ. Given the sorted error list $\mathcal{SL}$ (Eq. 4.11) and a user-defined δ, $\mathcal{F}_{\{\delta\}}$ can be determined by finding the lowest value of J, that satisfies $\sum_{j=1}^{J} \eta_{r_j} \geq \delta$. Subsequently, $\mathcal{F}_{\{\delta\}}$ can be given as $\{f_{r_J}, f_{r_{J+1}}, \ldots, f_{r_M}\}$, with size of $M - J + 1$.

4.2.7.3 k-EMOSS Analysis

As introduced before, the k-EMOSS analysis refers to finding the subset of k objectives (k-minimal objective subset or $\mathcal{F}_{\{k\}}$), such that the error associated with omission of the remaining objectives is the minimum, compared to that corresponding to any other possible combination of k objectives. Given the sorted error list $\mathcal{SL}$ (Eq. 4.11) and a user-defined k, $\mathcal{F}_{\{k\}}$ can be determined by selecting the last k objectives from $\mathcal{SL}$, given by $\{f_{r_{M-k+1}}, f_{r_{M-k+2}}, \ldots, f_{r_M}\}$. Hence, the omission of the remaining objectives (first $M - k$ objectives) would collectively result in the lowest possible error or the k-minimal error, given by $\eta_k = \sum_{j=1}^{M-k} \eta_{r_j}$.

4.3 Test Suite and Experimental Settings

The test suite used in this chapter, aligned with [34], consists of both redundant and non-redundant problems, described below.

Among the *redundant* problems, the DTLZ5(I, M) [11] and WFG3(M) [19] problems are used, whose significant features as described below:

1. DTLZ5(I, M): here, M denotes the number of objectives, while I denotes the dimension of PF, which is characterized by the parameter $g = 0$. Notably, the first $M - I + 1$ objectives are perfectly correlated, while the rest are in conflict with every other objective in the problem. Hence, an *essential* objective set is as given by Eq. 4.12.

$$\left.\begin{array}{l} \mathcal{F}_{\mathcal{S}} = \{f_k, f_{M-I+2}, \ldots, f_M\}, \text{ where} \\ k \in \{1, \ldots, M - I + 1\} : \text{in general, and} \\ k = M - I + 1 : \text{for L-PCA and NL-MVU-PCA in particular.} \end{array}\right\} \quad (4.12)$$

In the case of L-PCA and NL-MVU-PCA, $k = M - I + 1$ because among the possible indices for k, the variance of $f_{k=M-I+1}$ is the highest, and these algorithms are based on the premise that a greater variance corresponds to the signal, while a smaller variance corresponds to noise.

Table 4.1 Test problems: variable-settings and quality indicators (taken from [34])

Problem	Objectives	Variables		Quality indicators for PF	
		ρ	κ	g	$\mathcal{D}_{\mathcal{A}} = \mathcal{D}_{\mathcal{T}}$
DTLZ1(M)	$f_{1:M}$	$M - 1$	5	0	$\sqrt{0.25M}$
DTLZ2(M)	$f_{1:M}$	$M - 1$	10	0	$\sqrt{M}$
DTLZ3(M)	$f_{1:M}$	$M - 1$	10	0	$\sqrt{M}$
DTLZ4(M)	$f_{1:M}$	$M - 1$	10	0	$\sqrt{M}$
DTLZ5(I, M)	$f_{1:M}$	$M - 1$	10	0	$\sqrt{V}$*
WFG3(M)	$f_{1:M}$	$M - 1$	20	NA	NA

* V= $\left(\frac{1}{2}\right)^{M-I} + \sum_{i=2}^{M-I+1} \left(\frac{1}{2}\right)^{M-I+2-i} + I - 1$

2. WFG3(M): here, M denotes the number of objectives, and PF degenerates into a linear hyperplane such that $\Sigma_{m=1}^{M} f_m = 1$. Notably, the first $M - 1$ objectives are perfectly correlated, while the last objective is in conflict with every other objective in the problem. Hence, an *essential* objective set is as given by Eq. 4.13.

$$\left.\begin{array}{l} \mathcal{F}_{\mathcal{S}} = \{f_k, f_{M-I+2}, \ldots, f_M\}, \text{ where} \\ k \in \{1, \ldots, M - I + 1\} : \text{in general, and} \\ k = M - I + 1 : \text{for L-PCA and NL-MVU-PCA in particular.} \end{array}\right\} \quad (4.13)$$

Here, $k = M - 1$ for L-PCA and NL-MVU-PCA, as among all possible indices for k, the variance of $f_{k=M-I+1}$ is the highest.

For the *non-redundant* problems (the scalable DTLZ1 to DTLZ4), the degree of convergence can be quantified by a problem parameter g.[4] Notably, all the objectives in these problems are equally conflicting. Hence, the *essential* objective set is as given by Eq. 4.14.

$$\mathcal{F}_{\mathcal{S}} = \{f_1, \ldots, f_M\}. \quad (4.14)$$

These problems involve variables that have been categorized in [19] as the *distance* variables (say, κ, whose change results in comparable solutions) or the *position* variables (say, ρ, whose change results in incomparable solutions). The settings used for κ and ρ are reported in Table 4.1.

The performance of Framework 4.1 is evaluated based on experiments with:

- A solution set offering an exact PF-approximation (symbolizing only the *unnoised* signal): toward this end, the solutions are sampled on the true PF ($\mathcal{N}_{\mathcal{P}}$). Notably, the use of $\mathcal{N}_{\mathcal{P}}$ has been restricted here to a single problem, namely, DTLZ5(3, 5), to exemplify some of the salient features, including the interpretation of error as discussed in Sect. 4.2.7.1. To generate $\mathcal{N}_{\mathcal{P}}$, the sampling scheme in [44] was used. In that scheme, the number of solutions generated is given by $N = C_{m-1}^{H+m-1}$,

[4] This is also true for DTLZ7 but it is precluded here due to the inconsistency between its formulation and its PF, as presented in [14].

where m represents the dimension of PF and H is a user-defined parameter (whose recommended settings are $H = 8$ for $m \leq 10$ and $H = 4$ for $m > 10$).

- Solution sets offering inexact PF-approximation (symbolizing a combination of the *unnoised* and *noised* signal): toward this end, the non-dominated solution sets generated by two EMâOAs, namely ϵ-EMâOA [9] and NSGA-II [10] are considered. These sets of solutions, called $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$, respectively, correspond to a population size of 200 and runs of up to 2000 generations each. The recombination and mutation operators include SBX ($p_c = 0.9$ and $\eta_c = 5$) [5] and polynomial mutation ($p_m = 0.1$ and $\eta_m = 20$) [6] for an n variable problem. Furthermore, $\epsilon = 0.3$ is used for ϵ-EMâOA.

For each problem, the experiments with each of $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$, correspond to 20 different seeds. For all problems, except WFG problems, the quality of the PF-approximation offered by $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$ is assessed in terms of

- Convergence: by computing the value of the problem parameter g, and comparing it with $g = 0$ corresponding to the PF.
- diversity: by computing the value of the normalized maximum spread indicator, $I_s = \mathcal{D}_\mathcal{A}/\mathcal{D}_\mathcal{T}$ [31], where, $\mathcal{D}_\mathcal{A}$ represents the actual dispersal of solutions in $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$, and $\mathcal{D}_\mathcal{T}$ represents the dispersal of solutions on the true PF. Notably, the problem-wise derivations for $\mathcal{D}_\mathcal{T}$ can be found in the supplementary file to [34]. Table 4.1, reports only their final values for brevity. Naturally, if $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$ offered $I_s = 1$, it would mean that they have covered PF completely.

4.4 Implementation of Objective Reduction Framework on a Sample Problem

This section demonstrates the implementation of L-PCA and NL-MVU-PCA embedded in Framework 4.1, on a sample problem. Toward it, first the implementation based on $\mathcal{N}_\mathcal{P}$ (representing exact PF-approximation) is shown. Then, the implementation based on $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$ (representing inexact PF-approximation) is shown. For this purpose, the DTLZ5(3, 5) problem is used. It is a five-objective problem with a three-dimensional PF. This problem has been chosen due to its moderate degree of redundancy (40% objectives are redundant), with the aim of highlighting the absence of any *ad-hoc* fixes in L-PCA and NL-MVU-PCA, for problems with high or negligible degrees of redundancy.

4.4.1 Performance on a Solution Set with Exact PF-Approximation

The prerequisite correlation matrix R and the kernel matrix K based on $\mathcal{N}_\mathcal{P}$, and the corresponding eigenvalues and eigenvectors are presented in Table 4.2.

Table 4.2 DTLZ5$(3, 5)$: the eigenvalues and eigenvectors of matrices R and K, for $\mathcal{N}_{\mathcal{P}}$ (taken from [34])

(a) Correlation matrix (R)

	f_1	f_2	f_3	f_4	f_5
f_1	1.000	1.000	1.000	-0.474	-0.434
f_2	1.000	1.000	1.000	-0.474	-0.434
f_3	1.000	1.000	1.000	-0.474	-0.434
f_4	-0.474	-0.474	-0.474	1.000	-0.434
f_5	-0.434	-0.434	-0.434	-0.434	1.000

(b) Kernel matrix (K)

f_1	f_2	f_3	f_4	f_5
2.586	2.586	4.015	-4.723	-4.465
2.586	2.586	4.015	-4.723	-4.465
4.015	4.015	6.317	-7.401	-6.947
-4.723	-4.723	-7.401	23.061	-6.213
-4.465	-4.465	-6.947	-6.213	22.090

(c) Significant eigenvalues and eigenvectors of both R and K

R			K		
e_1 = 0.6867	e_2 = 0.2866	e_3 = 0.0266	e_1 = 0.5138	e_2 = 0.4855	e_3 = 0.0006
V_1	V_2	V_3	V_1	V_2	V_3
0.538	-0.009	0.209	0.134	0.273	-0.455
0.538	-0.009	0.209	0.134	0.273	-0.455
0.538	-0.009	0.209	0.211	0.426	0.757
-0.272	-0.691	0.668	-0.874	-0.173	0.077
-0.239	0.722	0.649	0.394	-0.799	0.075

4.4.1.1 L-PCA on $\mathcal{N}_{\mathcal{P}}$

Based on the eigenvalues and eigenvectors of the correlation matrix R, the L-PCA analysis is summarized in Table 4.3. In that:

- The eigenvalue analysis (Table 4.3a) shows that the top three principal components are needed to account for $\theta = 0.997$, leading to $\mathcal{F}_e = \{f_1, f_2, f_3, f_4, f_5\}$.
- The subsequent RCM analysis (Table 4.3b) suggests three *potentially correlated subsets*, namely, $\hat{\mathcal{S}}_1$, $\hat{\mathcal{S}}_2$, and $\hat{\mathcal{S}}_3$. Since $R_{12} = R_{13} = R_{23} = 1.0$ and $T_{cor} = 0.5879$, each objective pair in these subsets fulfills Eq. 4.4(ii), implying three *identically* correlated subsets, namely $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \{f_1, f_2, f_3\}$.
- As the selection scores of f_1, f_2 and f_3 happen to be equal (Table 4.3c), the analysis concludes with $\mathcal{F}_s = \{f_k, f_4, f_5\}$, where f_k could be f_1 or f_2 or f_3.
- Letting $\mathcal{F}_s = \{f_3, f_4, f_5\}$ implies $\mathcal{E}_0 = c_1^M(1.0 - R_{13}) + c_2^M(1.0 - R_{23}) = 0.0$.
- Notably, Iteration-2 of L-PCA fails to further reduce any objectives, and therefore is not shown here. In effect, $\mathcal{F}_{\mathcal{S}}$ is nothing but $\mathcal{F}_s$ in iteration-1.

Table 4.3 DTLZ5(3, 5): Iteration-1 of L-PCA with $\mathcal{N}_{\mathcal{P}}$ (taken from [34])

(a) Eigenvalue Analysis

PCA	Variance	Cumulative	Objectives Selected				
(N^v)	(%)	(%)	f_1	f_2	f_3	f_4	f_5
1	68.67	68.67			f_3	f_4	f_5
2	28.66	97.33	f_1	f_2	f_3	f_4	f_5
3	02.66	99.99				f_4	f_5

(b) RCM Analysis

1) Potentially correlated subsets	Eq 4.4(i)	$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_2 = \hat{\mathcal{S}}_3 = \{f_1, f_2, f_3\}$
2) Correlation threshold: T_{cor}	Eq 4.5	1.0-0.6867(1.0-2/5) = 0.5879
3) Correlated subsets	Eq 4.4(ii)	$\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \{f_1, f_2, f_3\}$

(c) Selection scheme

	e_1=0.6867	e_2=0.2866	e_3=0.0266	sc_i
	V_1	V_2	V_3	
f_1	0.538	-0.009	0.209	0.3778
f_2	0.538	-0.009	0.209	0.3778
f_3	0.538	-0.009	0.209	0.3778

4.4.1.2 NL-MVU-PCA on $\mathcal{N}_{\mathcal{P}}$

Based on the eigenvalues and eigenvectors of the kernel matrix K, this analysis is summarized in Table 4.4. In that:

- The eigenvalue analysis (Table 4.4a) leads to $\mathcal{F}_e = \{f_1, f_2, f_3, f_4, f_5\}$.
- Since $R_{12} = R_{13} = R_{23} = 1.0$ (Table 4.2a) and $T_{cor} = 0.7086$, the RCM analysis leads to $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \{f_1, f_2, f_3\}$ (Table 4.4b).
- Finally, based on the selection scores of f_1, f_2 and f_3 (Table 4.4c), f_3 emerges as the representative in each of $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, leading to $\mathcal{F}_s = \{f_3, f_4, f_5\}$ with $\mathcal{E}_0 = c_1^M(1.0 - R_{13}) + c_2^M(1.0 - R_{23}) = 0.0$.
- Iteration-2 of NL-MVU-PCA fails to further reduce any objectives, and hence is not shown here. In effect, $\mathcal{F}_{\mathcal{S}}$ is nothing but $\mathcal{F}_s$ in iteration-1.

4.4.1.3 Key Inferences from the Analysis on $\mathcal{N}_{\mathcal{P}}$

It can be seen in Fig. 4.4 for $\mathcal{N}_{\mathcal{P}}$ that: (i) f_1–f_2–f_3 are correlated, and (ii) the variance of f_1 and f_2 is equal, and less than the variance of f_3. Given a set of correlated objectives, both L-PCA and NL-MVU-PCA pick a representative objective in the

Table 4.4 DTLZ5(3, 5): iteration-1 of NL-MVU-PCA with $\mathcal{N}_{\mathcal{P}}$ (taken from [34])

(a) Eigenvalue Analysis

PCA	Variance	Cumulative	Objectives Selected				
(N^v)	(%)	(%)	f_1	f_2	f_3	f_4	f_5
1	51.38	51.38	f_1	f_2	f_3	f_4	f_5
2	48.55	99.93	f_1	f_2	f_3		f_5

(b) RCM Analysis

1) Potentially correlated subsets	Eq 4.4(i)	$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_2 = \hat{\mathcal{S}}_3 = \{f_1, f_2, f_3\}$
2) Correlation threshold: T_{cor}	Eq 4.5	1.0-0.5138(1.0-2/5) = 0.7086
3) Correlated subsets	Eq 4.4(ii)	$\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \{f_1, f_2, f_3\}$

(c) Selection scheme

	e_1=0.5138	e_2=0.4855	sc_i
	V_1	V_2	
f_1	0.134	0.273	0.2017
f_2	0.134	0.273	0.2017
f_3	0.211	0.426	0.3154

set. To this end, these algorithms compute the variance contribution of each correlated objective, along the significant principal components, and this is known as the selection score (sc_i, Eq. 4.6). Hence, for $\mathcal{N}_{\mathcal{P}}$, these algorithms should determine $sc_3 > sc_2 = sc_1$. However, unlike NL-MVU-PCA (Table 4.4), L-PCA fails to identify this fact (Table 4.3) plausibly owing to its inability to accurately determine the principal components in the first place. This is evident from Table 4.2c, where the significant principal components based on R are such that the variance contributions of f_1, f_2 and f_3 are equal. In contrast, in the case of NL-MVU-PCA, the significant principal components based on K (Table 4.2c) accurately capture the variance contributions of f_1, f_2 and f_3.

4.4.2 Performance on Solution Sets with Inexact PF-Approximation

The relative capabilities of L-PCA and NL-MVU-PCA were demonstrated above with respect to an ideal data set, $\mathcal{N}_{\mathcal{P}}$, directly sampled on the true *PF* (symbolizing the *unnoised* signal). This section seeks to demonstrate the relative capabilities of these algorithms with realistic data sets offered by EMâOAs, where the quality of the *PF*-approximation may or may not be good (symbolizing a combination of *unnoised* and *noised* signals). To do that, $\mathcal{N}_{\epsilon}$ and $\mathcal{N}_{\mathcal{NS}}$, have been considered.

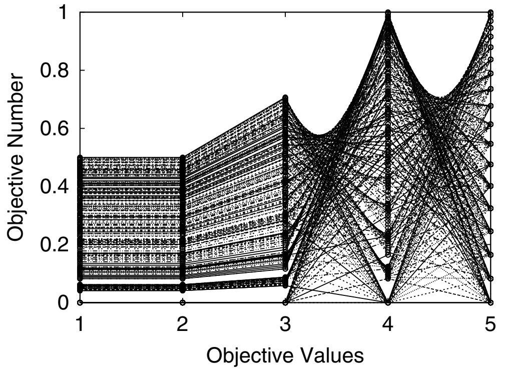


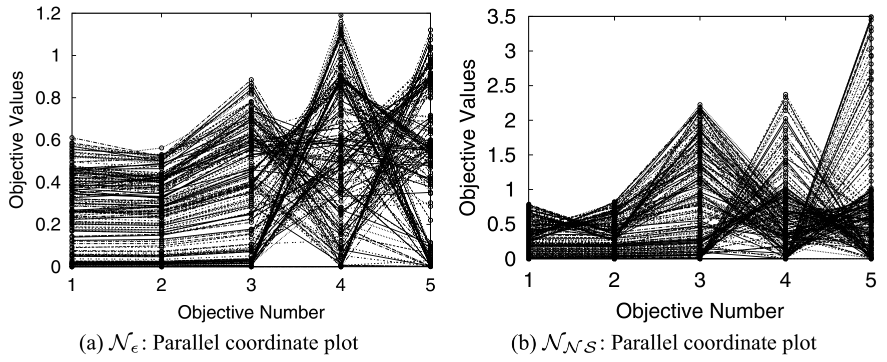
Fig. 4.4 DTLZ5(3, 5): Parallel coordinate plot for $\mathcal{N}_{\mathcal{P}}$ (taken from [34])

It may be noted with respect to $\mathcal{N}_{\mathcal{P}}$ (Fig. 4.4), that over the true *PF*: (i) the range of—f_1 and f_2 is [0 : 0.5]; f_3 is [0 : 0.707]; f_4 and f_5 is [0 : 1], and (ii) f_1-f_2-f_3 are correlated, while f_3-f_4-f_5 are in conflict. In contrast, Fig. 4.5a and b suggest that both $\mathcal{N}_{\epsilon}$ and $\mathcal{N}_{\mathcal{NS}}$ fail to approximate *PF* well, since both fail to:

- converge to *PF*: this is evident by larger ranges for the objectives.
- capture the correlation structure of *PF*: this is evident by *some* crossing lines between f_1-f_2 and f_2-f_3, symbolizing partial conflict even among f_1-f_2-f_3.

However, based on the degree of departure from the true *PF* characteristics, it can be inferred that $\mathcal{N}_{\mathcal{NS}}$ offers a worse *PF*-approximation than $\mathcal{N}_{\epsilon}$.

In this background, Iteration-1 of L-PCA and NL-MVU-PCA, based on $\mathcal{N}_{\epsilon}$, as summarized in a self explanatory Table 4.5 may be noted. In that, L-PCA inaccurately leads to $\mathcal{F}_s = \{f_1, f_4, f_5\}$, while NL-MVU-PCA accurately leads to $\mathcal{F}_s = \{f_3, f_4, f_5\}$. For both algorithms, Iteration-2 fails to reduce any objectives fur-



(a) $\mathcal{N}_{\epsilon}$: Parallel coordinate plot (b) $\mathcal{N}_{\mathcal{NS}}$: Parallel coordinate plot

Fig. 4.5 DTLZ5(3, 5): Parallel coordinate plots for $\mathcal{N}_{\epsilon}$ and $\mathcal{N}_{\mathcal{NS}}$ (taken from [34])

Table 4.5 DTLZ5(3, 5): Iteration-1 of L-PCA and NL-MVU-PCA with $\mathcal{N}_\epsilon$ (taken from [34])

(a) Correlation matrix (R)

	f_1	f_2	f_3	f_4	f_5
f_1	1.000	0.987	0.973	-0.431	-0.338
f_2	0.987	1.000	0.964	-0.449	-0.304
f_3	0.973	0.964	1.000	-0.468	-0.322
f_4	-0.431	-0.449	-0.468	1.000	-0.513
f_5	-0.338	-0.304	-0.322	-0.513	1.000

(b) Eigenvalues and eigenvectors of R, and selection scores of objectives

	e_1=0.6567	e_2=0.3007	e_3=0.0333	e_4=0.0069	e_5=0.0021	sc_i
	V_1	V_2	V_3	V_4	V_5	
f_1	0.546	-0.056	-0.219	0.225	-0.775	0.386
f_2	0.544	-0.030	-0.274	0.505	0.610	0.380
f_3	0.545	-0.030	-0.059	-0.820	0.165	0.375
f_4	-0.285	-0.660	-0.686	-0.114	0.007	0.409
f_5	-0.166	0.748	-0.635	-0.097	-0.020	0.356

(c) Eigenvalues and eigenvectors of K, and selection scores of objectives

	e_1=0.5322	e_2=0.4608	e_3=0.0057	e_4=0.0016	$e_5 \approx 0$	sc_i
	V_1	V_2	V_3	V_4	V_5	
f_1	0.059	0.269	-0.376	-0.764	-0.447	0.158
f_2	0.076	0.282	-0.553	0.639	-0.447	0.174
f_3	0.142	0.490	0.730	0.086	-0.447	0.305
f_4	-0.821	-0.337	0.108	0.027	-0.447	0.593
f_5	0.545	-0.704	0.091	0.011	-0.447	0.615

(d) Key highlights of L-PCA and NL-MVU-PCA

Features	L-PCA	NL-MVU-PCA
N_v; $M_{2\sigma}$; $\mathcal{F}_e$	4; 2; $\{f_1, f_2, f_3, f_4, f_5\}$	3; 2; $\{f_1, f_2, f_3, f_4, f_5\}$
$\hat{\mathcal{S}}_i$	$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_2 = \hat{\mathcal{S}}_3 = \{f_1, f_2, f_3\}$	
	$R_{12} = 0.987$, $R_{13} = 0.973$ and $R_{23} = 0.964$	
T_{cor}	0.6059	0.6806
$\mathcal{S}_i$	$\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \{f_1, f_2, f_3\}$	
$\mathcal{F}_s$	$\{f_1, f_4, f_5\}$	$\{f_3, f_4, f_5\}$
$\mathcal{E}_0$	0.007766	0.002455

ther, and is not shown here for the sake of brevity. To summarize, L-PCA leads to $\mathcal{F}_{\mathcal{S}} = \{f_1, f_4, f_5\}$, while NL-MVU-PCA leads to $\mathcal{F}_{\mathcal{S}} = \{f_3, f_4, f_5\}$.

Similarly, Iteration-1 of L-PCA and NL-MVU-PCA, based on $\mathcal{N}_{\mathcal{NS}}$, as summarized in a self explanatory Table 4.6 may be noted. In that, L-PCA fails to reduce any objective, and hence, no further Iterations can help. In contrast, NL-MVU-PCA reduces to $\mathcal{F}_s = \{f_1, f_3, f_4, f_5\}$. This necessitates Iteration-2, presented in Table 4.7. In that, $\{f_1, f_3, f_4, f_5\}$ serves as the set of input objectives, which, for ease of reference, is denoted as $\{F_1, F_2, F_3, F_4\}$. Evidently, Iteration-2 leads to $\mathcal{F}_s = \{f_3, f_4, f_5\}$, following which Iteration-3 is carried out. However, no further reduction is observed, hence, Iteration-3 is not reported for the sake of brevity. To summarize, NL-MVU-PCA leads to $\mathcal{F}_{\mathcal{S}} = \{f_3, f_4, f_5\}$.

The above results for L-PCA and NL-MVU-PCA, based on $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$, respectively, can naturally be explained on the basis of an interplay of algorithmic capability and quality of the dataset as follows:

- NL-MVU-PCA performs better than L-PCA, in each case of $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$. This could be attributed to the benefits that MVU facilitates progress toward removal of higher order dependencies in the data sets.
- Each of NL-MVU-PCA and L-PCA performs better with $\mathcal{N}_\epsilon$, than with $\mathcal{N}_{\mathcal{NS}}$. This could be attributed to the better *PF*-approximation by $\mathcal{N}_\epsilon$ than by $\mathcal{N}_{\mathcal{NS}}$.

4.5 Results on a Wider Range of Redundant Problems

This section presents the experimental results for a wider range of the DTLZ5(I, M) and WFG3(M) problems, with respect to $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$. Here, the performance of L-PCA and NL-MVU-PCA is compared to that of the Dominance-Relation-Preservation (DRP)-based 0-MOSS analysis (δ-MOSS with $\delta = 0$) by both greedy and exact algorithms [2, 3].

The results presented in Tables 4.8 and 4.9, reveal that the DRP-based greedy and exact algorithms fail to identify an *essential* objective set ($\mathcal{F}_{\mathcal{S}}$), for all the problems. Their failure, even with $\delta = 0$, could plausibly be attributed to the fact that the underlying data sets are *noisy*, and hence the inferences drawn do not accurately characterize the true *PF*. In general, for $\delta \neq 0$, the limitations of the DRP-based algorithms are linked to the underlying stringent assumptions [3, 16], including:

- δ error across all solutions is comparable: this assumption could be violated in situations where the objective function values are not equally distributed.
- δ error is comparable across all the objectives for any two given solutions: this assumption is likely to be violated unless the objectives are either linear or identically nonlinear.

Table 4.6 DTLZ5(3, 5): Iteration-1 of L-PCA and NL-MVU-PCA with $\mathcal{N}_{NS}$ (taken from [34])

(a) Correlation matrix (R)

	f_1	f_2	f_3	f_4	f_5
f_1	1.000	0.566	0.688	-0.285	-0.425
f_2	0.566	1.000	0.851	-0.244	-0.390
f_3	0.688	0.851	1.000	-0.265	-0.286
f_4	-0.285	-0.244	-0.265	1.000	-0.475
f_5	-0.425	-0.390	-0.286	-0.475	1.000

(b) Eigenvalues and eigenvectors of R, and selection scores of objectives

	e_1=0.5392	e_2=0.2938	e_3=0.0983	e_4=0.0516	e_5=0.0168	sc_i
	V_1	V_2	V_3	V_4	V_5	
f_1	0.515	-0.018	0.706	-0.335	0.352	0.376
f_2	0.548	-0.025	-0.527	0.325	0.563	0.381
f_3	0.561	-0.092	-0.332	-0.459	-0.596	0.396
f_4	-0.168	0.745	-0.246	-0.531	0.274	0.366
f_5	-0.303	-0.660	-0.231	-0.538	0.360	0.414

(c) Eigenvalues and eigenvectors of K, and selection scores of objectives

	e_1=0.5875	e_2=0.3440	e_3=0.0493	e_4=0.0191	e_5=$\approx$ 0	sc_i
	V_1	V_2	V_3	V_4	V_5	
f_1	0.108	-0.042	-0.732	-0.500	0.447	0.124
f_2	0.144	0.009	-0.278	0.838	0.447	0.118
f_3	0.548	0.509	0.453	-0.188	0.447	0.523
f_4	0.017	-0.798	0.391	-0.103	0.447	0.305
f_5	-0.817	0.321	0.166	-0.046	0.447	0.599

(d) Key highlights of L-PCA and NL-MVU-PCA

Features	L-PCA	NL-MVU-PCA
N_v; $M_{2\sigma}$; $\mathcal{F}_e$	5; 4; $\{f_1, f_2, f_3, f_4, f_5\}$	4; 3; $\{f_1, f_2, f_3, f_4, f_5\}$
$\hat{\mathcal{S}}_i$	$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_2 = \hat{\mathcal{S}}_3 = \{f_1, f_2, f_3\}$	
	$R_{12} = 0.566$, $R_{13} = 0.688$ and $R_{23} = 0.851$	
T_{cor}	0.8921	0.7649
$\mathcal{S}_i$	$\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3 = \emptyset$	$\mathcal{S}_1 = \emptyset$; $\mathcal{S}_2 = \mathcal{S}_3 = \{f_2, f_3\}$
$\mathcal{F}_s$	$\{f_1, f_2, f_3, f_4, f_5\}$	$\{f_1, f_3, f_4, f_5\}$
$\mathcal{E}_0$	0.0	0.00439

Table 4.7 DTLZ5(3, 5): Iteration-2 of NL-MVU-PCA based on $\mathcal{N}_{\mathcal{NS}}$ (taken from [34])

(a) Correlation matrix (R)

	$F_1 \equiv f_1$	$F_2 \equiv f_3$	$F_3 \equiv f_4$	$F_4 \equiv f_5$
F_1	1.000	0.751	-0.320	-0.385
F_2	0.751	1.000	-0.282	-0.222
F_3	-0.320	-0.282	1.000	-0.562
F_4	-0.385	-0.222	-0.562	1.000

(b) Eigenvalues/vectors of K, and selection scores of objectives

	e_1	e_2	e_3	e_4	
	0.6091	0.3582	0.0326	≈ 0	sc_i
	V_1	V_2	V_3	V_4	
F_1	0.087	0.126	0.852	0.500	0.126
F_2	0.155	0.742	-0.419	0.500	0.374
F_3	0.564	-0.603	-0.262	0.500	0.568
F_4	-0.806	-0.265	-0.172	0.500	0.592

(c) Key highlights of NL-MVU-PCA

Features	NL-MVU-PCA
N_v; $M_{2\sigma}$; $\mathcal{F}_e$; T_{cor}	3; 2; $\{F_1, F_2, F_3, F_4\}$; 0.6954
$\hat{\mathcal{S}}_i$	$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_2 = \{F_1, F_2\}$, where $R_{12} = 0.751$
$\mathcal{S}_i$	$\mathcal{S}_1 = \mathcal{S}_2 = \{F_1, F_2\}$
$\mathcal{F}_s$	$\{F_2, F_3, F_4\} = \{f_3, f_4, f_5\}$

In terms of L-PCA and NL-MVU-PCA, Tables 4.8 and 4.9 show that, in the case of the DTLZ5(I, M) problems:

1. The performance of NL-MVU-PCA is better than that of L-PCA, corresponding to both $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$. This could again be attributed to the underlying nonlinear unfolding, given which the principal components are more accurately determined (Sect. 4.4.1).
2. The performance of both NL-MVU-PCA and L-PCA on $\mathcal{N}_\epsilon$ is better than their respective performance on $\mathcal{N}_{\mathcal{NS}}$. This could be attributed to the fact that the PF-approximation by $\mathcal{N}_\epsilon$ is better than that by $\mathcal{N}_{\mathcal{NS}}$. Its justification lies in the better convergence and diversity measures corresponding to $\mathcal{N}_\epsilon$ than those corresponding to $\mathcal{N}_{\mathcal{NS}}$. This is further endorsed by the fact that, compared to $\mathcal{N}_{\mathcal{NS}}$, the T_{cor} values based on $\mathcal{N}_\epsilon$ have a smaller deviation from the T_{cor} values based on $\mathcal{N}_{\mathcal{P}}$ (Fig. 4.6b and c).

Furthermore, in the case of WFG3(M) problems:

Table 4.8 Redundant problems: Results based on $\mathcal{N}_{\mathcal{NS}}$ and $\mathcal{N}_\epsilon$ with $\theta = 0.997$. The entries below I and $\mathcal{F}_\mathcal{S}$ indicate the number of successful cases in the 20 runs. The dashes (-) imply inconsequential entries where the prerequisite of accurate I is not met. The footnotes report cases that require multiple iterations of the algorithm for accurate results, where in P—aR(bI) implies that for the problem P, a Runs out of 20, required b iterations each (taken from [34])

	Test problems	Presented approaches								DRP [2, 3]; δ-MOSS, 0% error							
		NL-MVU-PCA				L-PCA				Greedy approach				Exact approach			
	DTLZ5(I, M)	$\mathcal{N}_\epsilon$		$\mathcal{N}_{\mathcal{NS}}$		$\mathcal{N}_\epsilon$		$\mathcal{N}_{\mathcal{NS}}$		$\mathcal{N}_\epsilon$		$\mathcal{N}_{\mathcal{NS}}$		$\mathcal{N}_\epsilon$		$\mathcal{N}_{\mathcal{NS}}$	
I	M	Ib	$\mathcal{F}_\mathcal{S}$	Ic	$\mathcal{F}_\mathcal{S}$	Id	$\mathcal{F}_\mathcal{S}$	I	$\mathcal{F}_\mathcal{S}$	I	$\mathcal{F}_\mathcal{S}$	I	$\mathcal{F}_\mathcal{S}$	I	$\mathcal{F}_\mathcal{S}$	I	$\mathcal{F}_\mathcal{S}$
2	5	20	20	20	20	20	14	20	1	0	–	0	–	0	–	0	–
2	20	20	20	7	7	20	2	0	–	0	–	0	–	0	–	0	–
2	50	20	20	14	14	20	0	10	0	0	–	0	–	0	–	0	–
3	5	20	20	18	18	20	9	0	–	0	–	0	–	0	–	0	–
3	20	20	20	0	–	20	1	0	–	0	–	0	–	0	–	0	–
5	10	19	19	0	–	19	3	0	–	0	–	0	–	0	–	0	–
5	20	19	19	0	–	18	3	0	–	0	–	0	–	0	–	0	–
7	10	19	19	0	–	20	6	0	–	0	–	0	–	0	–	0	–
7	20	16	16	0	–	13	3	0	–	0	–	0	–	0	–	0	–
WFG3	5	20	20	20	20	20	19	20	4	0	–	0	–	0	–	0	–
	15	15	15	20	20	10	1	19	0	0	–	0	–	0	–	0	–
	25	9	9	20	20	6	0	20	0	0	–	0	–	0	–	0	–

[a] The tabulated results are obtained using the source codes at: http://www.tik.ee.ethz.ch/sop/download/supplementary/objectiveReduction/. The same codes are used for the results presented later in Tables 4.10–4.13

[b] DTLZ5: (5, 10)—3R(2I); (5, 20)—8R(2I); (7, 10)—7R(2I); (7, 20)—9R(2I) & 1R(4I); and WFG3: (15)—6R(2I) & 3R(3I); (25)—5R(2I), 1R(3I) & 1R(4I)

[c] DTLZ5: (2, 20)—7R(2I), 2R(4I), 2R(5I), 1R(6I) & 1R(7I); (2, 50)—8R(2I) & 2R(3I); (3, 5)—9R(2I)

[d] DTLZ5: (5, 20)—1R(2I); (7, 10)—3R(2I); (7, 20)—1R(2I); and WFG3: (5)—11R(2I); (15)—1R(2I).

1. The performance of NL-MVU-PCA is better than that of L-PCA, for both $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$.
2. Contrary to the case of DTLZ5(I, M) problems, the performance of both NL-MVU-PCA and L-PCA is better, corresponding to $\mathcal{N}_{\mathcal{NS}}$, than with $\mathcal{N}_\epsilon$. Although this difference in performance may seem significant in terms of the frequency of success in 20 runs (Table 4.8), it can be seen from Table 4.9 that this difference is only marginal in terms of the average dimension of PF in 20 runs. However, these results imply that the PF-approximation by $\mathcal{N}_{\mathcal{NS}}$ should be better than that by $\mathcal{N}_\epsilon$. This is validated by the fact that, compared to $\mathcal{N}_\epsilon$, the T_{cor} values corresponding to $\mathcal{N}_{\mathcal{NS}}$ are closer to those of $\mathcal{N}_\mathcal{P}$ (Fig. 4.6b and c).

Besides the above results, the following issues are noteworthy:

- Given the goal of T_{cor} (Sect. 4.2.4), a low and a high value of T_{cor} are desired for problems with high and low redundancy, respectively. The fact that the given formulation (Eq. 4.5) fulfills its goal is confirmed by Fig. 4.6b and c, where cor-

Table 4.9 Redundant problems: Results based on $\mathcal{N}_{\mathcal{NS}}$ and $\mathcal{N}_{\epsilon}$ with $\theta = 0.997$. The entries are formatted as $\mu \pm \sigma$, where μ and σ represent the mean and standard deviation of the number of essential objectives identified, in 20 runs (taken from [34])

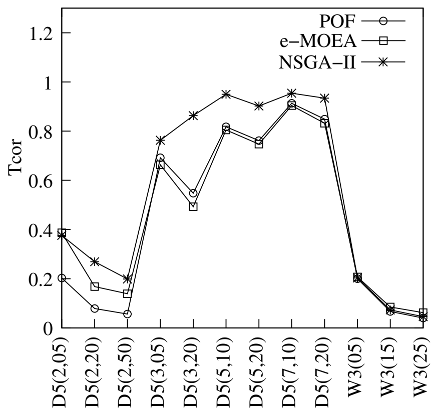
Test problems		Presented approaches				DRP [2, 3]; δ-MOSS, $\delta = 0$			
	DTLZ5(I,M)	NL-MVU-PCA		L-PCA		Greedy		Exact	
I	M	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$
2	5	02.0 ± 0.0	02.0 ± 0.0	02.0 ± 0.0	02.0 ± 0.0	05.0 ± 0.0	04.7 ± 0.5	05.0 ± 0.0	05.0 ± 0.0
2	20	02.0 ± 0.0	10.9 ± 7.8	02.0 ± 0.0	18.3 ± 2.2	16.0 ± 1.4	11.8 ± 1.8	15.5 ± 1.4	11.3 ± 1.5
2	50	02.0 ± 0.0	04.4 ± 3.8	02.0 ± 0.0	05.2 ± 4.2	19.2 ± 2.2	10.9 ± 1.2	18.9 ± 1.2	10.5 ± 1.0
3	5	03.0 ± 0.0	03.2 ± 0.6	03.0 ± 0.0	05.0 ± 0.0	05.0 ± 0.0	04.9 ± 0.3	05.0 ± 0.0	04.9 ± 0.2
3	20	03.0 ± 0.0	19.7 ± 0.8	03.0 ± 0.0	19.8 ± 0.3	13.1 ± 1.5	10.6 ± 1.2	12.2 ± 1.4	10.3 ± 1.1
5	10	05.0 ± 0.2	10.0 ± 0.0	05.0 ± 0.2	10.0 ± 0.0	09.1 ± 0.6	09.4 ± 0.5	08.7 ± 0.6	09.3 ± 0.4
5	20	05.0 ± 0.2	20.0 ± 0.0	05.1 ± 0.3	20.0 ± 0.0	10.7 ± 1.3	10.3 ± 1.0	09.7 ± 1.3	09.8 ± 1.0
7	10	07.0 ± 0.2	10.0 ± 0.0	07.0 ± 0.0	10.0 ± 0.0	08.8 ± 0.8	09.4 ± 0.5	08.4 ± 0.9	09.3 ± 0.4
7	20	07.2 ± 0.4	20.0 ± 0.0	07.3 ± 0.4	20.0 ± 0.0	10.3 ± 1.4	10.7 ± 0.9	09.6 ± 1.6	10.5 ± 0.6
WFG3	5	02.0 ± 0.0	02.0 ± 0.0	02.0 ± 0.0	02.0 ± 0.0	05.0 ± 0.0	04.7 ± 0.4	04.5 ± 0.5	04.1 ± 0.4
	15	02.2 ± 0.4	02.0 ± 0.0	02.7 ± 0.8	02.0 ± 0.2	05.7 ± 0.7	05.6 ± 0.5	04.9 ± 0.9	04.6 ± 0.5
	25	02.7 ± 0.7	02.0 ± 0.0	03.1 ± 0.9	02.0 ± 0.0	05.4 ± 0.5	05.4 ± 0.6	04.4 ± 0.6	04.6 ± 0.6

responding to $\mathcal{N}_{\mathcal{P}}$: (i) for a fixed I in DTLZ5(I, M) problems, as M increases (higher redundancy) the T_{cor} decreases, and (ii) for a fixed M, as I increases (lesser redundancy) the T_{cor} increases. The same trend holds for the WFG3(M) problems.

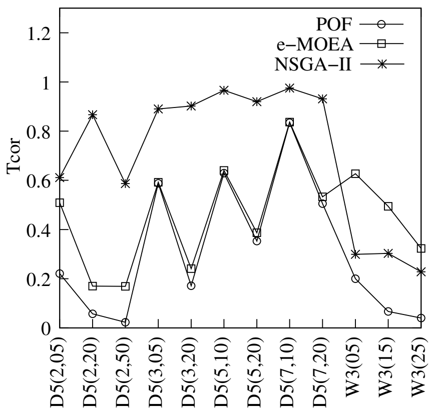
- The instances where more than one iteration is required are those in which: (i) the underlying solution set does not reveal *all* the identically correlated objectives, or (ii) the strength of correlation between the identically correlated objectives happens to be weaker than the computed T_{cor} (Sect. 4.2.4). In such cases, only some of the redundant objectives are eliminated in the first iteration, while the remaining ones disappear in subsequent iterations.

D5		Convergence (g)		Diversity (I_s)	
		$\mathcal{N}_\epsilon$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_\epsilon$	$\mathcal{N}_{\mathcal{NS}}$
I	M	($\mu\pm\sigma$)	($\mu\pm\sigma$)	($\mu\pm\sigma$)	($\mu\pm\sigma$)
2	05	0.15±0.09	0.48±0.63	1.93±0.01	4.20±0.17
2	20	0.20±0.07	2.10±0.58	1.91±0.02	8.06±0.17
2	50	0.23±0.08	2.34±0.34	1.99±0.02	8.18±0.22
3	05	0.08±0.04	0.70±0.61	1.48±0.00	3.54±0.02
3	20	0.17±0.07	2.25±0.03	1.59±0.02	6.67±0.01
5	10	0.14±0.07	2.06±0.34	1.38±0.02	4.29±0.02
5	20	0.15±0.07	2.25±0.31	1.37±0.00	5.00±0.01
7	10	0.16±0.07	1.99±0.38	1.27±0.01	3.30±0.01
7	20	0.16±0.08	2.17±0.38	1.28±0.00	3.94±0.02

(a) g (convergence) and I_s (diversity) measures



(b) T_{cor}: NL-MVU-PCA



(c) T_{cor}: L-PCA

Fig. 4.6 Redundant problems: reflecting on the quality of PF-approximation offered by $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$. For brevity, DTLZ5($I$, M) and WFG3(M) are abbreviated as D5(I, M) and W3(M), respectively. The values of g, I_s and T_{cor} are averaged over 20 runs (taken from [34])

4.6 Results on Non-redundant Problems

This section presents the results for the non-redundant problems, DTLZ1(M) to DTLZ4(M). The PF for these problems is constituted by M conflicting objectives, implying that no two objectives are correlated on the PF. The results corresponding to $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$ are presented in Table 4.10, where it can be seen that:

- Both the DRP-based algorithms fail to identify the true dimension ($m = M$) of the PF, except for the five-objective instances of the problems.
- Both NL-MVU-PCA and L-PCA corresponding to $\mathcal{N}_{\mathcal{NS}}$ accurately identify $m = M$ for all problems. However, their performance corresponding to $\mathcal{N}_\epsilon$ is poor for some problems.

Table 4.10 Non-redundant problems: Results based on $\mathcal{N}_{\mathcal{NS}}$ and $\mathcal{N}_{\epsilon}$ with $\theta = 0.997$. DTLZk(M) represents an M-objective DTLZk problem. The entries are formatted as $n(\mu \pm \sigma)$, where n indicates the number of times $\mathcal{F}_{\mathcal{S}}$ is accurately identified, in 20 runs, while μ and σ represent the mean and standard deviation of the number of essential objectives identified, in 20 runs (taken from [34])

DTLZk(M)	Presented approaches				DRP [2, 3]; δ-MOSS, $\delta = 0$			
	NL-MVU-PCA		L-PCA		Greedy approach		Exact approach	
	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$	$\mathcal{N}_{\epsilon}$	$\mathcal{N}_{\mathcal{NS}}$
DTLZ1(05)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)
DTLZ1(15)	06 (14.1 $\pm$ 0.7)	20 (15.0 $\pm$ 0.0)	14 (14.7 $\pm$ 0.4)	20 (15.0 $\pm$ 0.0)	08 (13.8 $\pm$ 1.0)	00 (11.5 $\pm$ 0.9)	07 (13.7 $\pm$ 1.0)	00 (10.5 $\pm$ 0.5)
DTLZ1(25)	00 (20.5 $\pm$ 2.2)	20 (25.0 $\pm$ 0.0)	03 (22.2 $\pm$ 2.1)	20 (25.0 $\pm$ 0.0)	00 (16.3 $\pm$ 1.6)	00 (11.9 $\pm$ 1.0)	00 (16.2 $\pm$ 1.4)	00 (11.6 $\pm$ 0.9)
DTLZ2(05)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (04.9 $\pm$ 0.2)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)
DTLZ2(15)	20 (15.0 $\pm$ 0.0)	20 (15.0 $\pm$ 0.0)	20 (15.0 $\pm$ 0.0)	20 (15.0 $\pm$ 0.0)	01 (11.4 $\pm$ 0.9)	00 (11.0 $\pm$ 0.6)	00 (10.8 $\pm$ 0.7)	00 (10.7 $\pm$ 0.7)
DTLZ2(25)	16 (24.7 $\pm$ 0.5)	20 (25.0 $\pm$ 0.0)	17 (24.7 $\pm$ 0.5)	20 (25.0 $\pm$ 0.0)	00 (12.4 $\pm$ 1.0)	00 (11.3 $\pm$ 0.8)	00 (12.1 $\pm$ 1.0)	00 (11.2 $\pm$ 0.9)
DTLZ3(05)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)
DTLZ3(15)	11 (14.3 $\pm$ 0.9)	20 (15.0 $\pm$ 0.0)	20 (14.8 $\pm$ 0.4)	20 (15.0 $\pm$ 0.0)	08 (13.1 $\pm$ 0.9)	00 (12.9 $\pm$ 1.0)	02 (12.7 $\pm$ 0.9)	00 (12.3 $\pm$ 0.8)
DTLZ3(25)	00 (20.9 $\pm$ 2.2)	20 (25.0 $\pm$ 0.0)	05 (22.0 $\pm$ 2.6)	20 (25.0 $\pm$ 0.0)	00 (15.5 $\pm$ 1.5)	00 (13.8 $\pm$ 1.9)	00 (14.8 $\pm$ 1.2)	00 (13.6 $\pm$ 1.7)
DTLZ4(05)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	14 (04.7 $\pm$ 0.4)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)	20 (05.0 $\pm$ 0.0)
DTLZ4(15)	20 (15.0 $\pm$ 0.0)	20 (15.0 $\pm$ 0.0)	20 (15.0 $\pm$ 0.0)	20 (15.0 $\pm$ 0.0)	00 (11.6 $\pm$ 0.8)	00 (11.2 $\pm$ 0.7)	00 (11.0 $\pm$ 0.6)	00 (11.5 $\pm$ 1.0)
DTLZ4(25)	20 (25.0 $\pm$ 0.0)	20 (25.0 $\pm$ 0.0)	20 (25.0 $\pm$ 0.0)	20 (25.0 $\pm$ 0.0)	00 (11.4 $\pm$ 0.8)	00 (12.0 $\pm$ 0.8)	00 (11.2 $\pm$ 0.7)	00 (12.2 $\pm$ 0.7)

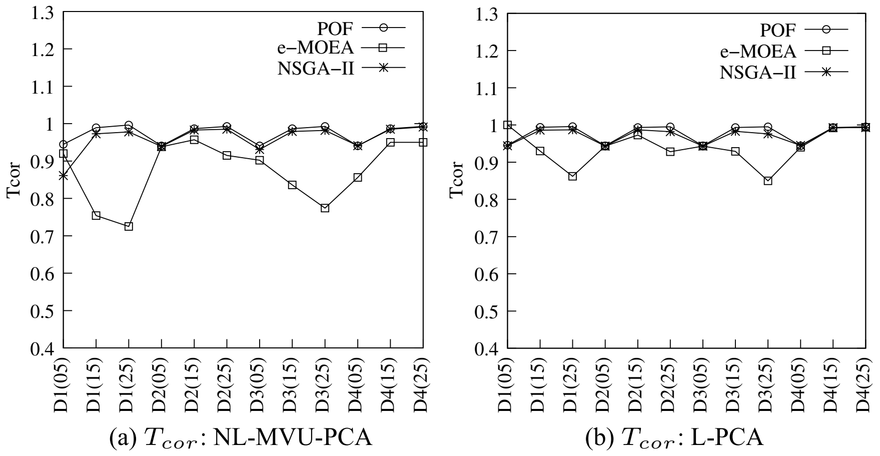


Fig. 4.7 Non-redundant problems: reflecting on the quality of PF-approximation offered by $\mathcal{N}_\epsilon$ and $\mathcal{N}_{\mathcal{NS}}$. For the sake of brevity, Dk(M) represents an M-objective DTLZk problem. The T_{cor} values are averaged over 20 runs (taken from [34])

As in the previous section, the performance of NL-MVU-PCA and L-PCA reported in Table 4.10 should be interpreted through the interplay of algorithmic capability and quality of the dataset. Here: (i) NL-MVU-PCA by construction promises to be more accurate than L-PCA, and (ii) in contrast to the case of redundant problems, the quality of PF-approximation in the case of non-redundant problems seems to be better for $\mathcal{N}_{\mathcal{NS}}$, than $\mathcal{N}_\epsilon$. This is supported by Fig. 4.7, where it can be seen that compared to $\mathcal{N}_\epsilon$, the values of T_{cor} corresponding to $\mathcal{N}_{\mathcal{NS}}$ are in stronger conformity with those based on $\mathcal{N}_{\mathcal{P}}$.

4.7 Decision Support for Real-World Problems

This section considers two real-world problems and demonstrates the utility of envisioned decision support in terms of the revelation of an essential/redundant objective set, δ-MOSS and k-EMOSS analysis. For both of these problems, the reported results are restricted to NL-MVU-PCA (L-PCA- and DRP-based approaches are excluded), since it emerged as the most accurate algorithm based on the test problems.

4.7.1 Storm Drainage System Problem

This is a five-objective, seven-constraint problem [28] which relates to optimal planning for a storm drainage system in an urban area. NSGA-II is run for 30 uniformly

Table 4.11 Storm drainage: analysis based on $\mathcal{N}_{\mathcal{NS}}$ (one run; iteration-1)

(a) Correlation matrix (R)

	f_1	f_2	f_3	f_4	f_5
f_1	1.0000	-0.2689	0.9938	-0.8966	-0.2000
f_2	-0.2689	1.0000	-0.2532	0.2644	-0.6915
f_3	0.9938	-0.2532	1.0000	-0.9160	-0.2220
f_4	-0.8966	0.2644	-0.9160	1.0000	0.2864
f_5	-0.2000	-0.6915	-0.2220	0.2864	1.0000

(b) Eigenvalues/vectors of K, and selection scores of objectives

	$e_1 = 0.9875$	$e_2 = 0.0124$	$e_3 \approx 0$	$e_4 \approx 0$	$e_5 \approx 0$	sc_i
	V_1	V_2	V_3	V_4	V_5	
f_1	0.1275	-0.3407	0.3785	0.7240	0.4472	0.1301
f_2	0.1262	-0.3452	0.4361	-0.6889	0.4472	0.1289
f_3	0.4739	0.7585	0.0002	-0.0028	0.4472	0.4774
f_4	-0.8529	0.2692	0.0013	-0.0000	0.4472	0.8456
f_5	0.1252	-0.3417	-0.8163	-0.0322	0.4472	0.1279

(c) Key highlights of NL-MVU-PCA

$N_v = 2$	$M_{2\sigma} = 1$	$\mathcal{F}_e\{f_1, f_2, f_3, f_4, f_5\}$
$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_3 = \{f_1, f_3\}$	$R_{13} = 0.9938$	$T_{cor} = 0.2099$
$\mathcal{S}_1 = \mathcal{S}_3 = \{f_1, f_3\}$	$sc_i(f_3) > sc_i(f_1)$	$\mathcal{F}_s = \{f_2, f_3, f_4, f_5\}$
$\mathcal{F}_r = \{f_1\}$	$\mathcal{E}_t^r = \{1.07\text{E-04}\}$	$\eta_t^r = \{1.09\text{E-04}\}$

distributed seeds, where each run corresponds to a population size of 200 and 2000 generations. The resulting $\mathcal{N}_{\mathcal{NS}}$ for one of the sample runs is subjected to NL-MVU-PCA, of which iteration-1 is presented in the self-explanatory Table 4.11. The resulting $\mathcal{F}_s = \{f_2, f_3, f_4, f_5\}$ is then fed as the input for the iteration-2 (Table 4.12), where the involved objectives are referred to as $\{F_1, F_2, F_3, F_4\}$, for ease of referencing.

Evidently, iteration-2 fails to reduce the objectives any further, and therefore NL-MVU-PCA is terminated. Notably, the relevant errors in the two iterations are calculated (as per Sect. 4.2.6) leading to the sorted normalized error list, $\mathcal{SL}$:{1.09E-04, 2.91E-02, 2.94E-02, 2.53E-01, 6.87E-01}, and the corresponding preference ranking of objectives, as $\{f_1 \lhd f_5 \lhd f_2 \lhd f_3 \lhd f_4\}$. The preference structure of the objectives, *learnt* above, paves the way for decision support as follows.

4.7.1.1 Essential Objective Set

$\mathcal{F}_{\mathcal{S}} = \{f_2, f_3, f_4, f_5\}$; $\mathcal{F}_{\mathcal{R}} = \{f_1\}$; and the error associated with omission of $\mathcal{F}_{\mathcal{R}}$ happens to be $\mathcal{E}^R = \mathcal{E}_0^r = 1.07\text{E-04} \equiv 0.0107\%$.

Table 4.12 Storm drainage: analysis based on $\mathcal{N}_{\mathcal{NS}}$ (one run; iteration-2)

(a) Correlation matrix (R)

	$F_1 \equiv f_2$	$F_2 \equiv f_3$	$F_3 \equiv f_4$	$F_4 \equiv f_5$
F_1	1.0000	-0.2464	0.2980	-0.6655
F_2	-0.2464	1.0000	-0.9221	-0.2630
F_3	0.2980	-0.9221	1.0000	0.2964
F_4	-0.6655	-0.2630	0.2964	1.0000

(b) Eigenvalues/vectors of K, and selection scores of objectives

	e_1 = 0.9906	e_2 = 0.0092	$e_3 \approx 0$	$e_4 \approx 0$	sc_i
	V_1	V_2	V_3	V_4	
F_1	0.1662	-0.4721	0.7066	0.5000	0.1690
F_2	0.5013	0.7061	0.0001	0.5000	0.5032
F_3	-0.8328	0.2372	0.0006	0.5000	0.8273
F_4	0.1653	-0.4712	-0.7075	0.5000	0.1681

(c) Key highlights of NL-MVU-PCA

$N_v = 2$	$M_{2\sigma} = 1$	$\mathcal{F}_e\{F_1, F_2, F_3, F_4\}$
$\hat{\mathcal{S}} = \emptyset$	$\mathcal{S} = \emptyset$	$\mathcal{F}_s = \{F_1, F_2, F_3, F_4\} \equiv \{f_2, f_3, f_4, f_5\}$
$\eta_t^s = \{2.94\text{E-}02, 2.53\text{E-}01, 6.87\text{E-}01, 2.91\text{E-}02\}$.		

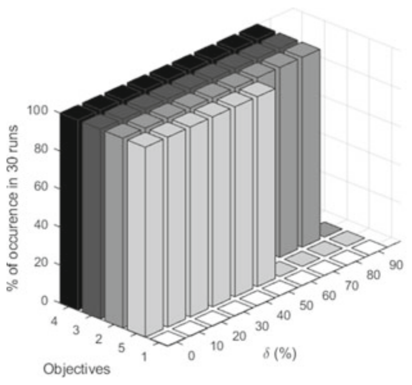
4.7.1.2 δ-MOSS and k-EMOSS Analysis

By treating $\mathcal{SL}$ and the corresponding preference ranking of objectives, across 30 applications of NL-MVU-PCA (corresponding to 30 $\mathcal{N}_{\mathcal{NS}}$) as reference, the δ-MOSS analysis with varying δ, and k-EMOSS analysis with varying k are summarized in Fig. 4.8a and b, respectively, in which the height of each bar indicates the percentage of runs (out of 30) in which a specific objective falls in the reduced objective set for the corresponding δ or k, as applicable. For instance, if the analyst is keen to know:

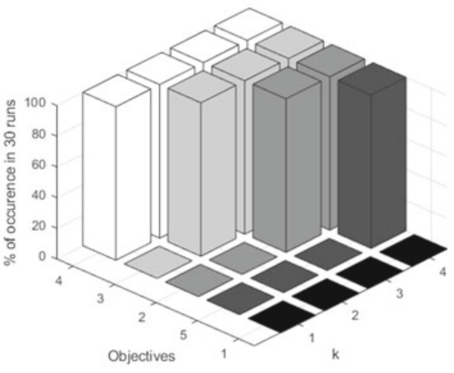
- The smallest subset of objectives ensures that the error associated with the omission of the remaining objectives does not exceed $\delta = 0.1 \equiv 10\%$; then, $\mathcal{F}_{\{0.1\}} = \{f_2, f_3, f_4, f_5\}$. Similarly, for $\delta = 0.7 \equiv 70\%$; $\mathcal{F}_{\{0.7\}} = \{f_2, f_3, f_4\}$.
- The subset of 2 objectives, such that the error associated with omission of the remaining objectives is the minimum, then $\mathcal{F}_{\{k=2\}} = \{f_3, f_4\}$.

4.7.2 *Radar Waveform Problem*

This is a nine-objective, eight-variable problem [22], dealing with the design of a waveform for a Pulsed Doppler Radar, typical of many airborne fighter radar systems. Such systems aim to unambiguously measure the range and velocity of targets that



(a) δ-MOSS: % of occurrence of objectives (varying δ)



(b) k-EMOSS: % of occurrence of objectives (varying k)

Fig. 4.8 Storm drainage problem: visual representation of the δ-MOSS and k-EMOSS analysis by NL-MVU-PCA based on $\mathcal{N}_{\mathcal{NS}}$ over 30 runs

may travel at very high velocities (Mach 5 possible) and may be located at very long distances (100 nautical miles typical). The physical meaning associated with each objective is highlighted in Eq. 4.15, where f_1 to f_8 are to be maximized, while f_9 is to be minimized.

$$\left.\begin{array}{rl} f_1(f_2): & \text{Median range(velocity) of target before schedule is not decodable} \\ f_3(f_4): & \text{Median range(velocity) of target before schedule has blind regions} \\ f_5(f_6): & \text{Minimum range(velocity) of target before schedule is not decodable} \\ f_7(f_8): & \text{Minimum range(velocity) of target before schedule has blind regions} \\ f_9: & \text{Time required to transmit total waveform} \end{array}\right\} \quad (4.15)$$

Interestingly, according to source [22]: (i) the objectives associated with the performance in the range, namely f_1-f_3, and f_5-f_7, tend to have a degree of correlation, just as the objectives associated with the performance in velocity, namely f_2-f_4, and f_6-f_8 do, and (ii) the objectives related to range and velocity conflict. It is critical to note that *objective formulations for this problem are not available in the public domain*. Instead, only non-dominated solution sets corresponding to 30 runs of MSOPS-II [21], each with 20000 function evaluations and sizes ranging from 8839 to 9716 points are available [20].

Given the lack of objective formulations, multiple iterations of NL-MVU-PCA, that naturally culminate in the envisioned decision support are not possible. Hence, the scope here is restricted to the determination of $\mathcal{F}_{\mathcal{S}}$ based on a single application of NL-MVU-PCA, and an *adapted* implementation of δ-MOSS and k-EMOSS analysis. It may be recalled that the δ-MOSS and k-EMOSS analysis rely on the sorted normalized error list, $\mathcal{SL}$, constituted by the normalized errors for the redundant objectives in the intermittent iterations, and the normalized errors for the essential objectives in the final/terminating iteration. In the current situation, errors for all objectives in a single iteration are made to contribute to $\mathcal{SL}$. For consistency with the previous notation, the set of solutions obtained from MSOPS-II is referred to as $\mathcal{N}_{\mathcal{MS}}$.

Table 4.13 shares the application of NL-MVU-PCA on one of the available $\mathcal{N}_{\mathcal{MS}}$, up to RCM analysis. While the preliminary steps are self explanatory, the following features in the RCM analysis are worth noting:

- There are nine *potentially correlated* objective subsets. Four of these are constituted by all range-related objectives, that is, $\{f_1, f_3, f_5, f_7\}$. The remaining five are constituted by all the velocity-related objectives together with the transmission time-related objective, that is, $\{f_2, f_4, f_6, f_8, f_9\}$.
- $T_{cor} = 0.371$ being relatively low, is indicative of redundancy in the problem.
- When the nine *potentially correlated* objective subsets are subjected to the test of T_{cor}, the following *correlated* subsets emerge:
  - $\mathcal{S}_1 = \mathcal{S}_5 = \{f_1, f_3, f_5, f_7\}$, for which f_3 emerges as the representative.
  - $\mathcal{S}_3 = \{f_1, f_3, f_5\}$, for which f_3 emerges as the representative.
  - $\mathcal{S}_7 = \{f_1, f_5, f_7\}$, for which f_7 emerges as the representative.
  - $\mathcal{S}_2 = \mathcal{S}_4 = \mathcal{S}_6 = \mathcal{S}_8 = \mathcal{S}_9 = \{f_2, f_4, f_6, f_8, f_9\}$, for which f_4 emerges as the representative.

Hence, $\mathcal{F}_s = \{f_3, f_4, f_7\}$, while $\mathcal{F}_r = \{f_1, f_2, f_5, f_6, f_8, f_9\}$. The follow-up error analysis is presented in Table 4.14, from which the following can be inferred:

- The collective error corresponding to $\mathcal{F}_r$ can be given by

$$\mathcal{E}_0^r = \sum_{i:f_i \in \mathcal{F}_r} \mathcal{E}_{0,i} = 0.03816 \equiv 3.816\%.$$

- The sorted normalized error list, $\mathcal{SL}$:{1.10E-03, 1.66E-03, 6.55E-03, 6.60E-03, 9.76E-03, 1.90E-02, 3.91E-02, 1.60E-01, 7.55E-01}.
- The preference ranking of objectives corresponding to $\mathcal{SL}$ occurs to be $\{f_9 \lhd f_2 \lhd f_1 \lhd f_8 \lhd f_5 \lhd f_6 \lhd f_4 \lhd f_7 \lhd f_3\}$.

The preference structure of the objectives, *learnt* above (within the restriction of a single application of NL-MVU-PCA), paves the way for the decision support as follows.

4.7.2.1 Essential Objective Set

$\mathcal{F}_{\mathcal{S}} = \mathcal{F}_s = \{f_3, f_4, f_7\}$; $\mathcal{F}_{\mathcal{R}} = \mathcal{F}_r = \{f_1, f_2, f_5, f_6, f_8, f_9\}$; and the error associated with omission of $\mathcal{F}_{\mathcal{R}}$ happens to be $\mathcal{E}^R = \mathcal{E}_0^r = 0.03816 \equiv 3.816\%$.

Table 4.13 Radar waveform problem: an application of NL-MVU-PCA on $\mathcal{N}_{\mathcal{MS}}$ (taken from [16])

(a) Correlation matrix (R)

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
f_1	1.000	-0.872	0.863	-0.916	0.593	-0.412	0.522	-0.744	-0.925
f_2	-0.872	1.000	-0.939	0.954	-0.516	0.485	-0.365	0.865	0.953
f_3	0.863	-0.939	1.000	-0.942	0.514	-0.471	0.286	-0.855	-0.984
f_4	-0.916	0.954	-0.942	1.000	-0.560	0.469	-0.441	0.823	0.970
f_5	0.593	-0.516	0.514	-0.560	1.000	-0.197	0.487	-0.406	-0.567
f_6	-0.412	0.485	-0.471	0.469	-0.197	1.000	-0.121	0.558	0.465
f_7	0.522	-0.365	0.286	-0.441	0.487	-0.121	1.000	-0.150	-0.388
f_8	-0.744	0.865	-0.855	0.823	-0.406	0.558	-0.150	1.000	0.843
f_9	-0.925	0.953	-0.984	0.970	-0.567	0.465	-0.388	0.843	1.000

(b) Kernel matrix (K)

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
f_1	0.0638	-0.0452	0.1782	-0.0467	-0.0250	-0.0446	0.0114	-0.0456	-0.0460
f_2	-0.0452	0.0487	-0.1955	0.0501	0.0241	0.0480	-0.0284	0.0489	0.0494
f_3	0.1782	-0.1955	1.0000	-0.2021	-0.1160	-0.1927	-0.0750	-0.1977	-0.1992
f_4	-0.0467	0.0501	-0.2021	0.0518	0.0250	0.0495	-0.0288	0.0505	0.0509
f_5	-0.0250	0.0241	-0.1160	0.0250	0.0266	0.0238	-0.0074	0.0244	0.0246
f_6	-0.0446	0.0480	-0.1927	0.0495	0.0238	0.0475	-0.0283	0.0483	0.0488
f_7	0.0114	-0.0284	-0.0750	-0.0288	-0.0074	-0.0283	0.2128	-0.0278	-0.0284
f_8	-0.0456	0.0489	-0.1977	0.0505	0.0244	0.0483	-0.0278	0.0494	0.0498
f_9	-0.0460	0.0494	-0.1992	0.0509	0.0246	0.0488	-0.0284	0.0498	0.0504

(c) Eigen-decomposition, Eigenvalue Analysis and Selection Scores

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	
	0.809	0.163	0.019	0.009	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	
	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	sc_i
f_1	-0.170	0.109	-0.916	-0.093	-0.001	0.004	-0.000	-0.002	0.333	0.173
f_2	0.181	-0.162	0.081	-0.147	-0.110	-0.197	-0.754	0.424	0.333	0.176
f_3	-0.889	-0.157	0.270	-0.026	-0.000	0.003	-0.000	-0.002	0.333	0.750
f_4	0.187	-0.165	0.086	-0.148	-0.474	0.740	0.151	0.016	0.333	0.181
f_5	0.105	-0.048	0.000	0.935	-0.000	-0.000	-0.000	-0.000	0.333	0.101
f_6	0.179	-0.161	0.080	-0.147	-0.064	-0.452	0.635	0.436	0.333	0.174
f_7	0.036	0.910	0.231	-0.075	-0.002	0.000	-0.000	-0.000	0.333	0.182
f_8	0.183	-0.160	0.082	-0.147	0.848	0.270	0.022	-0.082	0.333	0.177
f_9	0.185	-0.163	0.083	-0.149	-0.193	-0.368	-0.053	-0.788	0.333	0.179

(d) RCM Analysis

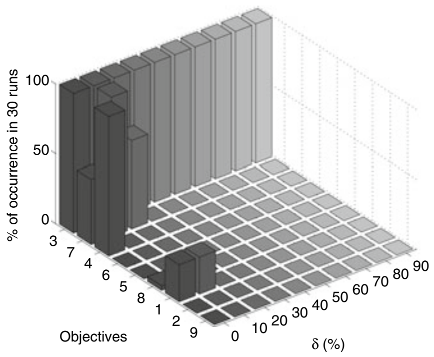
Potential identically correlated set	Eq 4.4(i)	$\hat{\mathcal{S}}_1 = \hat{\mathcal{S}}_3 = \hat{\mathcal{S}}_5 = \hat{\mathcal{S}}_7 = \{f_1, f_3, f_5, f_7\}$
		$\hat{\mathcal{S}}_2 = \hat{\mathcal{S}}_4 = \hat{\mathcal{S}}_6 = \hat{\mathcal{S}}_8 = \hat{\mathcal{S}}_9 = \{f_2, f_4, f_6, f_8, f_9\}$
Correlation threshold: T_{cor}	Eq 4.4(iii)	$1.0 - 0.808941 \times (1.0 - 2/9) = 0.370824$
Identically correlated set	Eq 4.4(ii)	$\mathcal{S}_1 = \mathcal{S}_5 = \{f_1, f_3, f_5, f_7\}$
		$\mathcal{S}_3 = \{f_1, f_3, f_5\}$; $\mathcal{S}_7 = \{f_1, f_5, f_7\}$
		$\mathcal{S}_2 = \mathcal{S}_4 = \mathcal{S}_6 = \mathcal{S}_8 = \mathcal{S}_9 = \{f_2, f_4, f_6, f_8, f_9\}$

Table 4.14 Radar waveform problem: error analysis for Table 4.13 (taken from [16])

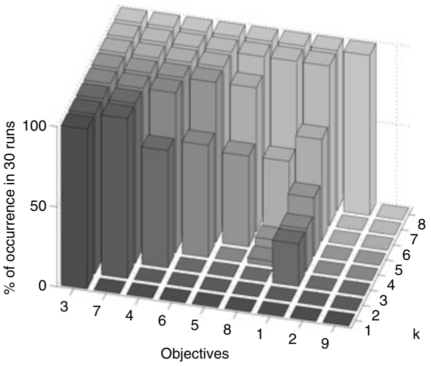
	Index $i \equiv f_i \in f_r$, and index $j \equiv f_j \in f_s$			
Category	c_i^M & c_j^M	$max\{\delta_{ij}.R_{ij}\}$	$\mathcal{E}_{0,i}$ & $\mathcal{E}_{0,j}$	$\eta_{0,i}$ & $\eta_{0,j}$
$f_1 \equiv f_i \in \mathcal{F}_r$	0.041116	0.863848 (R_{13})	0.005597	0.006558
$f_2 \equiv f_i \in \mathcal{F}_r$	0.031403	0.954665 (R_{24})	0.001423	0.001667
$f_3 \equiv f_j \in \mathcal{F}_s$	0.644745	–	0.644744	0.755382
$f_4 \equiv f_j \in \mathcal{F}_s$	0.033398	–	0.033397	0.039128
$f_5 \equiv f_i \in \mathcal{F}_r$	0.017168	0.514645 (R_{53})	0.008332	0.009762
$f_6 \equiv f_i \in \mathcal{F}_r$	0.030609	0.469935 (R_{64})	0.016224	0.019008
$f_7 \equiv f_j \in \mathcal{F}_s$	0.137228	–	0.137228	0.160776
$f_8 \equiv f_i \in \mathcal{F}_r$	0.031862	0.823009 (R_{84})	0.005639	0.006606
$f_9 \equiv f_i \in \mathcal{F}_r$	0.032472	0.970888 (R_{94})	0.000945	0.001107

4.7.2.2 δ-MOSS and k-EMOSS Analysis

By treating $\mathcal{SL}$ and the corresponding preference ranking of objectives, across 30 applications of NL-MVU-PCA (corresponding to 30 $\mathcal{N}_{\mathcal{MS}}$) as reference, the δ-MOSS analysis with varying δ, and k-EMOSS analysis with varying k are summarized in the Fig. 4.9a and b, respectively, in which the height of each bar indicates the percentage of runs (out of 30) in which a specific objective falls in the reduced objective set for the corresponding δ or k, as applicable. For instance, if the analyst is keen to know:



(a) δ-MOSS: % of occurrence of objectives (varying δ)



(b) k-EMOSS: % of occurrence of objectives (varying k)

Fig. 4.9 Radar waveform problem: visual representation of the δ-MOSS and k-EMOSS analysis by NL-MVU-PCA based on 30 available $\mathcal{N}_{\mathcal{MS}}$ (taken from [16])

- The smallest subset of objectives ensures that the error associated with the omission of the remaining objectives does not exceed $\delta = 0.1 \equiv 10\%$; then, $\mathcal{F}_{\{0.1\}} = \{f_1, f_3, f_4, f_7\}$. Similarly, for $\delta = 0.7 \equiv 70\%$; $\mathcal{F}_{\{0.7\}} = \{f_3\}$.
- The subset of 2 objectives, such that the error associated with omission of the remaining objectives is the minimum, then $\mathcal{F}_{\{k=2\}} = \{f_3, f_4\}$. Similarly, for the subset of 7 objectives, $\mathcal{F}_{\{k=7\}} = \{f_1, f_3, f_4, f_5, f_6, f_7, f_8\}$.

4.8 Summary

This chapter has presented an ML-based framework that operates on the objective vectors of the non-dominated solutions obtained from an EMâOA; *learns* the objectives' preference structure by preserving the correlation structure of the solutions; and provides decision support through revelation of (i) an essential objective set—a smallest set of m ($m \leq M$) conflicting objectives that generates the same PF as that of the original problem with M objectives, (ii) the smallest subset of objectives that ensures that the error associated with omission of the remaining objectives does not exceed a user-defined δ, and (iii) the subset of k objectives (k being user defined), such that the error associated with omission of the remaining objectives is minimum. Recognizing that the available solution sets may be *noisy* (not be representative of the PF), this framework has embedded *denoising* mechanisms, where the strength of correlation between an objective pair is tested against a dynamically computed T_{cor}. The efficacy of the framework is tested on a wide range of test problems, and its practical utility is demonstrated on two real-world problems. The significance of this chapter lies in the fact that, unlike other chapters where the definition of the problem is treated as a *given*, this chapter aims at knowledge discovery about the problem definition itself.

References

1. Beer, S.: Platform for change. John Wiley & Sons Inc, New York, NY, USA (1975)
2. Brockhoff, D., Zitzler, E.: Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. In: Runarsson, T.P., Beyer, H.G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) Parallel Problem Solving from Nature - PPSN IX, pp. 533–542. Springer, Berlin Heidelberg, Berlin, Heidelberg (2006)
3. Brockhoff, D., Zitzler, E.: Objective Reduction in Evolutionary Multiobjective Optimization: Theory and Applications. Evol. Comput. **17**(2), 135–166 (2009). https://doi.org/10.1162/evco.2009.17.2.135
4. Cohen, J.: Statistical power analysis for the behavioral sciences, 2nd edn. Erlbaum, Hillsdale, NJ (1988)
5. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems **9**(2), 115–148 (1995)
6. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Computer Science and Informatics **26**(4), 30–45 (1996)

7. Deb, K., Kumar, A.: Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, p. 781–788. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1276958.1277116
8. Deb, K., Kumar, A.: Light beam search based multi-objective optimization using evolutionary algorithms. In: 2007 IEEE Congress on Evolutionary Computation, pp. 2125–2132 (2007). https://doi.org/10.1109/CEC.2007.4424735
9. Deb, K., Mohan, M., Mishra, S.: Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. Evol. Comput. **13**(4), 501–525 (2005). https://doi.org/10.1162/106365605774666895
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
11. Deb, K., Saxena, D.K.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: IEEE Congress on Evolutionary Computation, pp. 3353–3360 (2006)
12. Deb, K., Sinha, A., Korhonen, P.J., Wallenius, J.: An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. IEEE Trans. Evol. Comput. **14**(5), 723–739 (2010). https://doi.org/10.1109/TEVC.2010.2064323
13. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. International Journal of Computational Intelligence Research (IJCIR) **2**(6), 273–286 (2006)
14. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, pp. 105–145. Springer London, London (2005). https://doi.org/10.1007/1-84628-137-7_6
15. Ding, R., Dong, H.B., Yin, G.S., Sun, J., Yu, X.D., Feng, X.B.: An objective reduction method based on advanced clustering for many-objective optimization problems and its human-computer interaction visualization of Pareto front. Computers & Electrical Engineering **93**, 107,266 (2021). https://doi.org/10.1016/j.compeleceng.2021.107266
16. Duro, J.A., Saxena, D.K., Deb, K., Zhang, Q.: Machine learning based decision support for many-objective optimization problems. Neurocomputing **146**, 30–47 (2014). https://doi.org/10.1016/j.neucom.2014.06.076
17. Gupta, R., Nanda, S.J.: Objective reduction in many-objective optimization with social spider algorithm for cloud detection in satellite images. Soft. Comput. **26**, 2935–2958 (2022). https://doi.org/10.1007/s00500-021-06655-8
18. Heisenberg, W.: Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. Z. Phys. **43**, 172–198 (1927)
19. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
20. Hughes, E.: Many-objective radar design software. Online (2007). Available: http://code.evanhughes.org
21. Hughes, E.J.: MSOPS-II: A general-purpose many-objective optimiser. In: IEEE Congress on Evolutionary Computation, pp. 3944–3951. IEEE Press (2007)
22. Hughes, E.J.: Radar waveform optimization as a many-objective application benchmark. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science, vol. 4403, pp. 700–714. Springer, Berlin / Heidelberg (2007)
23. Jaimes, A.L., Coello, C.A.C., Chakraborty, D.: Objective reduction using a feature selection technique. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 673–680 (2008)
24. J.F.Sturm: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software **11**(1), 625–653 (1999)

25. Li, K., Lai, G., Yao, X.: Interactive evolutionary multiobjective optimization via learning to rank. IEEE Trans. Evol. Comput. **27**(4), 749–763 (2023). https://doi.org/10.1109/TEVC.2023.3234269
26. Luo, N., Li, X., Lin, Q.: Objective reduction for many-objective optimization problems using objective subspace extraction. Soft. Comput. **22**, 1159–1173 (2018). https://doi.org/10.1007/s00500-017-2498-6
27. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81–97 (1956)
28. Musselman, K., Talavage, J.: A tradeoff cut approach to multiple objective optimization. Oper. Res. **28**(6), 1424–1435 (1980). https://doi.org/10.1287/opre.28.6.1424
29. Nguyen, X.H., Bui, T.L., Tran, C.T.: An improvement of clustering-based objective reduction method for many-objective optimization problems. Journal of Science and Technique **8** (2019). https://doi.org/10.56651/lqdtu.jst.v8.n02.65.ict
30. Nisbett, R.E., Wilson, T.D.: Telling more than we can know: Verbal reports on mental processes. Psychol. Rev. **84**(3), 231–259 (1977)
31. Purshouse, R.C., Fleming, P.J.: Evolutionary Many-Objective Optimization: An Exploratory Analysis. In: IEEE Congress on Evolutionary Computation, pp. 2066–2073 (2003)
32. Saul, L.K., Weinberger, K.Q., Ham, J.H., Sha, F., Lee, D.D.: Spectral methods for dimensionality reduction. In: Schoelkopf, O.C.B., Zien, A. (eds.) Semisupervised Learning. MIT Press, Cambridge, MA (2006)
33. Saxena, D., Deb, K.: Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science, vol. 4403, pp. 772–787. Springer, Berlin / Heidelberg (2007)
34. Saxena, D.K., Duro, J.A., Tiwari, A., Deb, K., Zhang, Q.: Objective reduction in many-objective optimization: Linear and nonlinear algorithms. IEEE Trans. Evol. Comput. **77**(1), 77–99 (2013)
35. Scholkopf, B., Smola, A., Muller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
36. Shlens, J.: A tutorial on principal component analysis. Tech. rep., Center for Neural Science, New York University, available at:http://www.snl.salk.edu/∼shlens/pca.pdf (accessed: May 2011) (2009)
37. Singh, H.K., Isaacs, A., Ray, T.: A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. IEEE Trans. Evol. Comput. **15**(4), 539–556 (2011)
38. Sinha, A., Deb, K., Korhonen, P., Wallenius, J.: Progressively interactive evolutionary multi-objective optimization method using generalized polynomial value functions. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE Press (2010)
39. Slovic, P., Lichtenstein, S.: Comparison of bayesian and regression approaches to the study of information processing in judgment. Organ. Behav. Hum. Perform. **6**(6), 649–744 (1971). https://doi.org/10.1016/0030-5073(71)90033-X
40. Thiele, L., Miettinen, K., Korhonen, P.J., Molina, J.: A preference-based evolutionary algorithm for multi-objective optimization. Evol. Comput. **17**(3), 411–436 (2009). https://doi.org/10.1162/evco.2009.17.3.411
41. Wang, H., Yao, X.: Objective reduction based on nonlinear correlation information entropy. Soft. Comput. **20**, 2393–2407 (2016). https://doi.org/10.1007/s00500-015-1648-y
42. Weinberger, K.Q., Saul, L.K.: Unsupervised learning of image manifolds by semidefinite programming. Int. J. Comput. Vision **70**(1), 77–90 (2006). https://doi.org/10.1007/s11263-005-4939-z
43. Yu, P.L.: Habitual domains. Oper. Res. **39**(6), 869–876 (1991)
44. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007). https://doi.org/10.1109/TEVC.2007.892759

Chapter 5
Learning to Converge Better: IP2 Operator

In the context of online innovization (Sect. 3.1.2, Chap. 3), it has been discussed that inter-variable relationships with pre-specified structures can be extracted in any intermediate generation of an evolutionary multi- and many-objective optimization algorithm (EMâOA) run. Subsequently, these relationships can be used for offspring *repair*, within the same EMâOA run, to help induce better convergence [7, 8]. Any attempt to eliminate the a priori specification of the relationship structure would require alternative criteria that could guide the improvement in convergence. It is suggested that if the scope of online innovization could be narrowed down to reference vector (RV)-based EMâOAs, RV-EMâOAs, then the underlying RVs could provide the aspired criteria. The rationale for this claim is rooted in the recognition that

1. The solutions associated with the RVs in the objective space ($\mathcal{Z}$ space) have their representations in the variable space ($\mathcal{X}$ space).
2. In any generation of an RV-EMâOA, the current best solution associated with an RV can be known in the $\mathcal{Z}$ space. Some previous generation solutions associated with the RV can be mapped onto the current best solution. Corresponding to this mapping in the $\mathcal{Z}$ space, a mapping in the $\mathcal{X}$ space can be obtained.
3. Since the mapping defined above in the $\mathcal{Z}$ space captures the improvement in convergence property of the previous solutions, the underlying mapping in the $\mathcal{X}$ space promises to capture efficient *search* directions for better convergence.

In this context, this chapter presents the *Innovized Progress 2 (IP2)* operator, designed for convergence enhancement in RV-EMâOAs.

The IP2 operator includes an ML-based approach that (a) *maps* the solutions from previous generations of an RV-EMâOA run to the best selected solution up to the current generation in $\mathcal{Z}$ space, *along* the respective RVs; (b) trains an ML model to *learn* the corresponding directional improvements in the $\mathcal{X}$ space; and (c) uses the learned ML model in the same generation to create 50% pro-convergence offspring through *progression* of 50% of the natural[1] offspring (in the $\mathcal{X}$ space). The choice of creating only 50% pro-convergence offspring in a particular generation has been justified later in Sect. 5.3, in the context of convergence–diversity balance and ML-driven risk–reward trade-off.

The remainder of this chapter is organized as follows: Sect. 5.1 describes the IP2 operator, followed by an outline of its integration with NSGA-III, an RV-EMâOA, in Sect. 5.2. Section 5.4 discusses the computational complexity of the IP2 operator, followed by a discussion on the experimental setup to demonstrate the efficacy of the IP2 operator in Sect. 5.5. Finally, the results and related discussions are presented in Sect. 5.6.

5.1 IP2 Operator for Convergence Enhancement

The IP2 operator whose scope is summarized above encapsulates three modules, including *Training-dataset construction*, *ML Training*, and *Offspring Progression*. The design and implementation of these modules is discussed below.

5.1.1 Training-Dataset Construction Module

In any generation t of the RV-EMâOA, the *training-dataset* is constituted by *mapping* between members of *input archive* A_t and members of *target archive* T_t. The process of constituting and updating A_t and T_t, and *mapping* of their members is presented below. Toward its prerequisite terminology, let P_t (of size N) be the parent population; Q_t (of size N) be the offspring population; $\mathcal{R}$ (of size N) be the RV set; and t_{past} be a user-defined parameter that represents the number of past generations involved in the composition of A_t.

5.1.1.1 Input Archive Composition and Update

In any generation t, *input archive* A_t is intended to serve as a pool of reasonably diverse distinct solutions from previous generations, which can be mapped onto representative solutions in the current generation (*target archive*), so that (a) an ML method could *learn* the directional improvements in the $\mathcal{X}$ space, and (b) such a *learning* could be utilized for effective progression of some of the current offspring. In this spirit, $A_t = \{P_{t-t_{\text{past}}}\} \cup \{Q_{t-t_{\text{past}}}, Q_{t-t_{\text{past}}+1}, \dots Q_{t-1}\}$. Notably

[1] Natural offspring refers to the offspring initially created using the natural variation operators, including recombination and mutation.

1. First, both parents and offspring in the $(t - t_{\text{past}})$th generation are included to account for maximum diversity without incorporating duplicate solutions (since parents and offspring in any generation are distinct).
2. In addition, only the offspring from the $(t - t_{\text{past}} + 1)$th until the $(t - 1)$th generation are included, while corresponding parents are excluded. This is done to avoid duplicity of (parent) solutions, since, in any particular generation, not all parents may be replaced by the offspring. This duplicity would be more prevalent in the later generations of an RV-EMâOA-IP2 run (RV-EMâOA integrated with IP2 operator), compared to the earlier generations.

Given its composition, A_t automatically updates with each increase in the generation counter t.

5.1.1.2 Target Archive Composition and Update

As indicated above, *target archive* in the current generation t, namely T_t is intended to serve as a set of representative solutions, onto which the A_t members could be mapped, providing a basis for ML training and its subsequent use. To this end, T_t seeks to identify N solutions along the N RVs. From implementation perspective, T_t is obtained by updating T_{t-1} by accounting for the members in P_t. This poses two pertinent questions, as to how (a) T_1 is initialized, and (b) T_2 is obtained by updating T_1 through the members in P_2, reflecting in general as to how T_t can be obtained by updating T_{t-1} through the members in P_t.

T_1 is initialized by *associating with each RV* one member of P_1 as the target. Different RV-EMâOAs can have different *association* criteria. For example, NSGA-III [5] uses minimum perpendicular distance (PD) of a solution from the RV, as the association criterion, while MOEA/D [15] relies on the minimum value of the scalarization function, such as achievement scalarization function (ASF) or penalty boundary intersection (PBI). These criteria require normalized objective values, which can be obtained as given by Eq. 5.1. Here, Z^{ideal} and Z^{nadir} represent the ideal and nadir points, respectively:

$$\bar{f}_m(X) = \frac{f_m(X) - Z_m^{\text{ideal}}}{Z_m^{\text{nadir}} - Z_m^{\text{ideal}}}, \quad \forall m \in \{1, 2, \ldots, M\}. \tag{5.1}$$

Any subsequent generation t $(t \geq 2)$ entails two steps: (a) *associating each member of* P_t with some RV, and (b) updating T_{t-1}. It may be noted that the association in $t = 1$ is in stark contrast to the association in $t \geq 2$. While the former ensured one associated solution per RV (each RV being the reference for association), the latter did not (each solution being the reference for association). Once the P_t members are associated, there may exist some RVs with just one associated target, while others may have two or more targets. When two or more target solutions exist, one of them is selected based on the RV-EMâOA's underlying association criterion (PD, ASF, PBI, etc., as the case may be). In this background, the process of updating T_t is formalized in Algorithm 5.1. Here, the objective values of the solutions in T_{t-1} and P_t are normalized (lines 1–2), and an array V, sized $N \times N$, is initialized as an empty array (line 3), so it can later store the association criterion values, referred to as the metric values. Subsequently, for *each solution* (indexed i) in P_t: first, the metric value is computed with respect to each RV, $\mathcal{R}_{(j)} \in \mathcal{R}$ (lines 5–7); the minimum metric value is stored in V^P, and the RV (indexed J) offering this value is noted (lines 8–9). Let V^T be the metric value of the existing target associated with $\mathcal{R}_{(J)}$. If V^P is better than V^T, then the considered (ith) parent solution becomes the target for $\mathcal{R}_{(J)}$ (lines 10–12).

Algorithm 5.1: `Update_Target_Archive` (P_t, T_{t-1}, $\mathcal{R}$, Z^{ideal}, Z^{nadir})

```
Input: Parent population $P_t$, last target archive $T_{t-1}$, set of RVs $\mathcal{R}$, ideal point $Z^{\text{ideal}}$, nadir
       point $Z^{\text{nadir}}$
Output: Updated target archive $T_t$
1  $\{F^T,\ F^P\} \leftarrow$ Objective function values in $\{T_{t-1},\ P_t\}$
2  $\{\bar{F}^T,\ \bar{F}^P\} \leftarrow$ Normalize the objective values using $Z^{\text{ideal}}$ and $Z^{\text{nadir}}$
3  $V_{[N\times N]} \leftarrow \emptyset$          % stores evaluated metric values (PD/ASF/PBI)
4  $T_t \leftarrow T_{t-1}$
5  for $i$ = 1 to $N$ do
6      for $j$ = 1 to $N$ do
7          $V_{i,j} \leftarrow$ Metric value of $\bar{F}^P_{(i)}$ w.r.t. $\mathcal{R}_{(j)}$
8      $V^P \leftarrow$ Store the minimum value of $V_{i,j}$, where $j \in [1, N]$
9      $J \leftarrow$ Calculate the index $j$, where $V_{i,j}$ is same as $V^P$
10     $V^T \leftarrow$ Metric value of $\bar{F}^T_{(J)}$ with respect to $\mathcal{R}_{(J)}$
11     if $V^P < V^T$ then
12         $T_{t,(J)} \leftarrow P_{t,(i)}$
```

For the benefit of the readers, the procedure for obtaining T_t ($t \geq 2$) by updating T_{t-1} through the members in P_t is illustrated through an example, in Fig. 5.1. Here, for each RV in $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_6\}$, there exists (a) a corresponding target from $T_{t-1} = \{m_1,\ m_2,\ \ldots,\ m_6\}$, and (b) no, unique, or multiple potential targets from members in $P_t = \{s_1, s_2, \ldots, s_6\}$. Assuming NSGA-III to be the base RV-EMâOA, preference among competing targets is given to non-dominated ones. However, if there are two or more non-dominated targets, then the one with the lowermost PD becomes the representative target.

1. For $\mathcal{R}_1, \mathcal{R}_3$, and $\mathcal{R}_6$, the associated targets from T_{t-1} are m_1, m_3, and m_6, respectively, while the corresponding targets from P_t are s_1, s_3, and s_6, respectively. As evident in the figure, for each of these RVs, the target member from P_t dominates the target member from T_{t-1}. Therefore, s_1, s_3 and s_6 become the targets of $\mathcal{R}_1$, $\mathcal{R}_3$, and $\mathcal{R}_6$, respectively, in T_t.
2. For $\mathcal{R}_2$, the associated target solution from T_{t-1} is m_2; however, there are two potential target solutions from P_t, namely s_2 and s_3. Here, s_2 emerges as the updated target solution, since it dominates both s_3 and m_2.

3. For $\mathcal{R}_4$, the associated target solution is s_5 since it is non-dominated to m_4 but offers a lower value of PD.
4. For $\mathcal{R}_5$, the associated target solution from T_{t-1} is m_5; however, there is no potential target solution from P_t. Even though s_6 dominates m_5, s_6 is not considered since it is associated with another RV $\mathcal{R}_6$. Hence, m_5 is retained as the updated target solution.

Finally, the archive T_t is constituted by $\{s_1, s_2, s_4, s_5, m_5, s_6\}$.

5.1.1.3 Mapping of Input and Target Archives

This is the last step of the *training-dataset construction* module, where the solutions in A_t are mapped onto those in T_t, to yield *training-dataset* D_t. Each solution in A_t is associated with some RV (as discussed above), and with each RV a solution from T_t is associated (as discussed above). Therefore, the RV involved provides a basis for the association of each solution in A_t with a particular solution in T_t. For example, if a solution $a_i \in A_t$ is associated with a particular RV, say $\mathcal{R}_j$, then there is also a solution, say $m_k \in T_t$, that is associated with $\mathcal{R}_j$. Hence, the solution a_i gets mapped onto m_k. Since the goal of the IP2 operator is to learn the directional improvements in the $\mathcal{X}$ space, only the variable vectors (X vectors) of these solutions are stored in D_t. In the context of the above example, the X vectors of a_i and m_k together form a $input$ -$target$ sample for D_t. The dataset construction based on the mapping of the input and target archives is summarized in Algorithm 5.2.

Before moving on to how the constructed dataset can be utilized, the reader may note the subtle considerations of *convergence-diversity trade-off* embedded in the dataset, by choice. In Fig. 5.1, m_5 is dominated by another target member (s_6). Hence, from the convergence perspective, it should ideally not be treated as one of the target members. However, being the best and sole representative for $\mathcal{R}_5$ till the tth

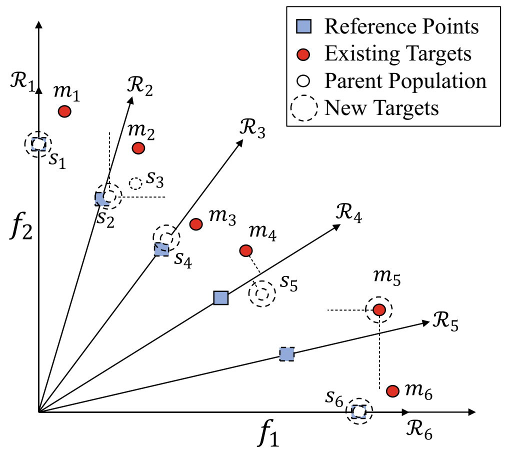


Fig. 5.1 Depicting the procedure for obtaining the current target archive (T_t) by updating the previous target archive (T_{t-1}) through the current population (P_t). NSGA-III is assumed to be the base RV-EMâOA, and hence, non-domination check and perpendicular distance constitute the criteria for association of a solution with a RV

Algorithm 5.2: `Archive_Mapping` $(A_t, T_t, \mathcal{R}, Z^{\text{ideal}}, Z^{\text{nadir}})$

```
Input: Input archive $A_t$, target archive $T_t$, set of RVs $\mathcal{R}$, ideal point $Z^{\text{ideal}}$, and nadir point $Z^{\text{nadir}}$
Output: training-dataset $D_t$
1  $F^A \leftarrow$ Objective function values in $A_t$
2  $\bar{F}^A \leftarrow$ Normalize the objective values using $Z^{\text{ideal}}$ and $Z^{\text{nadir}}$
3  $D_t \leftarrow \emptyset$
4  $V_{[1\times N]} \leftarrow \emptyset$                  % stores evaluated metric values
5  for $i = 1$ to $sizeof(A_t)$ do
6      for $j = 1$ to $N$ do
7          $V_j \leftarrow$ Metric value of $\bar{F}^A_{(i)}$ with respect to $\mathcal{R}_{(j)}$
8      $I \leftarrow$ Index value where $V_j$ is minimum
9      $input \leftarrow X$ vector of $A_{t,(i)}$; $target \leftarrow X$ vector of $T_{t,(I)}$
10     $D_t \leftarrow$ Add $[input, target]$ to $D_t$
```

generation, it may deserve consideration from the diversity perspective. Broadly, the following options are open to the handling of $\mathcal{R}_5$:

1. Both $\mathcal{R}_5$ and m_5 are left out of consideration: in this case, a significant part of the $\mathcal{X}$ space would remain unrepresented in D_t, and therefore may not be a good choice.
2. $\mathcal{R}_5$ is considered but without m_5 as its target solution: in this case, a non-dominated solution from a neighboring RV can be assigned as the target for $\mathcal{R}_5$, say s_6. However, this may imply preference for convergence over diversity.
3. $\mathcal{R}_5$ is considered with m_5 as its target solution: this choice would imply preference to diversity even at the cost of poorer convergence.

In the approach presented here, a judicious choice in favor of the third option has been made, guided by the rationale that (a) diversity preservation is crucial, especially in the early RV-EMâOA generations, and (b) the marginal compromise in convergence could possibly be overcome in subsequent generations as long as all regions of the $\mathcal{X}$ space are explored equitably.

5.1.2 ML Training Module

The goal here is to train an ML model toward capturing the directional improvement in the $\mathcal{X}$ space, which defines the transition of $input$ solutions to their respective $target$ solutions, along all the RVs. For this task, random forest (RF) [2] is chosen as the ML method here, though other multi-output regression methods could also be used, including artificial neural network (ANN), gradient boost, XGBoost, least angle regression, and support vector regression.

The ML training module is executed through (a) a pre-training step involving normalization of the training-dataset using the *dynamic normalization* method, and

(b) ML training itself, as presented in Algorithm 5.3. Notably, each time the IP2 operator is invoked, a new ML model is trained and the last trained ML model is discarded.

Algorithm 5.3: `Training` $(D_t, [X^{(L)}, X^{(U)}])$

```
Input: training-dataset $D_t$, lower & upper bounds of variables specified in the problem, $X^{(L)}$
       and $X^{(U)}$
Output: Trained ML model $ML$, Bounds $[X^{\min}, X^{\max}]$
1 $\{X^{(L,t)},\ X^{(U,t)}\} \leftarrow$ Minimum and Maximum of each variable in $D_t$
2 $X^{\min},\ X^{\max} \leftarrow$ Compute the dynamic normalization bounds using Eq. 5.2
3 Normalize $D_t$ using $X^{\min}$ and $X^{\max}$ as bounds
4 $ML \leftarrow$ Trained ML model using $D_t$
```

5.1.2.1 Dynamic Normalization of the Training-Dataset

Training of an ML model (based on the training-dataset D_t) involves minimizing a loss/error function, such as *mean squared error* (MSE). Also, it is known that the scales and ranges of different variables, depending on the problem at hand, may be different. Hence, to ensure a fair training process, it is critical to *even out* each variable's contribution to the loss function, necessitating their normalization.

In the approach presented here, the normalized value of any variable x_k, denoted by $\bar{x}_k$, is given by Eq. 5.2:

$$\bar{x}_k = \frac{x_k - x_k^{\min}}{x_k^{\max} - x_k^{\min}}, \tag{5.2}$$

where

$$x_k^{\min} = 0.5\left(x_k^{(L)} + x_k^{(L,t)}\right), \text{ and } x_k^{\max} = 0.5\left(x_k^{(U)} + x_k^{(U,t)}\right).$$

It is important to note that the variable bounds, $x_k^{\min}$ and $x_k^{\max}$, used above are *dynamic* in nature. In that, $x_k^{(L)}$ and $x_k^{(U)}$ denoting the static permitted bounds for x_k are problem-specific features. However, $x_k^{(L,t)}$ and $x_k^{(U,t)}$ denoting the extent to which the permitted bounds are explored until the generation t of RV-EMâOA are dynamic in nature, and implicitly factor in the RV-EMâOA's performance, thus far.

5.1.2.2 Training an RF Model

Once the normalization is executed, as discussed above, an RF model is trained on the normalized training-dataset. This requires three critical settings: the number of trees (N_{tr}); the number of variables/features considered while splitting a node (N_{feat});

and the loss function. Assuming that the training-dataset D_t is constituted by N_{sam} training samples, the parameters are fixed as $N_{\text{tr}} = N_{\text{sam}}$ and $N_{\text{feat}} = n$. The loss function used is MSE, and the rest of the RF settings are kept as default.[2]

5.1.3 *Offspring Progression Module*

Here, the trained ML model is directly used for progression of some of the natural offspring solutions, without their prior *evaluation*. This module is executed as a four-step process, where each step alters the X vector of the offspring. After progression, the pro-convergence offspring solutions are referred to as *progressed offspring solutions* (X^{pg}). The constitutive steps of offspring progression are described below.

5.1.3.1 Selection and Progression

Once the natural offspring solutions (Q_t) are created, 50% of them are randomly selected ($\lfloor 0.5N \rfloor$ offspring) to avoid their evaluation a priori; and then *trained* ML model is used for progression of the selected offspring. Since the ML model is trained on the normalized dataset, the X vectors of the randomly chosen natural offspring solutions are also normalized prior to their progression using the ML model, and their denormalization is done after progression.

5.1.3.2 Near-Boundary Restoration

Consider a problem where some of the optimum solutions are characterized by the extreme or boundary values for a particular variable x_k, implying either $x_k^* = x_k^{(L)}$ or $x_k^* = x_k^{(U)}$. It would be desirable for x_k to reach its respective extreme at the time of progression. However, during the initial or even intermediate generations of the underlying RV-EMâOA, the training-dataset may predominantly contain x_k values away from its extremes. Therefore, the trained ML model may be biased against the extreme values of x_k. Consequently, if a natural offspring that has a near-extreme value for x_k is subjected to progression using the ML model, it is likely that x_k is driven in the opposite direction to what is desired. In such a case, the progression of x_k could be self-defeating. To help avoid such instances, it is important that variables having values very close to their extremes are barred from progression. From an implementation perspective, where the entire X vector of the selected offspring is subjected to the trained ML model, without any direct control over individual variables in the vector (some $x_k \in X$), it is imperative that progression be undone for those variables which, prior to progression, had values very close to

[2] The RF regressor used in this study has been taken from Scikit-learn's Python implementation.

their extremes. To this effect, the original values are restored for all such variables that, prior to progression, had values within 1% of either of their extremes.

5.1.3.3 Jutting the Progressed Offspring

Understandably, the ML model *learns* the directional improvements that define the transition of solutions from previous generations to the best solution until the current generation, along each RV. The current offspring's progression using such an ML model is based on the assumption that the directions found pertinent in the past shall again help the offspring transition to better solutions. In this situation, the IP2 operator treats the *learnt* directions for different RVs as promising search directions, and introduces the notion of step length through the parameter η, leading to jutted offspring solutions, given by

$$X^{\mathrm{jpg}} = X + \eta \times (X^{\mathrm{pg}} - X), \tag{5.3}$$

where X and X^{pg} mark the original and progressed offspring, respectively, and X^{jpg} represents the consequent jutted offspring. Notably, $\eta = 1$ leads to the originally progressed offspring ($X^{\mathrm{jpg}} = X^{\mathrm{pg}}$), while $\eta > 1$ leads to a different offspring ($X^{\mathrm{jpg}} \neq X^{\mathrm{pg}}$). Fundamentally, jutting counters the limitation that many ML-based regression methods, including RF, are not suitable for extrapolation, despite their excellence in predicting the data that can be interpolated from the input training-dataset. In effect, the challenge that the IP2 operator may not help create offspring solutions in regions that can only be achieved by extrapolating on the current population is greatly alleviated by jutting.

5.1.3.4 Boundary Repair

In the EMâO domain, there are several common methods to prevent an operation from setting the value of a variable outside its original bound. These include (a) replacing by the variable value at its respective bound, (b) choosing an arbitrary value within the variable bound, or (c) mapping the value inward, proportionally as much as it was outside the bound (*reflection*). While the first method may deteriorate population diversity in the $\mathcal{X}$ space since several offspring may end up with the same variable values (their respective bound), the second method may deteriorate search efficiency by introducing random variable values into several offspring. Based on the above, the IP2 operator incorporates a sophisticated variant of the third method [12]. In that, any variable $x_k \in X^{\mathrm{jpg}}$ that goes outside its permitted bounds ($[x_k^{(L)}, x_k^{(U)}]$) is mapped to an inner value based on an Inverse Parabolic Spread Distribution [12].

The overall process of the offspring's progression, as described through the four steps above, is summarized in Algorithm 5.4.

Algorithm 5.4: `Progression` (Q_t, η, [$X^{\min}$, $X^{\max}$], [$X^{(L)}$, $X^{(U)}$], ML)

```
Input: Original offspring $Q_t$, jutting parameter $\eta$, bounds from Algorithm 5.3 [$X^{\min}$, $X^{\max}$],
       variable bounds in problem definition [$X^{(L)}$, $X^{(U)}$]
Output: Progressed offspring $Q_t$
1  $I \leftarrow$ Randomly selected 50% offspring from $Q_t$
2  for $X \in I$ do
3      $\bar{X} \leftarrow$ Normalize $X$ using $X^{\min}$ and $X^{\max}$
4      $\bar{X}^{\text{pg}} \leftarrow ML(\bar{X})$
5      $X^{\text{pg}} \leftarrow$ Denormalize $\bar{X}^{\text{pg}}$ using $X^{\min}$ and $X^{\max}$
6      for each variable $k \in [1, n]$ do
7          Restore $x_k^{\text{pg}}$ to $x_k$ if $x_k$ lies in 1% vicinity of its bounds
8      $X^{\text{jpg}} \leftarrow$ Compute the jutted offspring using Eq. 5.3
9      Perform the variable boundary repair on $X^{\text{jpg}}$
10     Replace the original offspring $X$ in $Q_t$ by $X^{\text{jpg}}$
```

5.2 Integration of IP2 Operator into NSGA-III

This section describes the integration of the IP2 operator with NSGA-III, leading to NSGA-III-IP2. This integration, summarized in Algorithm 5.5, is generic in nature and can be extended to any other RV-EMâOA.

Notably, Algorithm 5.5 represents any intermediate generation t of NSGA-III-IP2, and involves a new parameter, namely $t_{\text{freq}}^{\text{IP2}}$, that specifies the number of generations between two successive progressions. First, *target archive* T_t is updated using Algorithm 5.1 (line 1, Algorithm 5.5). Then, the prerequisite condition for the first invocation of IP2 operator—i.e., population being completely non-dominated—is checked. If complete non-domination is detected, the $startIP2$ flag is marked as $True$ (lines 2–3, Algorithm 5.5). The non-domination check is crucial toward ensuring that before the IP2 operator is invoked for the first time, a reasonable diversity in the $\mathcal{Z}$ space has been achieved, to effect better learning. Subsequently

- If $t_{\text{freq}}^{\text{IP2}}$ generations have passed after IP2's last invocation, then IP2 is invoked, leading to creation of 100% natural offspring and progression of randomly chosen 50% offspring. Overall, Q_t remains sized N (lines 4–8, Algorithm 5.5).
- Otherwise, if IP2 is not invoked, all natural offspring (100%) are created and no progression is executed (lines 9–10, Algorithm 5.5).
- The offspring Q_t are evaluated and the input archive is updated (lines 11–12, Algorithm 5.5).
- The survival selection procedure of NSGA-III is executed (line 13, Algorithm 5.5).
- The count of offspring N_t^{surv} that survived to the next generation is estimated. In a generation where IP2 is invoked, if this count has improved compared to the previous generation, implying good performance of the IP2 operator, then $t_{\text{freq}}^{\text{IP2}}$ is reduced by 1, resulting in a more frequent progression. Otherwise, if the count has reduced, $t_{\text{freq}}^{\text{IP2}}$ is increased by 1 (lines 14–17, Algorithm 5.5).

Algorithm 5.5: Generation t of NSGA-III-IP2

```
Input: Ideal point $Z^{\text{ideal}}$, Nadir point $Z^{\text{nadir}}$, Reference vector set $\mathcal{R}$, original variable
       bounds $[x^{(L)}, x^{(U)}]$, Parent population $P_t$, Target Archive $T_{t-1}$, Input Archive $A_t$,
       number of past generations $t_{\text{past}}$ used in $A_t$, frequency of progression $t_{\text{freq}}^{\text{IP2}}$, number of
       survived offspring in $(t-1)$th generation $N_{t-1}^{\text{surv}}$
Output: $P_{t+1}$, $T_t$, $t_{\text{freq}}^{\text{IP2}}$, $A_{t+1}$, $N_t^{\text{surv}}$
1  $T_t \leftarrow$ Update the target archive using Algorithm 5.1
2  if population is completely non-dominated then
3      $startIP2 \leftarrow$ True
4  if $startIP2$ & $t_{\text{freq}}^{\text{IP2}}$ generations passed after last invocation then
5      $D_t \leftarrow$ Create the training-dataset using Algorithm 5.2
6      $ML \leftarrow$ Train the ML model using Algorithm 5.3
7      $Q_t \leftarrow$ Create 100% offspring using natural variation operators
8      $Q_t \leftarrow$ Progression of randomly picked 50% offspring using Algorithm 5.4
9  else
10     $Q_t \leftarrow$ Create 100% offspring using natural variation operators
11 Evaluate $Q_t$
12 $A_{t+1} \leftarrow$ Update the input archive for IP2 operator
13 $P_{t+1} \leftarrow$ Perform survival selection on $P_t \cup Q_t$
14 $N_t^{\text{surv}} \leftarrow$ Count of offspring in $Q_t$ that survived to $P_{t+1}$
15 if IP2 was invoked in current generation then
16     if $N_t^{\text{surv}} > N_{t-1}^{\text{surv}}$ then reduce $t_{\text{freq}}^{\text{IP2}}$ by 1
17     if $N_t^{\text{surv}} < N_{t-1}^{\text{surv}}$ then increase $t_{\text{freq}}^{\text{IP2}}$ by 1
```

5.3 Degree of IP2 Operator Utilization: Critical Considerations

The efficacy of any RV-EMâOA-IP2 vis-à-vis the base RV-EMâOA is bound to be influenced by the following decisions: (a) the frequency with which the IP2 operator is invoked (value of $t_{\text{freq}}^{\text{IP2}}$), and (b) the percentage of total offspring created using the IP2 operator, in any generation where it is invoked. In principle, such decisions should address the dual considerations of

- *Convergence–diversity balance:* conventionally, the natural variation operators (say, recombination and mutation) utilize the principle of *guided randomness* to produce the offspring solutions Q^{V}, without any explicit consideration of their convergence or diversity characteristics. Such convergence–diversity-neutral offspring, along with parent solutions, serve as input to the *selection* operator, which pursues EMâO's dual goals of convergence and diversity. Hence, it is imperative that, in any generation, the offspring resulting through the IP2 operator are not overskewed in favor of either convergence or diversity.
- *Risk–reward trade-off:* notably, the IP2 operator relies on *learning* the efficient search directions (in $\mathcal{X}$ space), based on the mapping of *inter-generational* solutions in $\mathcal{Z}$ space. Clearly, there is a possibility that such a *learning* may not be

Source of offspring solutions that are subjected to selection	Linkage of offspring solutions with the dual goals in EMâOAs	
	Convergence	Diversity
RV-EMâOA	Q^{V}	
RV-EMâOA-IP2	Q^{IP2}	Q^{V}

Fig. 5.2 Symbolic depiction of the degree of IP2 operator's contribution to convergence and diversity, over an entire run of RV-EMâOA-IP2. The natural offspring Q^{V} do not impose any explicit preference for convergence or diversity, hence, are marked by a different color

meaningful due to multiple factors, including the nonlinearity in the given problem; training-dataset comprising intermediate generation solutions which may not be representative of the Pareto front (PF); and choice of ML methods. Hence, it is imperative that the degree of reliance on the IP2 operator is moderated, considering the risk–reward trade-off associated with the accuracy of the underlying ML models.

The approach presented here recognizes that both the key considerations of convergence–diversity balance and the risk–reward trade-off could be addressed by ensuring that over the entire run of RV-EMâOA-IP2 (accounting for all generations up to termination), the share of convergence–diversity-neutral offspring (Q^{V}) outweighs the contribution of pro-convergence offspring (Q^{IP2}). Although this could be ensured in multiple ways, here it is implemented by keeping the proportion of Q^{IP2} in each generation where IP2 is invoked as 50%; and keeping $t_{\mathrm{freq}}^{\mathrm{IP2}} \geq 1$. The justification is that (i) in those generations where IP2 is invoked, Q^{V} shall still contribute 50% of N, and (ii) in generations where IP2 is not invoked due to $t_{\mathrm{freq}}^{\mathrm{IP2}} \geq 1$, Q^{V} shall contribute 100% of N. Hence, as symbolically depicted in Fig. 5.2, the overall contribution of Q^{V} across all generations is guaranteed to be dominant. Notably, in this figure Q^{V} is separated from Q^{IP2} through a fuzzy boundary. This is due to the fact that a priori quantification of Q^{V}'s exact share for an entire run of RV-EMâOA-IP2 is not possible, since $t_{\mathrm{freq}}^{\mathrm{IP2}}$ is adapted on-the-fly based on the survival rate of Q^{IP2}.

5.4 Computational Complexity of IP2 Operator

This section discusses the computational complexity of the IP2 operator, in reference to the time and space complexities of each of its constituent modules.

5.4.1 Training-Dataset Construction Module

This module pertains to the update of A_t and T_t, and *mapping* of the solutions therein. The respective time complexities of these three steps are given below.

- Update of the input archive: The process of updating A_t to A_{t+1} involves replacing the $2N$ solutions. These solutions are known and their selection does not involve any additional computation. Hence, the time complexity of this step is $\mathcal{O}(N)$.
- Update of target archive: The process of updating T_t is summarized in Algorithm 5.1. It includes $N \times N$ computations of the metric (such as PD or PBI or ASF), which has the complexity of $\mathcal{O}(M)$. Hence, the resulting time complexity is $\mathcal{O}(MN^2)$.
- Archive mapping: The overall process is summarized in Algorithm 5.2. It includes $sizeof(A_t) \times N$ computations of the association criterion. From the defined composition of A_t, $sizeof(A_t) = N(t_{\text{past}} + 1)$. Hence, the resulting time complexity of the archive mapping step is $\mathcal{O}(MN^2 t_{\text{past}})$.

During *training-dataset construction*, the input archive A_t, the target archive T_t and the training-dataset D_t are stored. Taking into account the sizes of A_t, T_t, and D_t, their space complexities are $\mathcal{O}(Nt_{\text{past}})$, $\mathcal{O}(N)$, and $\mathcal{O}(Nt_{\text{past}})$, respectively. Hence, the worst time and space complexities of this module are $\mathcal{O}(MN^2 t_{\text{past}})$ and $\mathcal{O}(Nt_{\text{past}})$, respectively.

5.4.2 ML Training Module

The training-dataset (constructed in the previous module) is used to train the RF in this module. The worst-case time complexity of training an RF is $\mathcal{O}(N_{\text{tr}} n N_{\text{sam}}^2 \log(N_{\text{sam}}))$, where N_{tr} denotes the number of trees in the RF, N_{var} denotes the number of variables or features considered in the RF, and N_{sam} denotes the number of training samples or the size of training-dataset that is used to train the RF [11]. Similarly, the worst-case space complexity of the RF is $\mathcal{O}(N_{\text{tr}} n N_{\text{sam}})$. According to the parameter settings discussed in Sect. 5.1.2, $N_{\text{tr}} = N(t_{\text{past}} + 1)$, $n = n$, and $N_{\text{sam}} = N(t_{\text{past}} + 1)$. Upon substituting their values and simplifying, the obtained time and space complexities of the ML training module are $\mathcal{O}(N^3 t_{\text{past}}^3 n \log(Nt_{\text{past}}))$ and $\mathcal{O}(N^2 t_{\text{past}}^2 n)$, respectively.

5.4.3 Offspring Progression Module

Evidently, this module (Algorithm 5.4) is executed in four steps, namely (a) selection and progression, (b) near-boundary restoration, (c) jutting, and (d) boundary repair. In the last three steps (b–d), the respective functions have a time complexity of $\mathcal{O}(n)$, which are repeated for $\lfloor 0.5N \rfloor$ solutions. Hence, their time complexity is $\mathcal{O}(Nn)$. However, in the first step of selection and progression, the trained RF model is used for progression of $\lfloor 0.5N \rfloor$ offspring. From [11], the worst-case time complexity for prediction using an RF is $\mathcal{O}(N_{\text{tr}} n N_{\text{sam}})$, which is the same as the space complexity of the RF as discussed in the previous subsection. Upon simplifying, the time complexity

Table 5.1 Time- and space complexities of different modules of the IP2 operator

Module	Time complexity	Space complexity
Training-dataset generation	$\mathcal{O}(MN^2 t_{\text{past}})$	$\mathcal{O}(N t_{\text{past}})$
ML training	$\mathcal{O}(N^3 t_{\text{past}}^3 n \log(N t_{\text{past}}))$	$\mathcal{O}(N^2 t_{\text{past}}^2 n)$
Offspring's progression	$\mathcal{O}(N^3 t_{\text{past}}^2 n)$	–

of a prediction through RF becomes $\mathcal{O}(N^2 t_{\text{past}}^2 n)$. Since $\lfloor 0.5N \rfloor$ predictions are made (considering progression of only $\lfloor 0.5N \rfloor$ offspring), the resulting time complexity is $\mathcal{O}(N^3 t_{\text{past}}^2 n)$. Moreover, since the progressed offspring replace the original offspring, there is no related space complexity of the offspring's progression module.

The worst-case time and space complexity of each constituent module of the IP2 operator is summarized in Table 5.1. Evidently, training an RF has the highest time complexity, while the trained RF model has the highest space complexity.

5.5 Experimental Setup

This section sets the foundation for experimental investigation, by highlighting the (i) test suite considered, (ii) performance indicators used and related statistical analysis, and (iii) parameters pertaining to the RV-EMâOA(s) and the IP2 operator.

5.5.1 Test Suite

To demonstrate the search efficacy infused by the IP2 operator into an RV-EMâOA, several two- and three-objective problems with varying degrees of difficulty have been used. These include the ZDT [16], DTLZ [6] and MaF [3] problems[3] with the following specifications.

- ZDT ($M = 2$): their respective $g(X)$ have been modified to have the Pareto-optimal solutions at $x_k^* = 0.5 \ \forall \ k \in \{2, \ldots, 30\}$, instead of $x_k^* = 0$. Such a shifting of optima to a non-boundary value devoids the IP2 operator of any biased advantage, owing to its boundary repair method, leading to a fair comparison. To emphasize this difference, the modified ZDT problems are referred to as Z̃DT in the remainder of this book.
- DTLZ and MaF ($M = 3$): the distance variables k have been kept as 20, to make the problems convergence-harder, compared to the generally used $k = 5$ or 10.

[3] Notably, the redundant problems such as DTLZ5, DTLZ6, and MaF6 have been excluded. Since only a small number of RVs pass through *PF* of these problems, they are ineffective in demonstrating the efficacy of the IP2 operator that learns the search directions along the RVs. In addition, DTLZ7 has been omitted since it overlaps with MaF7, already considered in the test suite.

5.5.2 Performance Indicators and Statistical Analysis

Hypervolume has been used as the primary performance indicator, as it serves as a combined indicator of convergence and diversity. Computing the hypervolume requires a reference point to be specified, which is set as $R_{1\times M} = [1 + \frac{1}{p}, \dots, 1 + \frac{1}{p}]$ [9], where p is the number of gaps set for the Das–Dennis method [4] while generating the RVs for RV-EMâO. Notably

- While for the $\tilde{\text{Z}}$DT and DTLZ problems, the scales of different objectives are equal, they differ for some MaF problems. Hence, the solutions are normalized in the $\mathcal{Z}$ space using the theoretical PF extremes for the latter.
- While the median values of hypervolume from two algorithms can be compared directly (the higher, the better), these values are subjected to a statistical analysis using the Wilcoxon ranksum test [14]. Here, the threshold p-value of 0.05 (95% confidence level) has been used.

Notably, hypervolume jointly indicates the quality of convergence and diversity, as assessed in the $\mathcal{Z}$ space. For further insights into the convergence levels in the $\mathcal{X}$ space, the population mean of the $g(X)$ function values has also been reported.

5.5.3 Parameter Settings

In this subsection, the parameters and settings used for (a) the RV-EMâOA, that is, NSGA-III, and (b) the IP2 operator, i.e., t_{past}, $t^{\text{IP2}}_{\text{freq}}$, and η, have been discussed.

5.5.3.1 RV-EMâOA Settings

With an aim to obtain a reasonably sized set of RVs using the well-known Das–Dennis method [4], the gap parameter p is set as (i) $p = 99$ for two-objective problems, leading to 100 RVs, and (ii) $p = 13$ for three-objective problems, leading to 105 RVs. For coherence, the population sizes of $N = 100$ and $N = 105$ are used in NSGA-III, for two- and three-objective problems, respectively. Furthermore, the natural variation operators include SBX ($p_c = 0.9$ and $\eta_c = 20$) and polynomial mutation ($p_m = 1/n$ and $\eta_m = 20$) for an n variable problem.

For each test instance, the performance of each RV-EMâOA has been assessed over its runs with 31 random seeds. To avoid arbitrary fixation of termination (maximum allowed) generations $t_{\max}$, a stabilization tracking algorithm [13] has been included in NSGA-III-IP2. This stabilization tracking algorithm requires a set of parameters, kept as $\psi_{\text{term}} \equiv \{3, 50\}$, which suggests $t_{\max}$ for NSGA-III-IP2 on-the-fly. The mean $t_{\max}$ determined for NSGA-III-IP2 over 31 runs has been used as $t_{\max}$ for NSGA-III.

5.5.3.2 IP2 Operator Settings

The IP2 operator involves three parameters: t_{past}, $t_{\text{freq}}^{\text{IP2}}$, and η. Here, t_{past} controls the size of the input archive A_t and the training-dataset D_t; $t_{\text{freq}}^{\text{IP2}}$ controls the invocations of the IP2 operator; and η controls the degree of jutting of the progressed offspring.

Here, $t_{\text{freq}}^{\text{IP2}}$ has been adapted on-the-fly based on the survival of the offspring, as can be observed in Algorithm 5.5. Its initial value is set as 1. To avoid an ad hoc fixation of η, it is set as a random value in [1.0, 1.5], implying a maximum of 50% extra progression along the learned search directions. While $t_{\text{freq}}^{\text{IP2}}$ and η could be rationally adapted or set, the direct impact of t_{past} on the performance of the IP2 operator is uncertain. Toward this, the behavior of NSGA-III-IP2 with respect to different values of t_{past} has been briefly discussed and investigated here.

As mentioned above, t_{past} controls the size of the training-dataset D_t. The potential implications of varying t_{past}, on the performance of the IP2 operator are mentioned below.

- A lower value of t_{past} would lead to a smaller size of D_t, and would require a lower ML training time, since its dependence on t_{past} can be given as $\mathcal{O}(t_{\text{past}}^3 \log t_{\text{past}})$. However, the size of D_t may become insufficient to train the ML model well.
- On the other hand, a higher value of t_{past} would lead to a larger (sufficient) size of D_t, but would consequently require a higher ML training time. In addition, the directional improvements (in the $\mathcal{X}$ space) learnt from a longer history of solutions may not be pertinent for subsequent generations.

In the wake of the above, it is fair to infer that the value of t_{past} should be in a certain range, such that a sufficient size of D_t can be obtained, and a higher ML training time can be avoided. Toward this, a sample parametric study on t_{past} is presented here, on three problems from different test suites, namely ŽDT1, DTLZ1, and MaF1, with $t_{\text{past}} = \{1, 3, 5, 7, 9\}$. The median hypervolume obtained by NSGA-III-IP2 at t_{max} generations determined on-the-fly is shown in Table 5.2. In that, the best-obtained hypervolume and its statistically equivalent results are marked in bold.

Interestingly, Table 5.2 suggests that for the considered problems, the performance of NSGA-III-IP2 is not very sensitive to the choice of t_{past}. Notably, among the potential values, $t_{\text{past}} = 5$ is picked for further use in this book, since it (i) emerges as the lowest value offering good performance for all the problems, and (ii) promises a moderate ML training time. It may be fair to hypothesize that even for unknown problems, $t_{\text{past}} = 5$ may suitably balance the requirements of—a reasonably sized training-dataset and moderate ML training time, *if* a reasonable population size (N) is used by the underlying EMâOA.

Table 5.2 Median hypervolume obtained by NSGA-III-IP2 with different t_{past} values

Problem	$t_{\text{past}} = 1$	$t_{\text{past}} = 3$	$t_{\text{past}} = 5$	$t_{\text{past}} = 7$	$t_{\text{past}} = 9$
ŽDT1	**0.681859**	**0.681859**	**0.681859**	**0.681859**	**0.681859**
DTLZ1	1.222185	1.222243	**1.22407**	**1.222514**	1.22193
MaF1	0.233907	**0.235205**	**0.234613**	**0.236887**	0.233841

5.6 Results and Discussions

This section compares the performance of NSGA-III-IP2 vis-à-vis NSGA-III, on a wide range of test problems. It is important to acknowledge that the performance enhancements offered by the IP2 operator could be interpreted, in terms of

1. Improvement in the *quality* of PF-approximation: this is realistic in situations where the base RV-EMâOA may fail to approximate the PF well, owing to the difficulty of the underlying problem. In such a scenario, it may be fair to expect that the IP2 operator helps improve the quality of PF-approximation.
2. Improvement in the *speed* of PF-approximation: this is realistic in situations where the underlying problem is not difficult enough, and the base RV-EMâOA run sufficiently long, and is able to approximate the PF well. In such a scenario, the IP2 operator could only speed up the PF-approximation.

Notably, the scope of the proof-of-concept results, presented here, has been restricted to the use of only one ML method, i.e., RF. However, a recent study [1] has revealed that the performance of the IP2 operator is reasonably robust and not too sensitive to the choice of underlying ML method.

5.6.1 *General Trends*

As highlighted earlier, hypervolume has been used as the primary performance indicator, supported by the $g(X)$ function, for further insights. In this background, Table 5.3 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of $t_{\max}$ generations.

In that, $t_{\max}$ has been determined on-the-fly for NSGA-III-IP2, and the same has been used for NSGA-III. From this table, the following can be observed.

- In terms of the hypervolume: NSGA-III-IP2 performs either statistically better than or equivalent to NSGA-III in 20 out of the 21 test instances.
- In terms of the $g(X)$ values: NSGA-III-IP2 performed statistically better than or equivalent to NSGA-III in each of the 18 test instances, where $g(X)$ function was existent/computable (instances, where it is not existent, are marked by '–').

To share more insights into these results, sample two- and three-objective problems where NSGA-III-IP2 reported statistically better than or equivalent hypervolume measures, than NSGA-III, are discussed below. In addition, the anomalous instance of MaF1 where NSGA-III-IP2 reported statistically worse hypervolume measures, than NSGA-III, is also discussed.

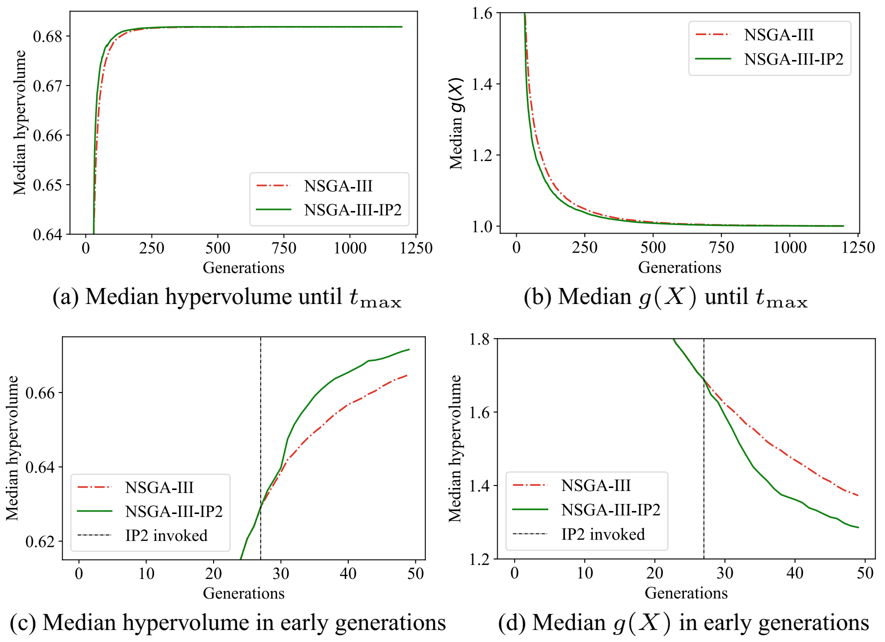
Table 5.3 Hypervolume and $g(X)$-based comparison of NSGA-III and NSGA-III-IP2 on benchmark Z̃DT, DTLZ, and MaF problems, at $t_{\max}$ generations determined on-the-fly for NSGA-III-IP2 using a stabilization tracking algorithm. In each row, the respective generations ($t_{\max}$) are shown with the median hypervolume values and median $g(X)$ values. The best performing algorithm and its statistical equivalent are marked in bold

	Problem	$t_{\max}$	Median hypervolume			Median $g(X)$			
			NSGA-III	NSGA-III-IP2	p-value	$g(X)^*$	NSGA-III	NSGA-III-IP2	p-value
$M = 2$	Z̃DT1	1197	**0.681860**	**0.681860**	8.46E-02	1	1.000732	**1.000427**	2.47E-07
	Z̃DT2	1280	0.348793	**0.348794**	3.66E-02	1	1.000580	**1.000315**	6.19E-08
	Z̃DT3	1005	**1.068427**	**1.068537**	8.98E-02	1	1.006285	**1.005028**	4.21E-04
	Z̃DT4	1768	**0.681859**	**0.681860**	4.10E-01	1	1.000040	**1.000001**	3.59E-07
	Z̃DT6	1808	0.312677	**0.319672**	6.86E-06	1	1.429366	**1.356089**	1.76E-06
$M = 3$	DTLZ1	1408	**1.221260**	**1.221979**	3.28E-01	0	**0.022966**	**0.014078**	3.35E-01
	DTLZ2	970	**0.667330**	**0.667333**	2.75E-01	0	**0.000021**	**0.000030**	8.38E-01
	DTLZ3	1658	0.649913	**0.660617**	3.78E-02	0	0.009864	**0.003851**	3.78E-02
	DTLZ4	1509	**0.667305**	**0.667316**	6.07E-01	0	**0.000021**	**0.000015**	9.05E-01
	MaF1	603	**0.236040**	0.234557	2.99E-05	0	0.001816	**0.001299**	4.32E-05
	MaF2	500	**0.396730**	**0.396523**	2.03E-01	0	0.145610	**0.097152**	8.32E-05
	MaF3	2078	**1.193480**	**1.193830**	1.49E-01	0	**0.004076**	**0.003067**	1.41E-01
	MaF4	1315	0.615647	**0.627573**	7.40E-05	0	0.023300	**0.010995**	9.35E-05
	MaF5	1344	**1.227613**	**1.227601**	8.21E-02	0	**0.000036**	**0.000047**	3.01E-01
	MaF7	1215	**0.375931**	**0.376036**	5.13E-01	1	**1.003598**	**1.003468**	7.84E-01
	MaF8	1510	**0.464343**	**0.463852**	1.16E-01	–	–	–	–
	MaF9	1326	**0.626818**	**0.626816**	6.62E-02	–	–	–	–
	MaF10	982	**0.527672**	**0.516973**	7.04E-02	0	**0.017232**	**0.018419**	6.17E-01
	MaF11	965	**0.980408**	**0.979677**	6.37E-01	0	**0.000993**	**0.001157**	8.46E-02
	MaF12	737	0.600200	**0.614154**	4.16E-09	0	0.260721	**0.180398**	8.72E-03
	MaF13	934	0.367100	**0.372636**	2.90E-03	–	–	–	–
Total $\longrightarrow$			15	**20**	of 21 probs.		8	**18**	of 18 probs.

Note (–) implies that the concerned problem does not have a $g(X)$ function; and $g(X)^* = g(X)|_{X \in PS}$, where PS is the Pareto-optimal set

5.6.2 Insights into Two- and Three-Objective Problems

For a sample discussion on a two-objective problem, the Z̃DT1 problem has been randomly chosen. Figure 5.3a and b shows the generation-wise median hypervolume and median $g(X)$ plots, respectively, among the 31 randomly seeded runs of NSGA-III and NSGA-III-IP2. The termination generation $t_{\max}$ had been set as 1197 for NSGA-III, as was determined on-the-fly for NSGA-III-IP2. Interestingly, both the hypervolume and the $g(X)$ measures suggest that the base NSGA-III alone could approximate PF well (note, $g(X)|_{X \in PS} = 1$, where PS refers to the Pareto-optimal set). Therefore, according to the premise for interpretation of the results, laid in the beginning of this section, the scope of possible enhancements by the IP2 operator



(a) Median hypervolume until $t_{\max}$

(b) Median $g(X)$ until $t_{\max}$

(c) Median hypervolume in early generations

(d) Median $g(X)$ in early generations

Fig. 5.3 Results and analysis of the IP2 operator on two-objective $\tilde{\text{Z}}$DT1 problem

reduces to *speeding up* of the PF-approximation. In fact, this is the case, as supported by Fig. 5.3c and d. In that, as the focus is restricted to only the early generations, NSGA-III-IP2 can be seen to offer superior hypervolume and $g(X)$ measures, right after the underlying IP2 operator is invoked.

As a follow-up, the MaF12 problem has been chosen for a sample discussion of a three-objective problem. Figure 5.4a and b shows the median hypervolume and median $g(X)$ plots by generation, respectively, among the 31 randomly seeded runs of NSGA-III and NSGA-III-IP2. The termination generation $t_{\max}$ was set as 737 for NSGA-III, as determined on-the-fly for NSGA-III-IP2. Evidently, this presents an instance where the base NSGA-III could not offer a good PF-approximation. This is because, even around 737 generations, the hypervolume for NSGA-III could not stabilize, and the corresponding $g(X)$ measures remained far from the desired $g(X)|_{X \in PS} = 0$. Hence, according to the premise for the interpretation of the results, laid in the beginning of this section, such a scenario points to the possibility of improvement in the *quality* of PF-approximation by the IP2 operator. This is indeed the case, as supported by

- Better hypervolume and $g(X)$ measures/trend for NSGA-III-IP2 in Fig. 5.4a and b, respectively.
- Figure 5.4c and d, where NSGA-III-IP2 can be seen to offer superior hypervolume and $g(X)$ measures, right after the underlying IP2 operator is invoked.

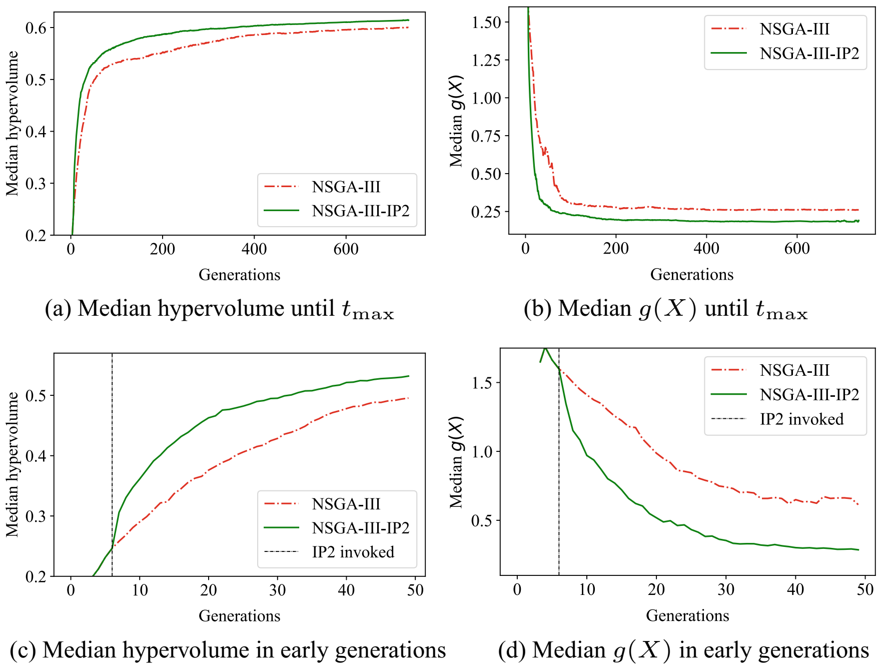
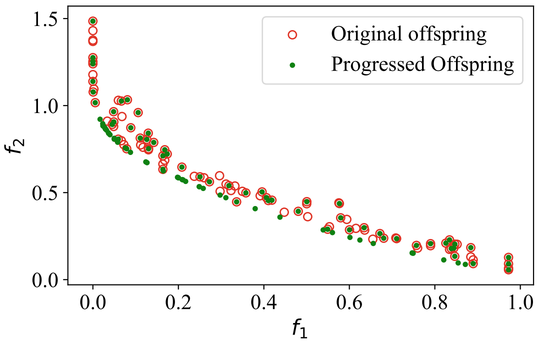


Fig. 5.4 Results and analysis of the IP2 operator on three-objective MaF12 problem

Fig. 5.5 Actual Offspring progression at $t = 35$ (a randomly chosen intermediate generation in which the IP2 operator was invoked) to visually depict the ability of the underlying RF model



While the above discussions explain the general trend of the results in Table 5.3, the focus here is to share visual insights on why the IP2 operator is able to enhance the performance of NSGA-III. Toward it, the Ž̃DT1 problem has been chosen, and the capability of the RF model trained in a randomly chosen intermediate generation in which the IP2 operator was invoked ($t = 35$, for the median run of NSGA-III-IP2) is visually depicted in Fig. 5.5. In that, nearly half of the progressed offspring population can be seen to be superior to the original offspring population, while the remaining half can be seen to overlap. This is in agreement with the offspring progression module, where only 50% of the original offspring is subjected to the

RF model, hoping for better convergence characteristics. To summarize, the IP2 operator's ability to enhance NSGA-III's performance could plausibly be attributed to the cumulative effect of

1. Direct improvements in terms of convergence, over all generations where IP2 is invoked,
2. Availability of better parents for recombination in the subsequent generations (where IP2 is not invoked). Notably, in the generation immediately after the invocation of the IP2 operator, the better fraction of the progressed offspring population is likely to be selected as recombination-contributing parents, leading to better offspring than otherwise possible. This has a recursive impact since better offspring in this generation may contribute as fitter parents in the subsequent generation, and so on.

5.6.3 Insights into the MaF1 Problem

As highlighted earlier, MaF1 happened to be the only problem in Table 5.3, where NSGA-III-IP2 reported statistically worse hypervolume measures than NSGA-III. This is accompanied by the fact that NSGA-III-IP2 still managed to have statistically better measures for $g(X)$, than NSGA-III. These trends imply that, compared to NSGA-III:

- NSGA-III-IP2 performs better in the convergence criterion ($g(X)$) but is poorer in the conjoint convergence–diversity criteria (hypervolume).
- *The loss in diversity corresponding to the use of IP2 is more significant than the corresponding gain in convergence.*

The latter is validated by Fig. 5.6, where NSGA-III can be seen to offer better diversity (in terms of both the spread and the distribution within that spread) than NSGA-III-IP2. This could, in turn, be explained through the two factors highlighted below

- MaF1 is a problem with inverted PF, which is known to pose challenges to RV-EMâOAs, since only a fraction of the underlying RVs pass through the true *PF* [10].
- The above challenge could be further augmented by the possibility that some of the 50% offspring that are randomly chosen for progression may have been the *sole* representatives of some of the RVs that pass through the true PF. After progression of the chosen offspring, some of the underlying RVs may not be left with any associated solution. This may negatively impact *PF* coverage/diversity.

Notably, the above potential pitfall points to the importance of the diversity enhancing (IP3) operator, or the unified operator (UIP) that simultaneously pursues both convergence and diversity. As already highlighted, these operators are discussed in the subsequent chapters.

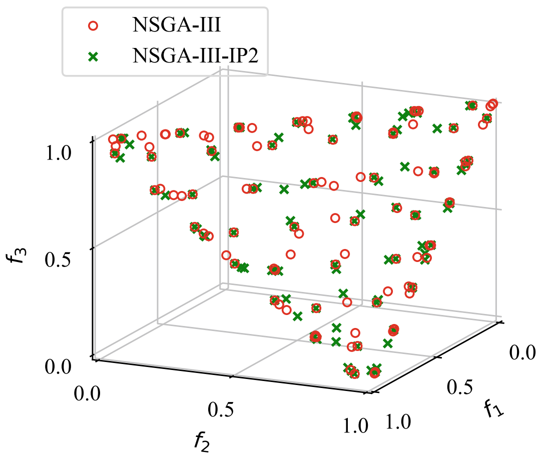


Fig. 5.6 NSGA-III and NSGA-III-IP2 populations at $t_{\max}$, for the MaF1 problem ($M = 3$)

5.7 Summary

In this chapter, the IP2 operator for convergence enhancement of RV-EMâOAs has been presented. The IP2 operator is constituted by three modules, including *training-dataset construction*, *ML training*, and *offspring progression*. The first module *maps* the inter-generational solutions *along* the RVs (generated in the *objective space*) and utilizes their underlying *variable vectors* to construct the training-dataset. The second module trains an ML model on the above dataset to *learn* the underlying directional improvements in *variable space*. The third module utilizes this trained model for progression of 50% of the natural offspring ($\lfloor 0.5N \rfloor$ in number, where N is the population size). Notably, the progressed offspring replace their underlying natural offspring, with the hope that such offspring will contribute to better convergence, over the RV-EMâOA generations. The hallmark of the IP2 operator is that its design and usage is guided by the overarching criteria to—avoid additional solution evaluations beyond those required by the base RV-EMâOA; favorably manage the convergence–diversity balance and ML-based risk–reward trade-off; and minimize ad hoc parameter fixes. Experimental results on several convergence-hard problems reveal that compared to the base NSGA-III, NSGA-III-IP2 performed better in about 29% test cases, and better or equivalent in about 95% test cases.

References

1. Bhasin, D., Swami, S., Sharma, S., Sah, S., Saxena, D.K., Deb, K.: Investigating innovized progress operators with different machine learning methods. In: Emmerich, M., Deutz, A., Wang, H., Kononova, A.V., Naujoks, B., Li, K., Miettinen, K., Yevseyeva, I. (eds.) Evolutionary Multi-Criterion Optimization, pp. 134–146. Springer Nature Switzerland, Cham (2023)

2. Breiman, L.: Random forests. Mach. Lear. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
3. Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., Yao, X.: A benchmark test suite for evolutionary many-objective optimization. Complex & Intell. Syst. **3**, 67–81 (2017). https://doi.org/10.1007/s40747-017-0039-7
4. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998)
5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, pp. 105–145. Springer London, London (2005). https://doi.org/10.1007/1-84628-137-7_6
7. Gaur, A., Deb, K.: Adaptive use of *innovization* principles for a faster convergence of evolutionary multi-objective optimization algorithms. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO '16 Companion, pp. 75–76. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2908961.2909019
8. Gaur, A., Deb, K.: Effect of size and order of variables in rules for multi-objective repair-based *innovization* procedure. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 2177–2184 (2017). https://doi.org/10.1109/CEC.2017.7969568
9. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: How to specify a reference point in hypervolume calculation for fair performance comparison. Evol. Comput. **26**(3), 411–440 (2018). https://doi.org/10.1162/evco_a_00226
10. Ishibuchi, H., Setoguchi, Y., Masuda, H., Nojima, Y.: Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. IEEE Trans. Evol. Comput. **21**(2), 169–190 (2017). https://doi.org/10.1109/TEVC.2016.2587749
11. Louppe, G.: Understanding random forests: From theory to practice (2015). arXiv:1407.7502
12. Padhye, N., Deb, K., Mittal, P.: Boundary handling approaches in particle swarm optimization. In: Bansal, J., Singh, P., Deep, K., Pant, M., Nagar, A. (eds.) Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012). Advances in Intelligent Systems and Computing, vol. 201, pp. 287–298. Springer, India (2013)
13. Saxena, D.K., Kapoor, S.: On timing the nadir-point estimation and/or termination of reference-based multi- and many-objective evolutionary algorithms. In: Deb, K., Goodman, E., Coello Coello, C.A., Klamroth, K., Miettinen, K., Mostaghim, S., Reed, P. (eds.) Evolutionary Multi-Criterion Optimization, pp. 191–202. Springer International Publishing, Cham (2019)
14. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945). http://www.jstor.org/stable/3001968
15. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007). https://doi.org/10.1109/TEVC.2007.892759
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000). https://doi.org/10.1162/106365600568202

Chapter 6
Learning to Diversify Better: IP3 Operator

It was emphasized earlier that evolutionary multi- and many-objective optimization algorithms, jointly referred to as EMâOAs, pursue the dual goals of convergence to and diversity across the true Pareto front (*PF*). In that, diversity must be interpreted in terms of the extent of *spread* (coverage of *PF*) and *uniformity of distribution* within a given spread. The goals of reference vector (RV)-based EMâOAs, namely RV-EMâOAs, are no different. In the last chapter, the machine learning (ML)-based IP2 operator has been presented aiming to enhance convergence of RV-EMâOAs, by creating pro-convergence offspring. This chapter presents an ML-based *Innovized Progress 3 (IP3)* operator to cater to the other goal of diversity, by creating pro-diversity offspring. In analogy with the IP2 case, an RV-EMâOA integrated with the IP3 operator is referred to as RV-EMâOA-IP3.

In any generation of an RV-EMâOA-IP3, if the IP3 operator gets invoked, it relies on (a) *learning* the efficient search directions in variable space ($\mathcal{X}$ space) based on a *mapping* of *intra-generational* solutions *across* the RVs, in the objective space ($\mathcal{Z}$ space); and (b) using the learned directions, for *creation* of pro-diversity offspring (in $\mathcal{X}$ space). Notably, if the population size used by RV-EMâOA-IP3 is N, then the IP3 operator—whenever invoked—contributes to $\lfloor 0.5N \rfloor$ pro-diversity offspring, referred to as Q^{IP3}. Furthermore, as in the case of IP2 operator, the frequency with which IP3 is invoked ($t^{\text{IP3}}_{\text{freq}}$) is adapted on-the-fly based on the survival rate of Q^{IP3}.

Although the details of the IP3 operator are presented in subsequent sections, a higher level depiction of how it operates when invoked is symbolically presented in Fig. 6.1. Here, the parent solutions, shown in white circles, are associated only with a few RVs. The fact that some RVs do not have an associated solution symbolizes poor distribution/diversity. The IP3 operator facilitates the progression of *some* of these parent solutions, so that the resulting solutions—considered as offspring—cover *some* of unassociated RVs, contributing to improved *uniformity*; and get driven beyond the normalized $\mathcal{Z}$ space, contributing to better *spread*. This figure also makes it intuitive that such a progression of parent solutions is possible by restricting the focus on the same-generation solutions associated with different RVs (hence, the term

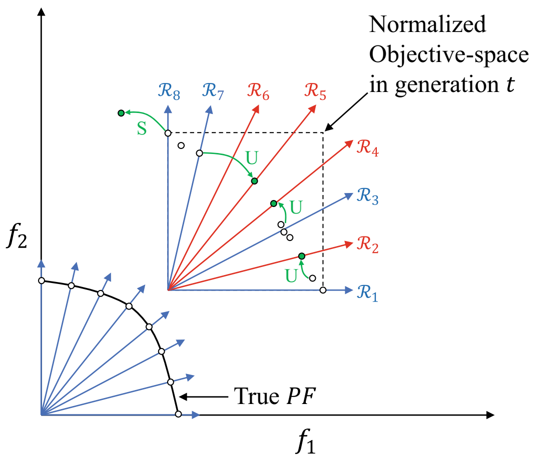


Fig. 6.1 A symbolic depiction of how the progression of *some* Parents in a given generation (shown in white circles), facilitated by the IP3 operator, can contribute to expanded *spread* (denoted by S), and improved *uniformity* (denoted by U)

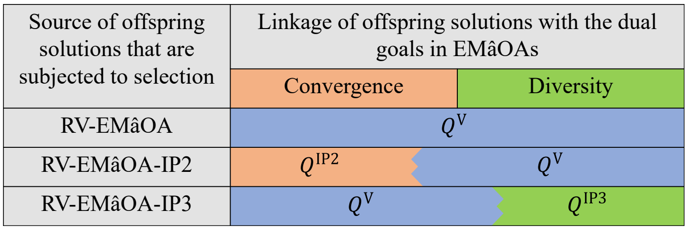
intra-generational is used), in contrast to the IP2 operator, where the use of solutions from multiple generations was necessary (hence, the term inter-generational was used).

It may be recalled that the efficacy of RV-EMâOA-IP2 over the base RV-EMâOA could be attributed to three key features of the IP2 operator—efficient management of the convergence-diversity balance and ML-based risk-reward trade-off, while avoiding any extra solution evaluations. It is significant to note that the limited details of the IP3 operator presented above suffice to establish that the IP3 operator is well poised to retain the three key features, as explained below

- In any generation of an RV-EMâOA-IP3 (working with a population size of N) run, if IP3 is invoked, it contributes to $\lfloor 0.5N \rfloor$ pro-diversity offspring. Clearly, in generations where IP3 is not invoked, all N offspring are natural[1] offspring. Hence, the overall contribution of natural offspring, Q^{V}, across all generations of an RV-EMâOA-IP3 run, is guaranteed to be dominant, as symbolically depicted[2] in Fig. 6.2. The predominance of convergence-diversity-neutral Q^{V} would ensure that the convergence-diversity balance is not disrupted and that the ML-based risk-reward trade-off is favorably managed.
- The avoidance of any extra solution evaluations by RV-EMâOA-IP3, compared to the base RV-EMâOA, is ensured by creating $\lceil 0.5N \rceil$ natural offspring and $\lfloor 0.5N \rfloor$ pro-diversity offspring through progression of the parent solutions (already evaluated).

[1] Natural offspring refer to the offspring initially created using the natural variation operators, including recombination and mutation. Such offspring are said to be convergence-diversity-neutral, since they are not created with an explicit aim to promote either convergence or diversity.

[2] Here, it is also notable that Q^{V} is separated from Q^{IP3} through a fuzzy boundary. This is due to the fact that $t^{\mathrm{IP3}}_{\mathrm{freq}}$ is adapted on-the-fly based on the survival rate of Q^{IP3}, and therefore a priori quantification of Q^{V}'s exact share for an entire run of RV-EMâOA-IP3 is not possible.



Source of offspring solutions that are subjected to selection	Linkage of offspring solutions with the dual goals in EMâOAs	
	Convergence	Diversity
RV-EMâOA	Q^{V}	
RV-EMâOA-IP2	Q^{IP2}	Q^{V}
RV-EMâOA-IP3	Q^{V}	Q^{IP3}

Fig. 6.2 Symbolic depiction of the convergence-diversity balance across all generations of RV-EMâOA-IP3 vis-à-vis RV-EMâOA-IP2 and base RV-EMâOA. The natural offspring Q^{V} do not impose any explicit preference for convergence or diversity, and, hence, are marked by a different color

The remaining chapter is organized as follows: the IP3 operator is detailed in Sect. 6.1, while its integration with NSGA-III [4], an RV-EMâOA, is presented in Sect. 6.2. Its computational complexity is highlighted in Sect. 6.3. Finally, Sect. 6.4 highlights the experimental settings, leading to the presentation of the results in Sect. 6.5.

6.1 IP3 Operator for Diversity Enhancement

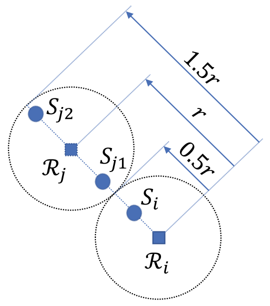
Analogous to the IP2 operator, the IP3 operator encapsulates three modules, including *Training-dataset construction*, *ML Training*, and *Offspring Creation*. The design and implementation of these modules is discussed below.

6.1.1 Training-Dataset Construction Module

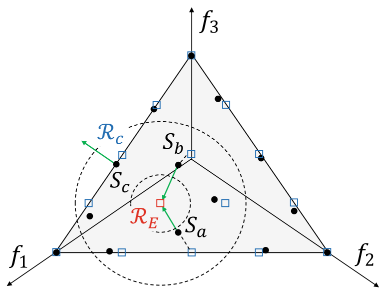
This subsection is divided into three parts, in that, first, the mapping requirements are identified vis-à-vis the end goal of diversity enhancement that the IP3 operator is expected to serve; then, the constitution of the training-dataset is presented, followed by its algorithmic implementation.

6.1.1.1 Deciphering Solution-Mapping Requirements Toward Diversity Enhancement

It must be acknowledged up front that the training-dataset is to be designed in a manner that the consequent ML model can cater to both aspects of diversity: (a) complete coverage of the true *PF*, i.e., *spread* of solutions, and (b) *uniform distribution* of solutions within the spread.



(a) Concept of neighborhood



(b) The solution-mapping challenge

Fig. 6.3 Setting the context for training-dataset construction for diversity enhancement (taken from [10])

As a precursor, the notion of *neighborhood* may be noted. Let r represent the average spacing on the unit simplex in the $\mathcal{Z}$ space between any two adjacent RVs, say $\mathcal{R}_i$ and $\mathcal{R}_j$ as in Fig. 6.3a. For a solution S_i associated with $\mathcal{R}_i$, its neighborhood is defined as the set of solutions that are associated with the adjacent $\mathcal{R}_j$, that is, $Nbd(S_i) \equiv \{S_j|\ 0.5r < dist(\mathcal{R}_i, S_j) \leq 1.5r\}$. Hence, in the context of Fig. 6.3a: $Nbd(S_i) \equiv \{S_{j1}, S_{j2}\}$. Given this, a sample scenario with scope for diversity improvement is shown in Fig. 6.3b. In that, the solutions are projected onto the unit simplex, where the RVs are created. In particular, the (RV) $\mathcal{R}_E$ does not have a solution associated with it, and the corresponding void marks a poor distribution of solutions. This could potentially be resolved if a trained ML model could enable the progression of one of the neighboring solutions (within dotted circles) to $\mathcal{R}_E$. A natural choice for progression to $\mathcal{R}_E$ is the closest solution S_a. This progression of S_a requires an improvement in f_1 and f_2, and a deterioration in f_3. Alternatively, if, say, S_a was not present, then the next closest S_b could be chosen for progression to $\mathcal{R}_E$, characterized by an improvement in f_3 and a deterioration in f_1 and f_2. These potential instances illustrate the challenge that a generic characterization of the progression of different solutions, onto an empty RV, is not possible in terms of transitions in all the objectives.

In view of this, *alternative* is that if M ML models can be trained (one for each objective, indexed as $m \in \{1, \ldots, M\}$) to help each solution improve in any specific objective, then both aspects of diversity can be improved, as highlighted below.

- *Uniform distribution*: toward it, (i) the nearest neighboring solution to an empty RV can be identified; (ii) the objective undergoing maximum change (improvement or deterioration) during the progression of this solution to the RV can be determined, say f_m; and (iii) the mth ML model can be used for the progression of the solution. For example, in Fig. 6.3b, suppose that the progression of S_a to $\mathcal{R}_E$ entails a maximum change (improvement) in f_2. Then S_a can be subjected to the second ML model. Alternatively, if the progression of S_a entails a maximum change (deterioration) in f_3, then the third ML model can be used, simply by reversing its direction of progression.

- *Spread expansion*: toward it, all boundary RVs having at least one associated solution can be identified first. This implies identifying RVs that have (i) at least one of their weight components as zero and (ii) at least one associated solution. Then, the ML model corresponding to a zero component of the RV can be used for spread expansion. For example, $\mathcal{R}_c$ in Fig. 6.3b, associated with S_c, will have its second component as zero. Therefore, the second ML model promising an improvement in f_2 could be applied to S_c, resulting in expansion of the overall spread, as symbolically highlighted by an arrow from S_c, away from the current unit simplex.

6.1.1.2 IP3 Operator's Training-Dataset

The need for M ML models points to the need for M training-dataset. Before detailing how these M dataset could be constituted, it is important to note the prerequisite step of projecting the current generation solutions in the $\mathcal{Z}$ space onto the unit simplex over which the RVs are sampled. This is needed for dimensional compatibility, so that the concept of neighborhood can be applied meaningfully. Toward this, as defined in Eq. 6.1, first for *each* objective, the normalized value $\bar{f}_m$ can be calculated using the ideal (Z^{ideal}) and nadir (Z^{nadir}) points available, following which its projected value on the unit simplex f_m^u can be calculated:

$$\bar{f}_m(X_i) = \frac{f_m(X_i) - Z_m^{\text{ideal}}}{Z_m^{\text{nadir}} - Z_m^{\text{ideal}}}, \quad f_m^u(X_i) = \frac{\bar{f}_m(X_i)}{\sum_{j=1}^{M} \bar{f}_j(X_i)}. \tag{6.1}$$

The M training-dataset can be given by $\mathcal{D} \equiv \{\mathcal{D}_1, \ldots, \mathcal{D}_M\}$, where $\mathcal{D}_m$ representing the ML model for the mth objective is defined as follows. For *any* solution S_i, let its underlying variable vector (X vector) be mapped onto the X vector of the neighbor S_j that offers *maximum improvement* in the mth objective. When repeated for *every* solution, this leads to $\mathcal{D}_m$. Hence, in principle, $\mathcal{D}_m$, which accounts for all solutions, captures the pertinent $\mathcal{X}$ space transitions that enable maximum improvement in the mth objective. Ideally, $\mathcal{D}$ can constitute a matrix of size $N \times M$. However, neighboring solutions that offer improvement in each objective may not be available for each solution. Therefore, $\mathcal{D}$ may not be fully populated.

For a visual depiction of the construction of the training-dataset, a three-objective scenario is presented in Fig. 6.4. In that, hypothetical solutions projected onto the unit simplex are shown. One of the solutions, marked as S_1, is treated as the *input* solution; $Nbd(S_1)$ has been marked by two dotted circles; and the solutions S_2 and S_3 belonging to $Nbd(S_1)$ are said to constitute the *target* cluster $\mathcal{C}$. For each objective $m \in \{1, 2, 3\}$, the selection of *target* solutions from $\mathcal{C}$ is discussed below.

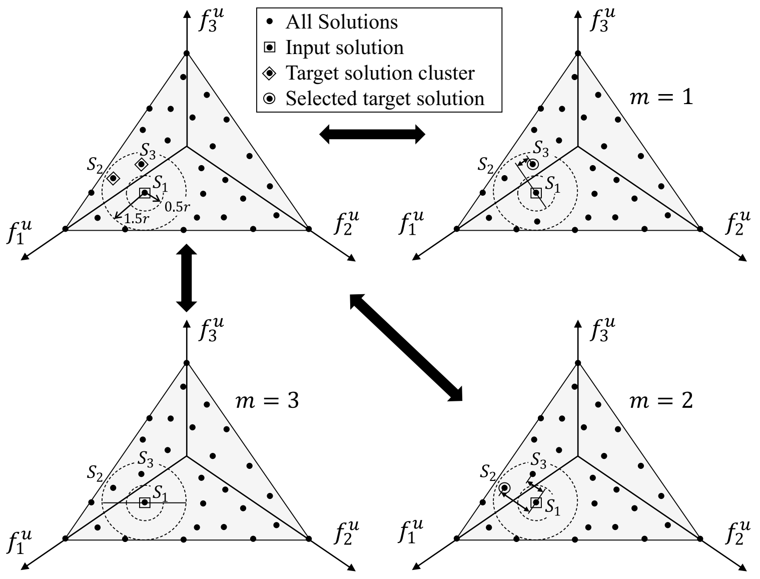


Fig. 6.4 A schematic representation of the training-dataset construction based on *objective-wise* mapping, within a predefined neighborhood, in the *projected* $\mathcal{Z}$ space. Here, S_1 is the *input* solution, and S_2 and S_3 constitute the *target* solutions for different objectives

1. $m = 1$: only one solution S_3 offers a better value in f_1^u than S_1, and hence is selected as the *target* solution.
2. $m = 2$: both S_2 and S_3 offer a better value in f_2^u than S_1. Among these, S_2 is selected as the *target* solution since it offers a greater improvement in f_2^u.
3. $m = 3$: there is no solution that offers a better value in f_3^u than S_1, implying no *target* solution. Given this, S_1 does not contribute to $\mathcal{D}_3$.

Algorithm 6.1: `Dataset_Construction` $(P_t, \mathcal{R}, r)$

```
Input: Parent population $P_t$, RVs $\mathcal{R}$, neighborhood radius $r$
Output: $M$ training-dataset $\mathcal{D}_1$–$\mathcal{D}_M$
1  Compute the projected objective values $f_1^u, \ldots, f_M^u$
2  Initialize $\mathcal{D} \equiv \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_M\}$ as an empty set
3  for each solution $S_i \in P_t$ do
4      Initialize cluster $\mathcal{C}$ as an empty set
5      for each solution $S_j \in P_t$ (other than $S_i$) do
6          if $S_j$ is in neighborhood of $S_i$: $S_j \in Nbd(S_i)$ then
7              Add solution $S_j$ to cluster $\mathcal{C}$
8      for each objective $m = 1$ to $M$ do
9          $S_j \leftarrow$ the solution in cluster $\mathcal{C}$ offering best value in $f_m^u$
10         if $f_m^u(S_j) < f_m^u(S_i)$ then
11             Add $[X(S_i), (X(S_j) - X(S_i))]$ to $\mathcal{D}_m$
```

6.1.1.3 Algorithmic Implementation of Training-Dataset Construction

In the background mentioned above, the overall procedure for constructing $\mathcal{D}_1$–$\mathcal{D}_M$ is summarized in Algorithm 6.1. In that, the first step is to calculate the projected objective values F^u, using Eq. 6.1. Subsequently, each solution $S_i \in P_t$ is considered a potential *input* solution (line 3, Algorithm 6.1), and subjected to the following steps:

1. *Identification of target solution cluster $\mathcal{C}$ (lines 4–7, Algorithm 6.1)*: the *target* cluster $\mathcal{C}$, corresponding to an *input* solution S_i, constitutes all such solutions $S_j \in P_t$ that belong to the neighborhood of S_i, implying $S_j \in Nbd(S_i)$.
2. *Identification of the target solutions from $\mathcal{C}$, toward each of $\mathcal{D}_1$–$\mathcal{D}_M$ (lines 8–11, Algorithm 6.1)*: for each objective $m \in [1, M]$, the solution $S_j \in \mathcal{C}$ offering the minimum value of $f_m^u(S_j)$ is identified. If the latter is better than $f_m^u(S_i)$, then the underlying X vectors are included in the corresponding training-dataset $\mathcal{D}_m$ (lines 11–12, Algorithm 6.1).

6.1.2 ML Training Module

The goal here is to train M ML models (one-per-dataset), such that for each objective m, an aggregated search direction in $\mathcal{X}$ space that promises improvement in f_m^u could be *learnt* from the solution-mapping embedded in $\mathcal{D}_m$. Toward it, k-nearest neighbors (kNN)[3] has been used as the ML method [3]. Notably, kNN's implementations are available for both classification and regression; however, owing to the contextual relevance, only the latter has been used here. For kNN training with regard to $\mathcal{D}_m$, when a solution's X vector is fed as the test input: (a) the kNN model identifies its k nearest *input* solutions within $\mathcal{D}_m$; and (b) *learns* the average of the corresponding *target* vectors.

Notably, a very low or high value of k (in kNN) is known to lead to overfitting or underfitting, respectively. An appropriate choice for k depends on the dataset and the intended application. In the context of the IP3 operator, $k = n$ (number of variables) has been used. This choice is based on the recognition that the training-dataset, say $\mathcal{D}_m$, involves $n \times 1$ dimensional *input* and *target* vectors. Hence, if each neighbor of the input vector differed only in one of the n elements, then $k = n$ would still allow variation in all the n elements, ensuring sufficient *input variation* and avoidance of potential overfitting. Furthermore, a sensitivity analysis for k is presented in Sect. 6.5.3.

Similar to the IP2 operator, the ML training module is executed as a two-step process: (a) a pre-training step involving normalization of the training-dataset using the *dynamic normalization* method,[4] and (b) ML training itself, as presented in Algorithm 6.2. Since the kNN model relies on the identification of the k-nearest

[3] The implementation of the kNN method has been taken from Scikit-learn package (in Python).

[4] Details of the dynamic normalization method can be found in Sect. 5.1.2 (Chap. 5).

neighbors, it is important that the dataset is normalized in the $\mathcal{X}$ space prior to training. Here, the above process is repeated M times for training M ML models.

Algorithm 6.2: `ML_Training` ($\mathcal{D}$, [$x^{(L)}$, $x^{(U)}$])

```
Input: Combined training-dataset $\mathcal{D}$, lower & upper bounds of variables specified in the
       problem, $x^{(L)}$ and $x^{(U)}$
Output: $M$ trained ML models $ML_1$–$ML_M$
1 $\{x^{(L,t)}, x^{(U,t)}\} \leftarrow$ Minimum and Maximum of each variable in $\mathcal{D}$ in generation $t$
2 Initialize $x^{\min}$ and $x^{\max}$ as empty sets
3 for $n = 1$ to $n$ do
4     $x_n^{\min} = 0.5 \times (x_n^{(L,t)} + x_n^{(L)})$
5     $x_n^{\max} = 0.5 \times (x_n^{(U,t)} + x_n^{(U)})$
6 $\bar{\mathcal{D}} \leftarrow$ Normalize $\mathcal{D}$ using $\mathbf{x}^{\min}$ and $\mathbf{x}^{\max}$ as bounds
7 for each dataset $m = 1$ to $M$ do
8     Train $ML_m$ using $\bar{\mathcal{D}}_m$
```

6.1.3 *Offspring Creation Module*

The IP3 operator is used for the progression of 50% of the parent population P_t (sized N), leading to $\lfloor 0.5N \rfloor$ pro-diversity offspring. In that, *boundary progression* and *gap progression* contribute each 25% offspring. The choice of 50% has already been justified in the beginning of this chapter, in the context of convergence-diversity balance and ML-based risk-reward trade-off. The implementation details are presented below.

6.1.3.1 Boundary Progression

The procedure for creating offspring using the boundary progression (Q_t^B) for better spread is summarized in Algorithm 6.3. This involves identifying the boundary solutions in P_t, and their progression using an appropriate ML model (rationale set earlier), as detailed below.

(a) *Identification of the boundary RVs (line 1, Algorithm 6.3)*: all boundary RVs are identified and stored in $\mathcal{R}_B$.
(b) *Identification of the solution for progression (lines 4–5, Algorithm 6.3)*: one of the RVs in $\mathcal{R}^B$ is randomly chosen, say $\vec{B}$, and its closest associated solution, say S_{start}, is chosen for progression.
(c) *Selection of the ML model (line 6, Algorithm 6.3)*: given that $\vec{B}$ is a boundary RV, some of its components may be zero. The index for one of these zero components

Algorithm 6.3: `Bound_Prog` (P_t, $\mathcal{R}$, ML, $[x^{\min}, x^{\max}]$, $[x^{(L)}, x^{(U)}]$)

```
Input: Current population $P_t$, RVs $\mathcal{R}$, ML models $ML_1$–$ML_m$, bounds from Algorithm 6.2
       $[x^{\min}, x^{\max}]$, variable bounds in problem definition $[x^{(L)}, x^{(U)}]$
Output: New solutions created $Q_t^B$
1  $\mathcal{R}^B \leftarrow$ All boundary RVs having at least one solution associated
2  Initialize $Q_t^B$ as an empty set
3  for $i = 1$ to $\lfloor 0.25N \rfloor$ do
4      $\vec{B} \leftarrow$ Randomly select an RV from $\mathcal{R}^B$
5      $S_{\text{start}} \leftarrow$ Find the solution nearest to $\vec{B}$
6      $m \leftarrow$ Randomly select an objective, such that $\vec{B}_m = 0$
7      $\bar{X}(S_{\text{start}}) \leftarrow$ Normalize $X(S_{\text{start}})$ using $x^{\min}$ and $x^{\max}$ as lower and upper bounds
8      $\bar{d_X} \leftarrow$ Pass $\bar{X}(S_{\text{start}})$ through the $m^{\text{th}}$ model $ML_m$
9      $d_X \leftarrow$ Denormalize $\bar{d_X}$ to the original $\mathcal{X}$ space using $x^{\min}$ and $x^{\max}$ as bounds
10     Obtain the new solution: $X(S_{\text{new}}) \leftarrow X(S_{\text{start}}) + \lambda_B \times \hat{d_X}$
11     Perform boundary repair on $X(S_{\text{new}})$, if required
12     Add $X(S_{\text{new}})$ to $Q_t^B$
```

(say m) is randomly selected, and ML_m becomes the model suitable for the progression of S_{start}, enabling the expansion of the boundaries.

(d) *Obtaining the search direction (lines 7–9, Algorithm* 6.3): considering that ML_m was trained on the normalized dataset: (i) $X(S_{\text{start}})$ requires normalization using the bounds computed in Algorithm 6.2, before being subjected to ML_m; and (ii) the normalized search direction ($\bar{d_X}$) resulting from the application of ML_m must be denormalized (using the same bounds) to represent the desired search direction d_X.

(e) *Progression and repair (lines 10–11, Algorithm* 6.3): the progression of S_{start} leading to S_{new} can be given by $X(S_{\text{new}}) = X(S_{\text{start}}) + \lambda_B \times \hat{d_X}$, where $\hat{d_X}$ is a unit vector along the search direction d_X (computed above), and λ_B is the step length. λ_B is chosen such that it can offer a spread *up to* twice the current spread. For symbolic depiction, Fig. 6.5 shows the unit simplex for a scenario with two objectives. In that, solutions α and γ, interspaced by $\sqrt{2}$, belong to extreme RVs. Hence, according to the specified goal, λ_B should be able to facilitate the progression of the solution γ to multiple locations bounded by γ', that is, $\sqrt{2}$ from γ. To appreciate the formulation of λ_B, consider two solutions α and β, associated with adjacent RVs. Their distance in the $\mathcal{Z}$ space can be treated as r (the spacing between their corresponding RVs). Their distance in the $\mathcal{X}$ space can be defined as the average spacing over all solutions associated with all pairwise adjacent RVs, denoted by r_X. To summarize: r in $\mathcal{Z}$ space corresponds to r_X in $\mathcal{X}$ space. Hence, spread expansion by a factor of $\sqrt{2}$ in the $\mathcal{Z}$ space will correspond to $\sqrt{2}r_X/r$ in the $\mathcal{X}$ space. Given the above, the formulation of λ_B is given by $\lambda_B = rand(0, 1) \times \sqrt{2}r_X/r$, where $rand(0, 1)$ creates a random number in the range [0, 1]. Here, the use of $rand(0, 1)$ allows the progression of

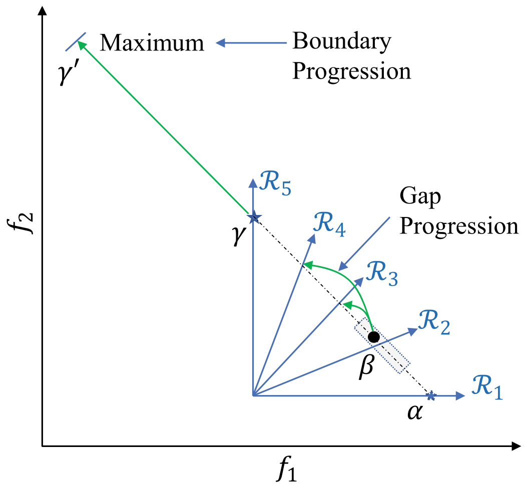


Fig. 6.5 A symbolic depiction of the boundary and gap progression on the unit simplex (taken from [10])

γ to intermittent positions up to γ'. Notably, the resulting $X(S_{\text{new}})$ may have some out-of-bound variables. These variables are repaired using the method proposed in [11].

The above steps are repeated $\lfloor 0.25N \rfloor$ times for the creation of $\lfloor 0.25N \rfloor$ offspring. In that, the random selection of a boundary solution and its progression with a random step length ensure that, across the generations, the spread expansion is fairly emphasized on all boundaries.

6.1.3.2 Gap Progression

The procedure for creating offspring using gap progression (Q_t^G) to improve the uniformity of the solutions is summarized in Algorithm 6.4. This involves the identification of empty RVs and the progression of one of their respective neighboring solutions to these empty RVs using an appropriate ML model (rationale set earlier), as detailed below.

(a) *Identification of the empty RVs (line 1, Algorithm* 6.4): all empty RVs are identified and stored in $\mathcal{R}_G$.
(b) *Identification of the solution for progression (lines 4–5, Algorithm* 6.4): one of the RVs in $\mathcal{R}_G$ is randomly selected, say $\vec{G}$. Its closest solution (associated with some other RV), say S_{start}, is chosen for progression.
(c) *Selection of the ML model (lines 6–7, Algorithm* 6.4): here, the objective in which S_{start} would undergo the maximum transition in its progression to $\vec{G}$ is identified, say f_m, such that ML_m becomes the model suited for progression.
(d) *Obtaining the search direction (lines 8–12, Algorithm* 6.4): considering that ML_m was trained on the normalized dataset: (i) $X(S_{\text{start}})$ requires normalization using

Algorithm 6.4: `Gap_Prog` (P_t, $\mathcal{R}$, ML, $[x^{\min}, x^{\max}]$, $[x^{(L)}, x^{(U)}]$)

```
Input: Current population $P_t$, RVs $\mathcal{R}$, ML models $ML_1$–$ML_M$, bounds from Algorithm 6.2
       $[x^{\min}, x^{\max}]$, variable bounds in problem definition $[x^{(L)}, x^{(U)}]$
Output: New solutions created $Q_t^G$
1  $\mathcal{R}^G \leftarrow$ Identify all empty RVs (having no solutions associated)
2  Initialize $Q_t^G$ as an empty set
3  for $i = 1$ to $\lfloor 0.25N \rfloor$ do
4      $\vec{G} \leftarrow$ Randomly select an RV from $\mathcal{R}^G$
5      $S_{\text{start}} \leftarrow$ Find the solution nearest to $\vec{G}$
6      Difference vector: $\vec{\delta} \leftarrow F^u(S_{\text{start}}) - \vec{G}$
7      $m \leftarrow$ Find the objective that maximizes $|\vec{\delta}_m|$
8      $\bar{X}(S_{\text{start}}) \leftarrow$ Normalize $X(S_{\text{start}})$ using $x^{\min}$ and $x^{\max}$ as lower and upper bounds
9      $\bar{d_X} \leftarrow$ Pass $\bar{X}(S_{\text{start}})$ through the $m^{\text{th}}$ model $ML_m$
10     $d_X \leftarrow$ Denormalize $\bar{d_X}$ to the original $\mathcal{X}$ space using $x^{\min}$ and $x^{\max}$ as bounds
11     if $\vec{\delta}_m < 0$ then
12         $d_X \leftarrow$ reverse the obtained search direction $(-1 \times d_X)$
13     Obtain the new solution: $X(S_{\text{new}}) \leftarrow X(S_{\text{start}}) + \lambda_G \times \hat{d_X}$
14     Perform boundary repair on $X(S_{\text{new}})$, if required
15     Add $X(S_{\text{new}})$ to $Q_t^G$
```

the bounds computed in Algorithm 6.2, before being subjected to ML_m; and (ii) the normalized search direction ($\bar{d_X}$) resulting from application of ML_m needs to be denormalized (using the same bounds), to represent the sought search direction d_X. In the case where the progression of S_{start} to $\vec{G}$ is characterized by deterioration in f_m, d_X is reversed by multiplying each of its components by (-1), as depicted in line 12 (Algorithm 6.4).

(e) *Progression and repair (lines 13–14, Algorithm* 6.4): the progression of S_{start} leading to S_{new} can be given by $X(S_{\text{new}}) = X(S_{\text{start}}) + \lambda_G \times \hat{d_X}$, where $\hat{d_X}$ is a unit vector along the search direction d_X (computed above), and λ_G is the step length. λ_G is chosen in a manner that the progression of a solution to $\vec{G}$ could be executed, regardless of whether or not that solution belongs to the neighborhood of $\vec{G}$. For symbolic depiction, Fig. 6.5 shows the potential progression of solution β, to its neighbor $\mathcal{R}_3$ and also non-neighbor $\mathcal{R}_4$. Let J be the (rounded-off) integer number of RV transitions required for progression of an existing solution onto an empty RV. In the above example, $J = 1$ for $\mathcal{R}_3$ and $J = 2$ for $\mathcal{R}_4$. Given this context and the mapping of r and r_X established earlier, the progression of S_{start} to $\vec{G}$ can be represented as $J \times r$ in $\mathcal{Z}$ space, and $J \times r_X$ in $\mathcal{X}$ space. Critically, S_{start} may not necessarily lie *on* the RV. Rather, it may lie anywhere within $\pm 0.5r$ around the RV. This is illustrated in Fig. 6.5, where β may lie anywhere in the engulfing rectangle characterized by $\pm 0.5r$. To accommodate this uncertainty, the formulation of λ_G is given by $\lambda_G = rand(J \pm 0.5) \times r_X$. Notably, the obtained $X(S_{\text{new}})$ may have some out-of-bound variables, which can be repaired using the same method as used in boundary progression.

These steps are repeated $\lfloor 0.25N \rfloor$ times for the creation of $\lfloor 0.25N \rfloor$ offspring. In that, the random selection of empty RVs ensures that each identified gap in the current population is emphasized fairly.

6.2 Integration of IP3 Operator into NSGA-III

This section describes the integration of the IP3 operator with NSGA-III, leading to NSGA-III-IP3. This integration, summarized in Algorithm 6.5 is generic in nature and can be extended to any other RV-EMâOA.

Algorithm 6.5: Generation t of NSGA-III-IP3

```
Input: RV set $\mathcal{R}$, original variable bounds $[x^{(L)}, x^{(U)}]$, Parent population $P_t$, frequency of
       progression $t^{\text{IP3}}_{\text{freq}}$, neighborhood radius $r$, number of survived offspring in $(t-1)^{\text{th}}$
       generation $N^{\text{surv}}_{t-1}$
Output: $P_{t+1}$, $t^{\text{IP3}}_{\text{pg}}$, $t^{\text{IP3}}_{\text{freq}}$, $N^{\text{surv}}_t$
1  if population has mildly stabilized then
2      $startIP3 = True$
3  if $startIP3$ & $t^{\text{IP3}}_{\text{freq}}$ generations passed after last invocation then
4      $\mathcal{D} \leftarrow$ Construct the training-dataset using Algorithm 6.1
5      Train the $M$ ML models using Algorithm 6.2
6      $Q^B_t \leftarrow$ Create 25% offspring using boundary progression (Algorithm 6.3)
7      $Q^G_t \leftarrow$ Create 25% offspring using gap progression
       (Algorithm spsrefalgo:IP3spsgapspsprog)
8      $Q^V_t \leftarrow$ Create rest 50% offspring using natural variation operators
9      $Q_t \leftarrow$ Merge $Q^B_t$, $Q^G_t$ and $Q^V_t$ (total $N$ offspring)
10 else
11     $Q_t \leftarrow$ Create 100% offspring using natural variation operators
12 Evaluate $Q_t$
13 $P_{t+1} \leftarrow$ Perform the survival selection on $P_t \cup Q_t$
14 $N^{\text{surv}}_t \leftarrow$ Count of offspring $Q_t$ that survived to $P_{t+1}$
15 if IP3 was invoked in current generation then
16     if $N^{\text{surv}}_t > N^{\text{surv}}_{t-1}$ then reduce $t^{\text{IP3}}_{\text{freq}}$ by 1
17     if $N^{\text{surv}}_t < N^{\text{surv}}_{t-1}$ then increase $t^{\text{IP3}}_{\text{freq}}$ by 1
```

Notably, Algorithm 6.5 represents any intermediate generation t of NSGA-III-IP3, and involves a new parameter, namely $t^{\text{IP3}}_{\text{freq}}$, which specifies the number of generations between two successive progressions. In Algorithm 6.5, first the prerequisite condition for invocation of the IP3 operator is checked, that is, mild population stabilization. To do that, a stabilization tracking algorithm [12] has been used.[5] If

[5] The same algorithm can be used for (a) triggering the IP3 operator with a mild setting, and (b) terminating an RV-EMâOA run with a strict setting.

stability is detected, the *startIP*3 flag is marked as *True* (lines 1–2, Algorithm 6.5). It is important to let the population mildly stabilize (ensuring some degree of convergence) before applying the IP3 operator; otherwise, premature focus on diversity enhancement may lead to a delayed convergence, at the cost of additional computational expense. Notably, the underlying stabilization tracking algorithm requires an additional set of parameters (ψ_{mild}) to detect mild stabilization, which is discussed in Sect. 6.4.3.2. Subsequently:

- If $t^{\text{IP3}}_{\text{freq}}$ generations have passed after the last invocation of the IP3 operator, then IP3 is invoked, leading to the creation of 50% offspring using the `IP3` function; and rest 50% natural offspring, leading to Q_t, sized N (lines 3–6, Algorithm 6.5).
- Otherwise, if IP3 is not invoked, all (100%) natural offspring are created (lines 7–8, Algorithm 6.5).
- The offspring Q_t are evaluated and the survival selection procedure of NSGA-III is executed (lines 9–10, Algorithm 6.5).
- The count of offspring N^{surv}_t that survived to the next generation is estimated. In a generation where IP3 is invoked, if this count has improved compared to the previous generation, implying good performance of the IP3 operator, then $t^{\text{IP3}}_{\text{freq}}$ is reduced by 1, resulting in a more frequent progression. Otherwise, if the count has reduced, $t^{\text{IP3}}_{\text{freq}}$ is increased by 1 (lines 11–14, Algorithm 6.5).

Notably, in the absence of gaps in the population, that is, if no RVs are unassociated, then the $\lfloor 0.25N \rfloor$ offspring solutions created using gap progression are created using boundary progression, ensuring that overall $\lfloor 0.5N \rfloor$ offspring solutions are created using the IP3 operator.

6.3 Computational Complexity of IP3 Operator

As described in the previous section, the IP3 operator consists of three modules. The following subsections discuss the time and space complexities of each constituent module, followed by its overall summary.

6.3.1 Training-Dataset Construction Module

The process of constructing M training-dataset $\mathcal{D}_1$–$\mathcal{D}_M$ has been summarized in Algorithm 6.1. In that, for each solution in P_t, first the *target* solutions cluster is identified which requires N computations and then, the *target* solutions are picked for each dataset, which requires $M \times M$ computations. Collectively, this procedure is repeated for each solution in P_t (sized N). Given that the above procedure is repeated for N solutions, the resulting time complexity is $\max\{\mathcal{O}(N^2), \mathcal{O}(NM^2))\}$. Generally, the population sizes N used in RV-EMâOAs (as also used in this book) satisfy $N \geq M^2$. Hence, the resulting time complexity of this module can be approximated as $\mathcal{O}(N^2)$.

Furthermore, only the training-dataset $\mathcal{D}_1$–$\mathcal{D}_M$ are additionally created, over the base NSGA-III algorithm. Each dataset in $\mathcal{D}$ can have a maximum of N mapped pairs of solutions, considering which the resulting space complexity of this module is $\mathcal{O}(MN)$.

6.3.2 ML Training Module

The training-dataset constructed above is used to train M kNN regressor models in this module. The worst-case time complexity of training a kNN model is $\mathcal{O}(N_{\text{dim}}N_{\text{sam}}\log(N_{\text{sam}}))$, where N_{dim} denotes the dimensionality of the training-dataset and N_{sam} denotes the number of samples. Similarly, the worst-case space complexity of kNN is $\mathcal{O}(N_{\text{dim}}N_{\text{sam}})$. In this case, $N_{\text{dim}} = n$, $\max(N_{\text{sam}}) = N$, and M trainings are executed. Upon substituting their values, the resulting time and space complexities of the ML training module are $\mathcal{O}(MNn\log(N))$ and $\mathcal{O}(MNn)$, respectively.

6.3.3 Offspring Creation Module

Evidently, this module constitutes two submodules, each of which follows the same procedure from the computational complexity perspective. Hence, a common discussion is presented here. Each submodule is executed through four steps: (a) identification of the solution for progression; (b) selection of the ML model; (c) identification of the search direction; and (d) progression and repair. The worst-case computational complexity of Step-(a) is $\mathcal{O}(MN)$, owing to the identification of the closest solution to a given RV. Step-(b) requires $M + M$ computations, leading to a complexity of $\mathcal{O}(M)$. Step-(c) involves making a prediction using one of the learnt kNN models. The prediction time complexity of a kNN model is $\mathcal{O}(kN_{\text{sam}})$, where k is the number of neighbors. Since $k = n$ has been used, the resulting time complexity for making a prediction is $\mathcal{O}(Nn)$. Finally, Step-(d) involves the progression of a given solution with $\mathcal{O}(M)$ complexity. Among the four steps, the worst time complexity can be given as $\max\{\mathcal{O}(MN), \mathcal{O}(Nn)\}$. Generally, $n > M$ is the majority of the multi-objective problems (MOPs). Hence, the worst-case time complexity can be approximated as $\mathcal{O}(Nn)$. Since these steps are repeated for $\lfloor 0.25N \rfloor$ solutions, the resulting time complexity becomes $\mathcal{O}(N^2n)$. Furthermore, since the offspring solutions created through progression are included in the N offspring solutions that are created by any RV-EMâOA, there is no related space complexity of this module.

Table 6.1 Time and space complexities of different modules of the IP3 operator

Module	Time complexity	Space complexity
Training-dataset construction	$\mathcal{O}(N^2)$	$\mathcal{O}(MN)$
ML training	$\mathcal{O}(MNn\log(N))$	$\mathcal{O}(MNn)$
Offspring creation	$\mathcal{O}(N^2n)$	–

The worst-case time complexity and space complexity of each constituent module of the IP3 operator are summarized in Table 6.1. Evidently, training of M kNN models has the highest time complexity, and the M trained kNN models have the highest space complexity.

6.4 Experimental Setup

This section sets the foundation for experimental investigation, by highlighting the (a) test suite considered; (b) performance indicators used and related statistical analysis; and (c) parameters pertaining to the RV-EMâOA(s) and the IP3 operator.

6.4.1 Test Suite

To demonstrate the search efficacy infused by the IP3 operator into an RV-EMâOA, several two- and three-objective problems with varying degrees of difficulty have been used. These include CIBN [7], DASCMOP [5], and MW [9] problems with the following specifications:

- CIBN: CIBN1–3 are two-objective problems, and CIBN4–5 are three-objective problems, with $n = 10$.
- DASCMOP: DASCMOP1–6 are two-objective problems, and DASCMOP7–9 are three-objective problems, with $n = 30$. Since different difficulty settings are available for these problems [5], setting 5 has been used here that corresponds to the diversity-hardness in these problems.
- MW: majority of these problems are two-objective, except for MW4, MW8, and MW14 which are three-objective. Here, $n = 15$ for all the problems MW1–14.

Notably, all these test problems have been proposed in recent years, and are diversity-hard due to the presence of multiple constraints. Standard EMâOAs, such as NSGA-III, that rely on the constraint dominance principle for handling constraints, exhibit severe under-performance in some of these problems [8].

6.4.2 Performance Indicators and Statistical Analysis

The choice of performance indicators is exactly the same as adopted earlier in Chap. 5. The key details are reiterated below

- Hypervolume is used as the primary indicator with the reference point given as $R_{1\times M} = [1 + \frac{1}{p}, \ldots, 1 + \frac{1}{p}]$, where p is the number of gaps set for the Das–Dennis method while generating the RVs for RV-EMâOAs. Further, for the problems where the scales of different objectives are different, the solutions are normalized in the $\mathcal{Z}$ space using the true *PF* extremes.
- The population mean of the $g(X)$ function is used as the secondary indicator, to provide insights into the convergence levels in the $\mathcal{X}$ space. Although the IP3 operator explicitly focuses on diversity enhancement, it would be intriguing to assess whether there is an adverse impact on the convergence of solutions.

Further, in the context of the statistical analysis on these performance indicators:

- When comparing *only two* algorithms, at a time, the Wilcoxon ranksum [13] test is performed on the indicator values reported over multiple/independently seeded runs. In that, the p-value of 0.05 (95% confidence interval) is used as threshold.
- When comparing *more than two* algorithms, at a time, the Kruskal–Wallis test [6] with the threshold p-value of 0.05 is used to infer if their overall differences are statistically insignificant or not. If not, the Wilcoxon ranksum test is used for their pairwise comparisons, when the algorithm reporting the best median hypervolume is treated as the reference. Furthermore, the threshold p-value is adjusted using the standard Bonferroni correction [1], to retain the same overall confidence.

6.4.3 Parameter Settings

In this subsection, the parameters and settings used for (a) the RV-EMâOA, i.e., NSGA-III; and (b) the IP3 operator, i.e., r, $t^{\text{IP3}}_{\text{freq}}$, η_j, and ψ_{mild}, have been discussed.

6.4.3.1 RV-EMâOA Settings

These settings have been kept exactly the same as those used for the IP2 operator in Chap. 5. The key details are reiterated below.

- For generating RVs, Das–Dennis method has been used, with (a) $p = 99$ for $M = 2$, leading to $N = 100$ and (b) $p = 13$ for $M = 3$, leading to $N = 105$.
- The natural variation operators include (a) SBX crossover, with $p_c = 0.9$ and $\eta_c = 20$, and (b) polynomial mutation, with $p_c = 1/n$ and $\eta_m = 20$.
- Each of NSGA-III and NSGA-III-IP3 has been run 31 times, with random seeds.

- For NSGA-III-IP3, $t_{\max}$ has been determined on-the-fly using the stabilization tracking algorithm, using $\psi_{\text{term}} \equiv \{3, 50\}$. For NSGA-III, the mean $t_{\max}$ determined for NSGA-III-IP3 over 31 runs has been used as $t_{\max}$.

6.4.3.2 IP3 Operator Settings

The IP3 operator involves three parameters: r, $t_{\text{freq}}^{\text{IP3}}$, and ψ_{mild}. In that, the neighborhood radius r governs the identification of the cluster of target solutions during mapping; $t_{\text{freq}}^{\text{IP3}}$ controls the invocations of the IP3 operator; and ψ_{mild} (similar to ψ_{term}) governs the degree of stabilization required to trigger the first invocation of the IP3 operator during an RV-EMâOA-IP3 run.

Here, r is simply derived from the given RV set $\mathcal{R}$, and $t_{\text{freq}}^{\text{IP3}}$ has been adapted on-the-fly based on the survival of the offspring, as can be observed in Algorithm 6.5. The initial value of $t_{\text{freq}}^{\text{IP3}}$ is set to 1. While the first two (of three) parameters could be rationally derived or adapted, a direct impact of ψ_{mild} on the performance of the IP3 operator is not that straightforward.

As mentioned above, ψ_{mild} governs the degree of stabilization required to trigger the first invocation of the IP3 operator. While it is intuitive that ψ_{mild} should correspond to a lower degree of stabilization than ψ_{term} that is used to terminate the NSGA-III-IP3 run, its exact setting is borrowed from the suggestion made in [12]. According to that, the mild stabilization corresponds to $\psi_{\text{mild}} = \{2, 20\}$.

6.5 Experimental Results

This section compares the performance of NSGA-III-IP3 vis-à-vis NSGA-III that includes (a) an assessment of the general performance trends on a wide range of test problems; and (b) an investigation on some sample two- and three-objective problems. In the end, an assessment of the sensitivity of the IP3 operator's performance with the variation in k (used for kNN, the underlying ML method) has been presented. Notably, the premise laid earlier in Sect. 5.6 (Chap. 5), in context of the interpretation of results with IP2 operator, is also applicable for the results with IP3 operator.

Further, the scope of the proof-of-concept results, presented here, has been restricted to the use of only one ML method, i.e., kNN. However, a recent study [2] has revealed that the performance of the IP3 operator is reasonably robust and not too sensitive to the choice of underlying ML method.

6.5.1 General Trends

As highlighted earlier, hypervolume has been used as the primary performance indicator, supported by the $g(X)$ function values for further insights. In this context, Table 6.2 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of $t_{\max}$ generations. In that, $t_{\max}$ has been

Table 6.2 Hypervolume and $g(X)$-based comparison of NSGA-III and NSGA-III-IP3 on benchmark CIBN, DASCMOP, and MW problems, at $t_{\max}$ generations determined on-the-fly for NSGA-III-IP3 using a stabilization tracking algorithm. Each row shows the median hypervolume and $g(X)$ values at the end of $t_{\max}$ generations. The best performing algorithm and its statistical equivalent are marked in bold

	Problem	$t_{\max}$	Median hypervolume			Median $g(X)$			
			NSGA-III	NSGA-III-IP3	p-value	$g(X)^*$	NSGA-III	NSGA-III-IP3	p-value
$M=2$	CIBN1	1404	0.328355	**0.483381**	1.34E-11	0	**0.000581**	0.001742	1.34E-11
	CIBN2	753	0.655604	**0.669094**	2.89E-11	0	0.003847	**0.002461**	1.48E-09
	CIBN3	889	0.213285	**0.219149**	1.34E-11	0	0.003371	**0.003022**	1.05E-04
	DASCMOP1	1948	0.089614	**0.31699**	3.22E-09	0	**0.000266**	0.004733	1.34E-11
	DASCMOP2	1793	0.414381	**0.637702**	1.34E-11	0	**0.000292**	0.012345	1.34E-11
	DASCMOP3	1639	0.391362	**0.39416**	4.95E-02	0	**0.000525**	**0.000899**	1.66E-01
	DASCMOP4	1978	**0.336838**	**0.336812**	7.95E-01	0	**0.000149**	**0.000158**	8.60E-01
	DASCMOP5	2101	**0.672598**	**0.672677**	3.21E-01	0	**0.000136**	**0.000129**	9.61E-01
	DASCMOP6	2377	0.549901	**0.574818**	2.53E-03	0	**0.000093**	**0.000072**	6.83E-02
	MW1	1047	**0.415296**	**0.415318**	4.68E-01	1	**1.000057**	**1.000034**	1.13E-01
	MW2	836	**0.482964**	**0.482899**	7.95E-01	1	**1.020645**	**1.020625**	9.49E-01
	MW3	875	**0.469838**	**0.469803**	5.54E-01	1	**1.041892**	1.043891	1.23E-03
	MW5	1821	0.083018	**0.197173**	1.86E-04	1	**1.000027**	1.000176	1.81E-05
	MW6	1229	**0.298354**	**0.298348**	9.94E-01	1	**1.026767**	**1.026708**	9.61E-01
	MW7	892	**0.366328**	**0.366413**	5.59E-01	1	**1.094907**	**1.096497**	1.53E-01
	MW9	1071	0.293771	**0.29641**	7.47E-04	1	1.452991	**1.42566**	1.48E-09
	MW10	1063	**0.246928**	**0.247365**	8.38E-01	1	**1.049239**	**1.05019**	8.82E-01
	MW11	961	**0.268168**	0.259526	2.66E-02	1	**1.277597**	**1.243012**	7.04E-02
	MW12	1068	0.570536	**0.570814**	3.81E-03	1	**1.246089**	1.249201	3.65E-03
	MW13	972	**0.328753**	**0.328191**	7.41E-01	1	**1.069005**	**1.072882**	4.68E-01
$M=3$	CIBN4	438	0.912571	**0.917063**	9.39E-03	0	**0.012301**	0.014547	1.01E-03
	CIBN5	287	**0.629831**	**0.629746**	6.07E-01	0	**0.008496**	**0.008635**	5.40E-01
	DASCMOP7	1693	**1.02602**	**1.025306**	5.31E-01	0	**0.001225**	**0.001369**	6.27E-01
	DASCMOP8	1650	0.628281	**0.658097**	1.02E-02	0	0.010428	**0.001833**	1.15E-02
	DASCMOP9	1539	0.346689	**0.647012**	1.34E-11	0	**0.004983**	**0.005401**	2.34E-01
	MW4	743	**1.041376**	**1.041362**	7.09E-01	1	**1.000239**	**1.000337**	4.68E-01
	MW8	718	**0.626856**	**0.626362**	9.49E-01	1	**1.014327**	**1.014104**	7.51E-01
	MW14	914	**0.154225**	**0.158839**	2.13E-01	1	**1.016592**	**1.017586**	9.83E-01
Total $\longrightarrow$			15	**27**	of 28 probs.		**24**	21	of 28 probs.

Note $g(X)^* = g(X)|_{X \in UPS}$, where UPS denotes the Pareto-optimal set for the unconstrained version of the MOP determined on-the-fly for NSGA-III-IP3, and the same has been used for NSGA-III. From this table, the following can be observed.

- In terms of hypervolume: NSGA-III-IP3 performed either statistically better than or equivalent to NSGA-III in 27 out of 28 test instances.
- In terms of $g(X)$ values: NSGA-III-IP3 performed either statistically better than or equivalent to NSGA-III in only 21 out of 28 instances, whereas NSGA-III performed either statistically better than or equivalent to NSGA-III-IP3 in 24 out of 28 instances.

The preliminary conclusion from the above trends is that overall, NSGA-III-IP3 performed better in hypervolume but worse in $g(X)$ values, than NSGA-III. While the latter could be attributed to the partial shift in focus of offspring creation (due to IP3 operator) toward diversity enhancement, the former suggests that the improvement in diversity was significantly higher than the loss in convergence, leading to better hypervolume values. This clearly endorses the search efficacy infused by the IP3 operator into NSGA-III, toward diversity enhancement.

6.5.2 *Insights into Two- and Three-Objective Problems*

For further insights into the performance of IP3 operator, some sample test instances have been chosen for discussion, including (a) CIBN1—a two-objective problem where NSGA-III fails to achieve a reasonable *PF* approximation; (b) MW12—a two-objective problem where NSGA-III achieves a reasonable *PF* approximation; and (c) DASCMOP9—a three-objective problem, as presented below.

6.5.2.1 CIBN1 Problem ($M = 2$)

Here, the CIBN1 problem is chosen for a sample discussion on a two-objective problem where NSGA-III fails to achieve a reasonable *PF*-approximation. Figure 6.6 shows the final set of solutions obtained in the respective median runs of NSGA-III and NSGA-III-IP3 (out of the 31 randomly seeded runs each). The termination generation $t_{\max}$ has been set as 1404 for NSGA-III, determined on-the-fly for NSGA-III-IP3. As evident, this presents an instance where NSGA-III could not offer a good *PF*-approximation since the obtained spread of solutions is only a subset of the actual spread of the *PF*. Hence, such a scenario points to the possibility of improving the *quality* of the *PF* approximation.

As can be observed in Fig. 6.6, NSGA-III-IP3 clearly achieved a better *PF* approximation than NSGA-III. Although NSGA-III-IP3 could not achieve the desired spread across the true *PF*, the improvement in the spread of solutions over NSGA-III could only be attributed to the search efficacy infused by the IP3 operator toward diversity enhancement.

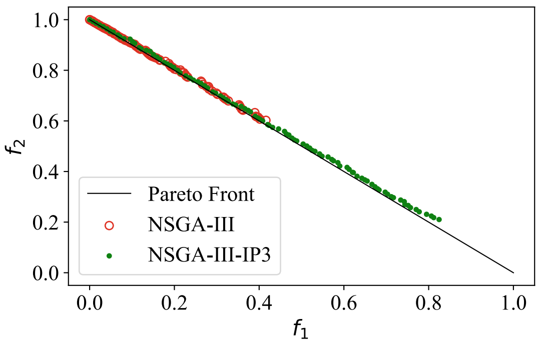
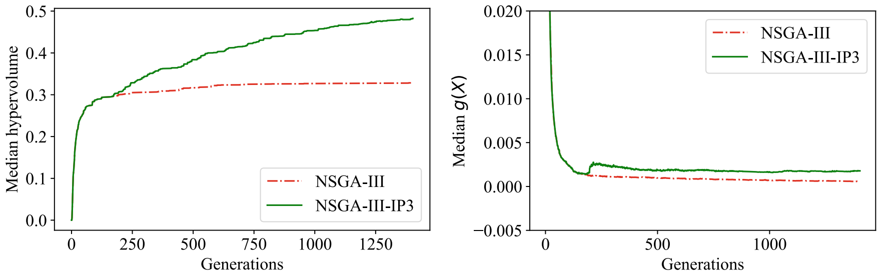


Fig. 6.6 Final obtained solutions in the respective median runs of NSGA-III and NSGA-III-IP3 on CIBN1 problem

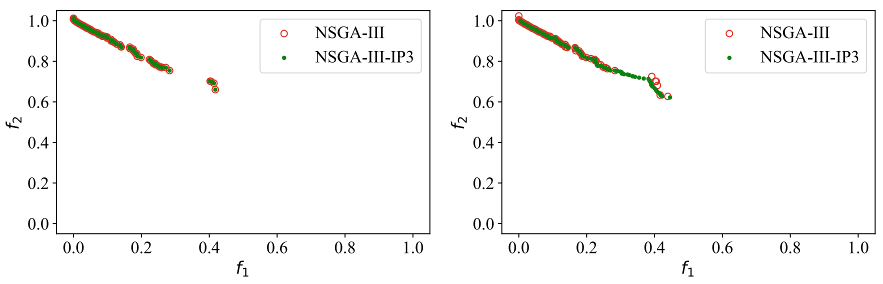


(a) Hypervolume plot (b) $g(X)$ plot

Fig. 6.7 Generation-wise performance of NSGA-III and NSGA-III-IP3 on CIBN1

For further insights into performance, Fig. 6.7a and b shows the generation-wise median hypervolume and median $g(X)$ plots, respectively. In that, there is a significant improvement in hypervolume, but a slight deterioration in the $g(X)$ value, suggesting a slightly poorer convergence of NSGA-III-IP3. However, the improvement in hypervolume suggests that the better diversity across the *PF* compensates and overcomes the loss in convergence to the *PF*. Moreover, it can be observed that even at $t_{\text{max}} = 1404$, the hypervolume measures for NSGA-III-IP3 could not stabilize. This suggests that although the population had stabilized according to the defined termination criterion, and the NSGA-III-IP3 run should have been terminated from a practical perspective, there was scope for further improvement in spread across the *PF*.

It may be noted that the IP3 operator attempts to cater to both aspects of diversity enhancement, that is, spread expansion and uniformity within the spread. While the first aspect has clearly been demonstrated through the discussion presented above in the context of the CIBN1 problem, it is imperative to gather insights into the performance of the IP3 operator in terms of achieving a better uniformity of solutions within a given spread. Toward it, the parent solutions in generation $t_{\text{mild}} = 178$, right before the IP3 operator is invoked for the first time, are shown in Fig. 6.8a; and



(a) Parent solutions $t = t_{\text{mild}} = 178$ (b) Parent solutions $t = 200 > t_{\text{mild}}$

Fig. 6.8 Parent solutions in the respective median runs of NSGA-III and NSGA-III-IP3 on CIBN1 problem at $t = t_{\text{mild}} = 178$, and at $t = 200$ (an arbitrary generation afterward). Notice how the spread is similar in both generations, but the uniformity of solutions in NSGA-III-IP3 has improved significantly, owing to the gap progression submodule of the IP3 operator

the parent solutions in a subsequent arbitrary generation, say $t = 200$, are shown in Fig. 6.8b. As can be observed in $t = t_{\text{mild}}$, the solutions of NSGA-III and NSGA-III-IP3 coincide, implying that NSGA-III-IP3 behaves exactly the same as NSGA-III until the generation in which the IP3 operator is invoked for the first time. In that, there is a perceivable gap between the solutions, which is applicable to both NSGA-III and NSGA-III-IP3. However, at $t = 200$, while there is still a gap between the solutions of NSGA-III, the uniformity of the solutions has improved significantly for NSGA-III-IP3. This improvement in the uniformity of solutions can only be attributed to the efficacy of the IP3 operator.

6.5.2.2 MW12 Problem ($M = 2$)

Here, the MW12 problems are considered for discussion, where NSGA-III is able to achieve a reasonable *PF*-approximation. Reference may be made to Fig. 6.9a and b, which presents the generation-wise median hypervolume and $g(X)$ plots, respectively, among the 31 randomly seeded runs of NSGA-III and NSGA-III-IP3. The termination generation t_{max} had been set as 1068 for NSGA-III, determined on-the-fly for NSGA-III-IP3. As evident, both NSGA-III and NSGA-III-IP3 achieved a similar *PF*-approximation at the end of t_{max} generations. Hence, the scope of possible enhancements by the IP3 operator reduces to *speeding up* of the *PF*-approximation. This is supported by the reference to an arbitrarily chosen generation $t = 250$ in Fig. 6.9a, where NSGA-III-IP3 clearly achieved a better hypervolume than NSGA-III.

Toward a deeper investigation, the solutions obtained in the respective median runs of NSGA-III and NSGA-III-IP3 at $t = t_{\text{max}}$ and at $t = 250$ are shown in Fig. 6.10a and b, respectively. In that, while the solutions at $t = t_{\text{max}}$ reflect similar performance of NSGA-III and NSGA-III-IP3, the solutions at $t = 250$ clearly reflect the better

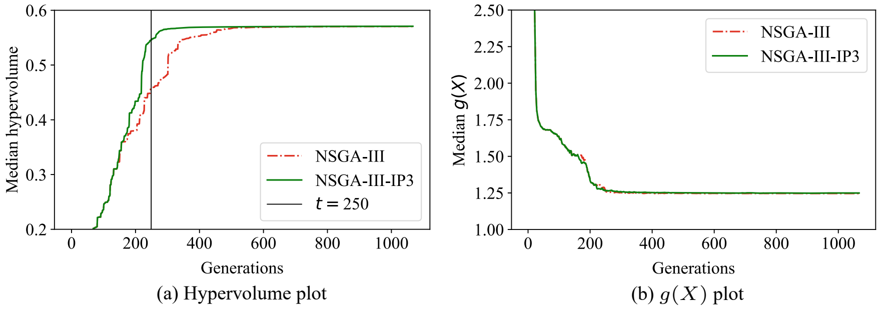


Fig. 6.9 Generation-wise performance of NSGA-III and NSGA-III-IP3 on MW12

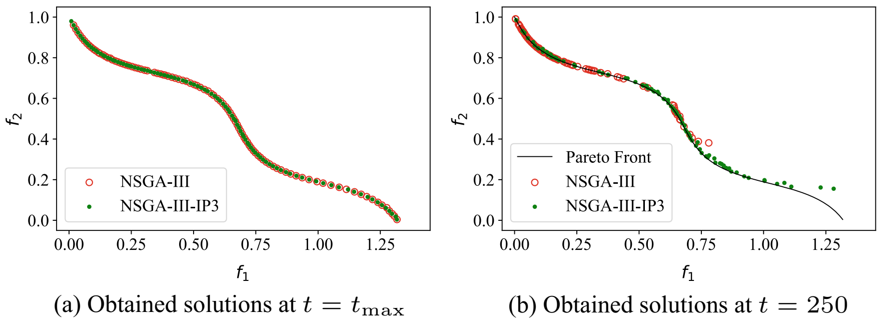


Fig. 6.10 Solutions obtained in the respective median runs of NSGA-III and NSGA-III-IP3, at termination and at an arbitrarily fixed generation, in MW12 problem

performance of NSGA-III-IP3 in terms of diversity. This improvement sped up the *PF* approximation, which could only be attributed to the search efficacy infused by the IP3 operator into NSGA-III.

6.5.2.3 DASCMOP9 Problem ($M = 3$)

The efficacy of the IP3 operator has been demonstrated above on two two-objective problems, namely CIBN1 and MW12. Widening the scope of this discussion, a three-objective problem, namely DASCMOP9, has been discussed here. In that, Fig. 6.11a and b shows the generation-wise median hypervolume and $g(X)$ plots, respectively, among the 31 randomly seeded runs of NSGA-III and NSGA-III-IP3. For NSGA-III, $t_{\max} = 1539$ had been used, as determined on-the-fly for NSGA-III-IP3. Clearly: (a) in terms of hypervolume, NSGA-III-IP3 performs significantly better than NSGA-III, and (b) in terms of $g(X)$, NSGA-III-IP3 performs worse in the intermediate generations, but performed (statistically) similarly at termination.

Further, the final obtained solutions in the respective median runs of NSGA-III and NSGA-III-IP3 on DASCMOP9 problem are shown in Fig. 6.12. As evident,

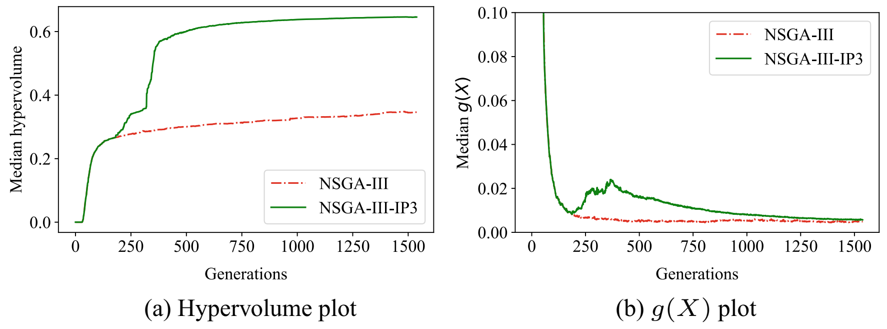
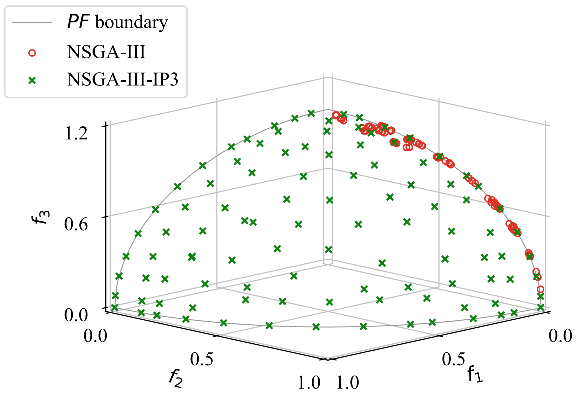


Fig. 6.11 Generation-wise performance of NSGA-III and NSGA-III-IP3 on DASCMOP9

Fig. 6.12 Final obtained solutions in the respective median runs of NSGA-III and NSGA-III-IP3 on DASCMOP9 problem



NSGA-III failed to achieve a reasonable diversity across the *PF*, while NSGA-III-IP3 achieved a diverse set of solutions across the *PF*, providing a proof-of-concept that the IP3 operator is suitable for diversity enhancement in more than two objectives as well.

6.5.3 Operator Sensitivity to Number of Nearest Neighbors

As discussed in Sect. 6.1.2, setting the number of nearest neighbors k (in kNN) is important since keeping it very low or very high may lead to underfitting or overfitting, respectively. Although the choice of $k = n$ has been reasoned, it is imperative to analyze the sensitivity of the IP3 operator's performance toward the variation in k. In this background, the performance of NSGA-III-IP3 is presented here with three different settings of k, on all test problems considered. These include (a) $k = 0.5n$, leading to a value lower than the recommended setting; (b) $k = n$, the recommended setting; and (c) $k = 1.5n$, leading to a value higher than the recommended setting.

Table 6.3 Hypervolume-based comparison of NSGA-III-IP3, across different settings of k (used in the ML method). Here, $t_{\max}$ determined on-the-fly for NSGA-III-IP3 with $k = n$ has been used for other values of k. The best performing algorithm and the statistically equivalent algorithms are marked in bold

	Problem	$t_{\max}$	$k = 0.5n$	$k = n$	$k = 1.5n$
$M = 2$	CIBN1	1404	0.461365	**0.483381**	**0.480270**
	CIBN2	753	0.668817	**0.669094**	**0.670272**
	CIBN3	889	0.218083	**0.219149**	**0.219981**
	DASCMOP1	1948	0.092495	**0.316990**	**0.310258**
	DASCMOP2	1793	0.423645	**0.637702**	**0.643858**
	DASCMOP3	1639	**0.391166**	**0.394160**	**0.421778**
	DASCMOP4	1978	**0.336946**	**0.336812**	**0.336798**
	DASCMOP5	2101	**0.672619**	**0.672677**	**0.672666**
	DASCMOP6	2377	**0.574846**	**0.574818**	**0.571492**
	MW1	1047	**0.415397**	**0.415318**	**0.415296**
	MW2	836	**0.483818**	**0.482899**	**0.482363**
	MW3	875	**0.470024**	**0.469803**	**0.454545**
	MW5	1821	**0.196080**	**0.197173**	**0.190051**
	MW6	1229	**0.298365**	**0.298348**	**0.287468**
	MW7	892	**0.366522**	**0.366413**	**0.366388**
	MW9	1071	**0.295749**	**0.296410**	**0.295482**
	MW10	1063	**0.247155**	**0.247365**	**0.199358**
	MW11	961	**0.266061**	**0.259526**	**0.243113**
	MW12	1068	**0.570748**	**0.570814**	**0.570793**
	MW13	972	**0.328788**	**0.328191**	**0.326313**
$M = 3$	CIBN4	438	**0.921200**	**0.917063**	**0.926683**
	CIBN5	287	**0.629971**	**0.629746**	**0.629372**
	DASCMOP7	1693	**1.022840**	**1.025306**	**1.016004**
	DASCMOP8	1650	**0.658195**	**0.658097**	**0.649069**
	DASCMOP9	1539	**0.647740**	**0.647012**	**0.646312**
	MW4	743	**1.041465**	**1.041362**	**1.041295**
	MW8	718	**0.626492**	**0.626362**	**0.619107**
	MW14	914	0.151894	**0.158839**	0.153753
Total (out of 28) $\longrightarrow$			22	**28**	27

The median hypervolume obtained by NSGA-III-IP3 with all three settings of k, at the end of $t_{\max}$ generations determined on-the-fly for $k = n$, is shown in Table 6.3. In that, the best obtained hypervolume, and its statistically equivalent results are marked in bold. From Table 6.3, the following may be observed.

- With $k = n$ (recommended), the performance was either statistically better than or equivalent to other settings of k in all (28 out of 28) instances. As is evident, the recommended setting of k performed well, compared to other settings.

- With $k = 0.5n$, the performance was statistically better than or equivalent to other settings of k in only 22 out of 28 instances. This deterioration of performance could be attributed to the lower value of k than desired, implying that the performance of the IP3 operator is sensitive to variation in k, for $k < n$.
- With $k = 1.5n$, the performance was statistically better than or equivalent to other settings of k in 27 out of 28 instances. Despite this slight deterioration in the overall performance, it is fair to infer that the IP3 operator's performance is not very sensitive toward variation in k, for $k > n$.

This endorses the use of $k = n$ for the kNN method in the IP3 operator. Notably, only a limited variation in k has been investigated here, owing to the scope of this book that focuses on providing the proof-of-concept that ML methods could be used for such performance enhancements in RV-EMâOAs, rather than tuning the parameters of these ML methods.

6.6 Summary

In this chapter, the IP3 operator for diversity enhancement of RV-EMâOAs has been presented. The IP3 operator is constituted by three modules, including *training-dataset construction*, *ML training*, and *offspring creation*. The first module *maps* the intra-generational solutions *across* the RVs (generated in the *objective space*) and utilizes their underlying *variable vectors* to construct M training-dataset. The second module trains M ML models on the above dataset (one per dataset) to *learn* the underlying directional improvements in *variable space*, where each ML model facilitates improvement in a particular objective. The third module utilizes these trained models, one at a time, for creation of 50% offspring ($\lfloor 0.5N \rfloor$ in number, where N is the population size). In that, half of these offspring ($\lfloor 0.25N \rfloor$ in number) are created for improvement in spread, and the other half ($\lfloor 0.25N \rfloor$ in number) for improvement in uniformity of solutions. Notably, in generations where IP3 is invoked, since 50% offspring are created already using IP3, only 50% natural offspring are created, leading to creation of N offspring in total. The hallmark of the IP3 operator is that its design and usage is guided by the overarching criteria to—avoid additional solution evaluations beyond those required by the base RV-EMâOA; favorably manage the convergence-diversity balance and ML-based risk-reward trade-off; and minimize the ad hoc parameter fixes. Experimental results on several diversity-hard problems reveal that compared to the base NSGA-III, NSGA-III-IP3 performed better in about 46% instances and better or equivalent in about 96% instances.

References

1. Abdi, H.: Bonferroni and Sidak corrections for multiple comparisons. Encyclopedia of Measurement and Statistics, pp. 103–107 (2007)
2. Bhasin, D., Swami, S., Sharma, S., Sah, S., Saxena, D.K., Deb, K.: Investigating innovized progress operators with different machine learning methods. In: Emmerich, M., Deutz, A., Wang, H., Kononova, A.V., Naujoks, B., Li, K., Miettinen, K., Yevseyeva, I. (eds.) Evolutionary Multi-Criterion Optimization, pp. 134–146. Springer Nature Switzerland, Cham (2023)
3. Cover, T.: Estimation by the nearest neighbor rule. IEEE Trans. Inf. Theory **14**(1), 50–55 (1968). https://doi.org/10.1109/TIT.1968.1054098
4. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
5. Fan, Z., Li, W., Cai, X., Li, H., Wei, C., Zhang, Q., Deb, K., Goodman, E.: Difficulty adjustable and scalable constrained multiobjective test problem toolkit. Evol. Comput. **28**(3), 339–378 (2020). https://doi.org/10.1162/evco_a_00259
6. Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. J. Amer. Stat. Assoc. **47**(260), 583–621 (1952). http://www.jstor.org/stable/2280779
7. Lin, J., Liu, H., Peng, C.: The effect of feasible region on imbalanced problem in constrained multi-objective optimization. In: 2017 13th International Conference on Computational Intelligence and Security (CIS), pp. 82–86 (2017)
8. Ma, H., Wei, H., Tian, Y., Cheng, R., Zhang, X.: A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints. Inf. Sci. **560**, 68–91 (2021). https://doi.org/10.1016/j.ins.2021.01.029
9. Ma, Z., Wang, Y.: Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons. IEEE Trans. Evol. Comput. **23**(6), 972–986 (2019). https://doi.org/10.1109/TEVC.2019.2896967
10. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: A unified innovized progress operator for performance enhancement in evolutionary multi- and many-objective optimization. IEEE Trans. Evol. Comput. 1–1 (2023). https://doi.org/10.1109/TEVC.2023.3321603
11. Padhye, N., Deb, K., Mittal, P.: Boundary handling approaches in particle swarm optimization. In: Bansal, J., Singh, P., Deep, K., Pant, M., Nagar, A. (eds.) Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012). Advances in Intelligent Systems and Computing, vol. 201, pp. 287–298. Springer, India (2013)
12. Saxena, D.K., Kapoor, S.: On timing the nadir-point estimation and/or termination of reference-based multi- and many-objective evolutionary algorithms. In: Deb, K., Goodman, E., Coello Coello, C.A., Klamroth, K., Miettinen, K., Mostaghim, S., Reed, P. (eds.) Evolutionary Multi-Criterion Optimization, pp. 191–202. Springer International Publishing, Cham (2019)
13. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945). http://www.jstor.org/stable/3001968

Chapter 7
Learning to Simultaneously Converge and Diversify Better: UIP Operator

It has been highlighted earlier that all evolutionary multi- and many-objective optimization algorithms (EMâOAs), including the reference vector (RV)-based EMâOAs or RV-EMâOAs, pursue the dual goals of convergence-to and diversity-across the true Pareto front (PF). In previous chapters, IP2 and IP3 operators have been discussed with a focus solely on convergence enhancement and diversity enhancement, respectively. Interestingly, on convergence-hard problems: NSGA-III integrated with IP2, referred to as NSGA-III-IP2, reported (statistically) significantly better convergence over base NSGA-III, without compromising on diversity. Similarly, on diversity-hard problems: NSGA-III integrated with IP3, referred to as NSGA-III-IP3, reported significantly better diversity over base NSGA-III, without compromising on convergence. Such revelations testify that the IP2 and IP3 operators could successfully maintain the delicate convergence-diversity balance, which is essential for a good PF-approximation. Plausibly, the above was possible due to the implementation of IP2 and IP3 operators in a manner which ensures that the share of convergence–diversity neutral natural[1] offspring dominates the share of pro-convergence or pro-diversity offspring, across all generations of an NSGA-III-IP2 or NSGA-III-IP3 run.

It is critically important to recognize that a priori characterization of a given problem as convergence-hard or diversity-hard, is not a trivial task. Given this, the suitability of the IP2 or IP3 operators cannot be assessed a priori. Hence, it becomes rather compelling to develop an operator that provides the scope for improvement in both convergence and diversity, without assuming a priori, the characteristics of a given problem. To this effect, this chapter presents the *Unified Innovized Progress (UIP)* operator, which invokes both the IP2 and IP3 operators, for the creation of pro-

[1] Natural offspring refer to the offspring initially created using the natural variation operators, including, recombination and mutation. Such offspring are said to be convergence–diversity neutral, since they are not created with an explicit aim to promote either convergence or diversity.

D. K. Saxena et al., *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization*, Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-2096-9_7

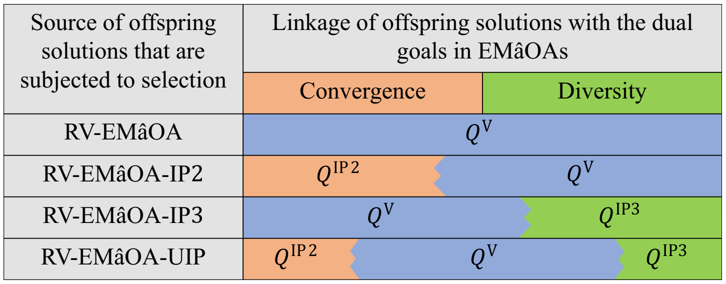


Fig. 7.1 Symbolic depiction of the convergence–diversity balance across all generations of RV-EMâOA-UIP vis-à-vis RV-EMâOA-IP3, RV-EMâOA-IP2 and RV-EMâO. The natural offspring Q^{V} do not impose any explicit preference for convergence or diversity, hence, are marked by a different color

convergence offspring Q^{IP2} and pro-diversity offspring Q^{IP3}, respectively. Aligned with the earlier used terminology, an RV-EMâOA integrated with the UIP operator is referred to as RV-EMâOA-UIP. Although the details are shared later in Sect. 7.1.4, it must be noted upfront that the share of convergence–diversity neutral offspring (Q^{V}) over all the generations of RV-EMâOA-UIP, dominates the combined share of pro-convergence (Q^{IP2}) and pro-diversity (Q^{IP3}) offspring. The same has been symbolically depicted in Fig. 7.1.

The remaining chapter is organized as follows: the UIP operator is detailed in Sect. 7.1, along with its integration with some RV-EMâOAs, including NSGA-III [6], θ-DEA [19], MOEA/DD [12] and LHFiD [17]. Its computational complexity is highlighted in Sect. 7.2, followed by its comparison with some common enhancements used in EMâOAs in Sect. 7.3. The experimental settings are discussed in Sect. 7.4, followed by the results in Sects. 7.5 and 7.6. Finally, a small analysis of the additional run-time associated with the UIP operator is presented in Sect. 7.7.

7.1 UIP Operator for Convergence and Diversity Enhancement

It has been highlighted above that the UIP operator relies on invocations of both IP2 and IP3 operators in such a manner that the delicate convergence–diversity balance is not disrupted. The constitutive modules of the IP2 operator were presented in Chap. 5, and their integration with NSGA-III was summarized in Algorithm 5.5. Similarly, in Chap. 6, the constitutive modules of the IP3 operator were presented, and their integration with NSGA-III was summarized in Algorithm 6.5. To avoid clutter in the algorithmic description of NSGA-III-UIP, the IP2 and IP3 operators have been defined below as—compact, yet self-sufficient *functions*, which can be appropriately invoked, as part of NSGA-III-UIP.

7.1.1 Representation of IP2 Operator as a Function

The IP2 operator, as a *function*, is presented in Algorithm 7.1. It includes: (a) construction of the training-dataset D_t using Algorithm 5.2, (b) training of the machine learning (ML) model on D_t using Algorithm 5.3, (c) creation of 50% natural offspring, denoted by Q_t^{V}, and (d) progression of all offspring solutions in Q_t^{V} using Algorithm 7.4, leading to 50% progressed offspring solutions Q_t^{IP2}.

Algorithm 7.1: `IP2`$(A_t, T_t, \mathcal{R}, [X^{(L)}, X^{(U)}], P_t)$

```
Input: Input archive $A_t$, target archive $T_t$, RV set $\mathcal{R}$, original variable bounds $[X^{(L)}, X^{(U)}]$,
       parent population $P_t$
Output: Offspring solutions $Q_t^{\mathrm{IP2}}$
1 $D_t \leftarrow$ Construct the training-dataset using Algorithm 5.2
2 Train the ML model using Algorithm 5.3
3 $Q_t^{\mathrm{V}} \leftarrow$ Create 50% offspring using natural variation operators
4 $Q_t^{\mathrm{IP2}} \leftarrow$ Progression of randomly picked 50% of $Q_t^{\mathrm{V}}$ using Algorithm 7.4
```

7.1.2 Representation of IP3 Operator as a Function

The IP3 operator, as a *function*, is presented in Algorithm 7.2. It includes: (a) construction of M training-dataset $\mathcal{D}_1$–$\mathcal{D}_M$ using Algorithm 6.1, (b) training of M ML models on $\mathcal{D}_1$–$\mathcal{D}_M$ (one per dataset) using Algorithm 6.2, (c) creation of 25% offspring, Q_t^B, using Algorithm 6.3 for better spread of solutions, (d) creation of 25% offspring, Q_t^G, using Algorithm 6.4 for better uniformity of solutions, and (e) merging of Q_t^B and Q_t^G, leading to 50% offspring solutions, denoted by Q_t^{IP3}.

Algorithm 7.2: `IP3`$(P_t, \mathcal{R}, r, [X^{(L)}, X^{(U)}])$

```
Input: Parent population $P_t$, RV set $\mathcal{R}$, neighborhood radius $r$, original variable bounds
       $[X^{(L)}, X^{(U)}]$
Output: Offspring solutions $Q_t^{\mathrm{IP3}}$
1 $\mathcal{D}_1$–$\mathcal{D}_M \leftarrow$ Construct the training-datasets using Algorithm 6.1
2 Train the $m$ ML models using Algorithm 6.2
3 $Q_t^B \leftarrow$ Create 25% offspring using boundary progression (Algorithm 6.3)
4 $Q_t^G \leftarrow$ Create 25% offspring using gap progression (Algorithm 6.4)
5 $Q_t^{\mathrm{IP3}} \leftarrow$ Merge $Q_t^B$ and $Q_t^G$ (total 50% offspring)
```

7.1.3 UIP Operator and Integration with NSGA-III

The UIP operator is not an independent operator which requires a goal-driven training-dataset construction and ML training, leading to offspring creation. Instead, the UIP operator is about invocation of the IP2 and IP3 operators, simultaneously or in isolation of each other, leading to creation of pro-convergence and / or pro-diversity. For the sake of clarity, this subsection highlights the steps that enable the generation of an RV-EMâOA-UIP, as statements of fact (the *what* aspect). The rationale for the above steps is detailed in the subsequent subsections (the *why* and *how* aspects).

Notably, in any generation t of an RV-EMâOA-UIP, none, one, or both of the IP2 and IP3 operators may be invoked. The first invocation of these constitutive operators is subjected to specific initiation conditions being met (as described in Chaps. 5 and 6). Furthermore, their subsequent invocations are guided by the adaptive frequency parameters ($t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$), which represent the number of generations between two successive invocations for each operator. Four different scenarios are possible, depending on which of the initiation conditions are met and the values of the corresponding frequency parameters. First, the most comprehensive scenario where both IP2 and IP3 are invoked in an RV-EMâOA-UIP generation, is discussed below, with reference to Fig. 7.2. In that scenario:

- First 100% (N) natural offspring are created, denoted by Q^{V} (in boxes (a) and (b) under 'Offspring Population').
- IP2 creates 50% pro-convergence offspring Q^{IP2} (in box (c)), through the progression of 50% of randomly chosen Q^{V}.
- IP3 creates 50% pro-diversity offspring Q^{IP3} (in box (d)), through the progression of 50% of the judiciously chosen parent solutions.

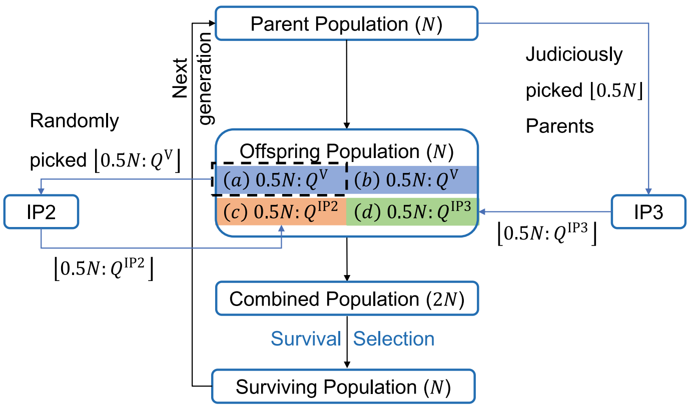

Parent Population (N)
Next generation
Judiciously picked $\lfloor 0.5N \rfloor$ Parents
Randomly picked $\lfloor 0.5N: Q^{\text{V}} \rfloor$
Offspring Population (N)
(a) $0.5N: Q^{\text{V}}$
(b) $0.5N: Q^{\text{V}}$
(c) $0.5N: Q^{\text{IP2}}$
(d) $0.5N: Q^{\text{IP3}}$
IP2
IP3
$\lfloor 0.5N: Q^{\text{IP3}} \rfloor$
$\lfloor 0.5N: Q^{\text{IP2}} \rfloor$
Combined Population ($2N$)
Survival Selection
Surviving Population (N)

Fig. 7.2 A schematic for an RV-EMâOA-UIP generation. When IP2 is invoked, $0.5N$ pro-convergence offspring Q^{IP2} are created by subjecting $0.5N$ natural offspring Q^{V} to the IP2 operator. Similarly, when IP3 is invoked, $0.5N$ pro-diversity offspring Q^{IP3} are created by subjecting $0.5N$ parents to the IP3 operator (taken from [14])

- The new offspring Q^{IP2} and Q^{IP3} (in boxes (c) and (d), collectively of size N) are combined with the parent population, leading to the combined population, of size $2N$. Notably, the *extra* offspring, in boxes (a) and (b), are not evaluated.
- Finally, the survival selection of the underlying RV-EMâOA is executed, leading to the surviving population (of size N), which serves as the parent population for the next generation.

Figure 7.2 also helps to interpret the other scenarios, where in a generation t of an RV-EMâOA-UIP, none or one of the IP2 and IP3 operators may be invoked. For example:

- If in an RV-EMâOA-UIP generation, neither IP2 nor IP3 is invoked: this generation is implemented as the base RV-EMâOA generation. In that, Q^{V}, of size N (boxes (a) and (b)) constitute the final offspring.
- If in an RV-EMâOA-UIP generation, only IP2 gets invoked: Q^{V} of size N (boxes (a) and (b)) are created and half of them randomly chosen (box (a)) are subjected to the IP2 operator. The newly created Q^{IP2} (box (c)) along with the unused Q^{V} (box (b)) constitute the final offspring.
- If in an RV-EMâOA-UIP generation, only IP3 is invoked: only half of Q^{V} (box (a)) is created and along with the newly created Q^{IP3} (box (d)), constitute the final offspring.

The enabling steps in an RV-EMâOA-UIP generation, highlighted above, prompt several critical questions, including:

(a) Why are the proportions of Q^{IP2} and Q^{IP3}, kept at 50% each?
(b) If Q^{IP2} is created by subjecting Q^{V} to the IP2 operator, then why is Q^{IP3} created by subjecting the parent solutions to the IP3 operator?
(c) What is the rationale for the criterion for the first invocation of IP2 and IP3 operators?
(d) While the subsequent invocations of IP2 and IP3 operators are based on $t^{\text{IP2}}_{\text{freq}}$ and $t^{\text{IP3}}_{\text{freq}}$, respectively, what is the criterion for their on-the-fly updates?

The answers to the above questions are detailed in the following subsections, and as depicted in Fig. 7.3 question-wise, they are all guided by the need to ensure that: (i) neither convergence nor diversity gets over-emphasized at the cost of the other, (ii) the performance of the base RV-EMâOA is not significantly impacted in a detrimental sense, if the learning (through underlying ML models) happens to be erroneous, for some reason, (iii) no extra solution evaluations vis-à-vis the base RV-EMâOA are incurred, and (iv) ad-hoc fixation of introduced parameters in avoided, as far as possible.

7.1.4 Proportion of Offspring Created by IP2/IP3 Operators

The success of EMâOAs, including RV-EMâOAs, largely depends on a fine *exploration-exploitation* balance [5, 8]. A prerequisite for this balance is that the search

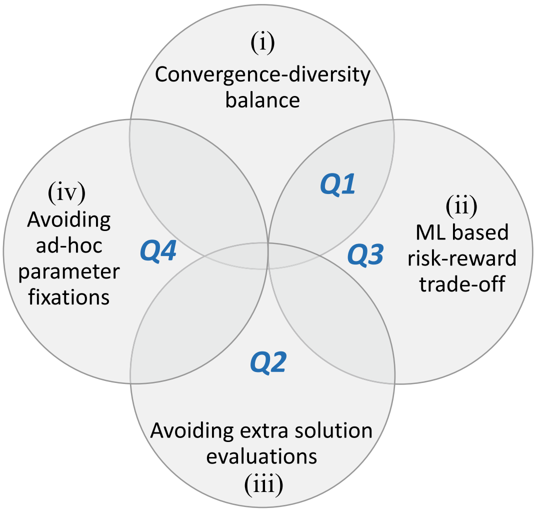

(i)
Convergence-diversity balance
Q1
(ii)
ML based risk-reward trade-off
Q3
Q2
Avoiding extra solution evaluations
(iii)
(iv)
Avoiding ad-hoc parameter fixations
Q4

Fig. 7.3 UIP operator: the key considerations (i–iv) in the wake of critical questions (Q1–Q4) listed earlier (taken from [14])

space exploration ($\mathcal{X}$ space) is convergence–diversity neutral. This implies that natural *variation* operators should not have an explicit bias toward creating offspring that are particularly superior in terms of convergence or diversity. Failure to ensure this during exploration may eventually lead *selection* (exploitation) operators to undesirably propagate faster convergence at the expense of loss of diversity or vice versa. This philosophy is endorsed by Fig. 7.1, where 100% offspring in the case of an RV-EMâOA are shown to be convergence–diversity neutral (Q^{V}). It must be noted that the *choice* of $Q^{\mathrm{IP2}} = 50\%$ and $Q^{\mathrm{IP3}} = 50\%$, along with the constraints on adaptive[2] $t^{\mathrm{IP2}}_{\mathrm{freq}}$ and $t^{\mathrm{IP3}}_{\mathrm{freq}}$, respectively, help to fulfill the key considerations of the convergence–diversity balance and ML-based risk-reward trade-off. For example:

- In a generation of RV-EMâOA-UIP where only IP2 is invoked: $Q^{\mathrm{IP2}} = 50\%$ implies its proportion equal to Q^{V}. In such a scenario, the dominant share of Q^{V} is ensured, as in the case of RV-EMâOA-IP2, as explained earlier in Sect. 5.3 (Chap. 5).
- In a generation of RV-EMâOA-UIP where only IP3 is invoked: $Q^{\mathrm{IP3}} = 50\%$ implies its proportion equal to Q^{V}. In such a scenario, the dominant share of Q^{V} is ensured, as in the case of RV-EMâOA-IP3 (by extending the same argument presented for RV-EMâOA-IP2).
- In a generation of RV-EMâOA-UIP where both IP2 and IP3 operators are invoked: $Q^{\mathrm{IP2}} = 50\%$ and $Q^{\mathrm{IP3}} = 50\%$ together imply zero contribution of convergence–diversity neutral Q^{V}. To avoid over-dependence on IP2- and IP3-based offspring, the constraints of $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 2$ and $t^{\mathrm{IP3}}_{\mathrm{freq}} \geq 2$ are imposed, so at least in the next generation, neither IP2 nor IP3 is invoked, and the contribution of Q^{V} shall be 100%.

[2] The on-the-fly adaptation is detailed in Sect. 7.1.6.

Furthermore, the fact that, under on-the-fly adaptation, $t_{\mathrm{freq}}^{\mathrm{IP2}} > 2$ and $t_{\mathrm{freq}}^{\mathrm{IP3}} > 2$ are possible ensures that across all generations the share of $Q^{\mathrm{V}} > (Q^{\mathrm{IP2}} \cup Q^{\mathrm{IP3}})$, as depicted in Fig. 7.1. This in turn shall ensure the retention of the convergence–diversity balance, and the desired safety net, in case the *learning* for the IP2 and IP3 operators is erroneous.

7.1.5 First Invocations of IP2 and IP3 Operators

To avoid erroneous *learning*, the first invocations of these operators are deferred until the base RV-EMâOA population attains some degree of stability considered conducive to convergence and diversity enhancements, respectively. The criteria for the same, reiterated here, have also been described in earlier Chaps. 5 and 6. For the pro-convergence IP2, it is desired that sufficient diversity be achieved first by the population so that most RVs get associated with a solution, and efforts to enhance convergence along each RV could be pursued. To this effect, the base RV-EMâOA is given sufficient *search* opportunity, and IP2 is first invoked only after the entire population becomes non-dominated. Similarly, the IP3 operator is first invoked after the RV-EMâOA population reports a mild stability [16], after which drastic variations in the spread of the boundaries or the internal distribution of solutions may be less likely.

7.1.6 Subsequent Invocations of the IP2 and IP3 Operators

To avoid *arbitrary parameter fixations*, the subsequent invocations of IP2 and IP3 are not enforced by rigidly fixed parameters, but guided by independent parameters, namely, $t_{\mathrm{freq}}^{\mathrm{IP2}}$ and $t_{\mathrm{freq}}^{\mathrm{IP3}}$, respectively, which are adapted on-the-fly as mentioned earlier. The adaptation of $t_{\mathrm{freq}}^{\mathrm{IP2}}$ and $t_{\mathrm{freq}}^{\mathrm{IP3}}$ is based on the survival rates of Q^{IP2} and Q^{IP3}, respectively. In the context of IP2, if the fraction of Q^{IP2} in generation t that survives to the next generation exceeds its counterpart in the previous generation, then $t_{\mathrm{freq}}^{\mathrm{IP2}}$ is reduced by 1, which implies a more frequent invocation of IP2. Otherwise, if the fraction of Q^{IP2} in generation t that survives to the next generation falls short of its counterpart in the previous generation, then $t_{\mathrm{freq}}^{\mathrm{IP2}}$ is increased by 1, which implies a less frequent invocation of IP2. The same logic is applicable for the adaptation of $t_{\mathrm{freq}}^{\mathrm{IP3}}$, based on the fraction of Q^{IP3} that survives to the next generation.

7.1.7 Avoiding Additional Solution Evaluations by IP2/IP3

The designs of the IP2 and IP3 operators are such that their invocations *do not necessitate any additional solution evaluations*, in addition to those required by the base RV-EMâOA. Notably, any generation of the base RV-EMâOA requires N solution evaluations, corresponding to the N natural offspring. Notably:

- In any generation where IP2 is invoked: first 100% natural offspring are created, and then, the trained ML model is used for the progression of 50% of these offspring (randomly picked). Subsequently, these progressed offspring replace their respective original offspring, maintaining the overall offspring count as N, while ensuring that any additional solution evaluations are avoided.
- In any generation where IP3 is invoked: only 50% natural offspring are created, while the remaining 50% are pro-diversity offspring created by subjecting judiciously picked parents to the IP3 operator. Hence, only N solution evaluations are required. Notably, the reliance on the parents here (unlike the IP2 case) is natural, since the diversity requirements with respect to *gap filling* and *boundary expansion* are directly captured by the parents (current generation solutions) themselves.

Hence, any extra solution evaluations are avoided in the case of any IP2 and/or IP3 invocations.

7.1.8 Algorithmic Implementation of RV-EMâOA-UIP

Toward a generic discussion, the UIP operator has been integrated with multiple RV-EMâOAs, including, NSGA-III, θ-DEA, MOEA/DD, and LHFiD. For a smoother reading, the integration with NSGA-III has been presented here, while the integration with the others is detailed in the appendix of this chapter (Sect. 7.9).

Algorithm 7.3 highlights a representative generation t of NSGA-III-UIP. It may be recalled that the training-dataset for the IP2 operator (Chap. 5) relies on mapping the solutions of the current generation (target-archive) and the solutions from the past generations (input-archive). In contrast, the IP3 operator (Chap. 6) is based on generating the training-dataset only from the solutions in the current generation. Therefore, at tth generation, the target-archive for the IP2 operator is updated (line 1, Algorithm 7.3), while no such update is needed for IP3. Then the prerequisite conditions for the invocations of IP2 and IP3 are checked and, if fulfilled, the $startIP2$ and $startIP3$ flags are activated for the invocation of IP2 and IP3 operators, respectively (lines 2–5, Algorithm 7.3). In the subsequent generations:

- If the IP2 operator is invoked, then the trained ML model is used for the progression of 50% of the natural offspring (randomly chosen), leading to Q_t^{IP2} (lines 6–7, Algorithm 7.3).

Algorithm 7.3: Generation t of NSGA-III-UIP

```
Input: RV set $\mathcal{R}$, variable bounds $[X^{(L)}, X^{(U)}]$, parent population $P_t$
       IP2-specific: target archive $T_{t-1}$, input archive $A_t$, frequency $t_{\text{freq}}^{\text{IP2}}$
       IP3-specific: neighborhood radius $r$, frequency $t_{\text{freq}}^{\text{IP3}}$
Output: $P_{t+1}$, $T_t$, $A_{t+1}$, $t_{\text{freq}}^{\text{IP2}}$, $t_{\text{freq}}^{\text{IP3}}$
1  $T_t \leftarrow$ Update the target archive for IP2 operator
2  if population is completely non-dominated then
3      $startIP2 = True$
4  if population has mildly stabilized then
5      $startIP3 = True$
6  if $startIP2$ & $t_{\text{freq}}^{\text{IP2}}$ generations passed after last invocation then
7      $Q_t^{\text{IP2}} \leftarrow$ Create 50% offspring using IP2 (Algorithm 7.1)
8  if $startIP3$ & $t_{\text{freq}}^{\text{IP3}}$ generations passed after last invocation then
9      $Q_t^{\text{IP3}} \leftarrow$ Create 50% offspring using IP3 (Algorithm 7.2)
10 if offspring created are not sufficient ($< N$) then
11     $Q_t^{\text{V}} \leftarrow$ Create rest of offspring using natural variation operators
12 $Q_t \leftarrow$ Merge $Q_t^{\text{IP2}}$, $Q_t^{\text{IP3}}$ and $Q_t^{\text{V}}$ (total $N$ offspring)
13 Evaluate($Q_t$)
14 $A_{t+1} \leftarrow$ Update the input archive for IP2 operator
15 $P_{t+1} \leftarrow$ Perform survival selection on $P_t \cup Q_t$
16 if IP2 was invoked in current generation then
17     Update $t_{\text{freq}}^{\text{IP2}}$ (Section 7.1.6)
18 if IP3 was invoked in current generation then
19     Update $t_{\text{freq}}^{\text{IP3}}$ (Section 7.1.6)
20 if both IP2 and IP3 were invoked in current generation then
21     if $t_{\text{freq}}^{\text{IP2}} < 2$ then $t_{\text{freq}}^{\text{IP2}} = 2$
22     if $t_{\text{freq}}^{\text{IP3}} < 2$ then $t_{\text{freq}}^{\text{IP3}} = 2$
```

- If the IP3 operator is invoked, then the trained ML model is used for the progression of 50% of the parent solutions (judiciously chosen), leading to Q_t^{IP3} (lines 8–9, Algorithm 7.3).
- If none or only one of IP2 and IP3 is invoked, the number of offspring shall be less than N. Hence, the remaining offspring are created using the variation operators, denoted by Q_t^{V} (lines 10–11, Algorithm 7.3).
- All offspring, namely Q_t^{IP2}, Q_t^{IP3} and Q_t^{V} are merged into Q_t (size N) and evaluated (lines 12–13, Algorithm 7.3).
- Update the input-archive A_{t+1} as required by the IP2 function (line 14, Algorithm 7.3).
- The combined P_t and Q_t are subjected to survival selection of NSGA-III, leading to the new P_{t+1} (line 15, Algorithm 7.3).
- $t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$ are adapted, if the respective operators were invoked in the current generation t (lines 16–19, Algorithm 7.3). This adaptation is based on the sur-

vival of the respective offspring vis-à-vis the natural offspring, as described in Sect. 7.1.6.

- Finally, if both IP2 and IP3 are invoked in the current generation, it is ensured that neither $t_{\text{freq}}^{\text{IP2}}$ nor $t_{\text{freq}}^{\text{IP3}}$ has a value lower than two (lines 20–22, Algorithm 7.3), as explained earlier in Sect. 7.1.4.

7.2 Computational Complexity of UIP Operator

As detailed in Sect. 7.1, the UIP operator consists of two modules. These modules are based on the IP2 and IP3 operators, which have computational complexity as presented in Sect. 5.4 (Chap. 5) and Sect. 6.3 (Chap. 6), respectively. From this analysis, the worst-case time and space complexities of IP2-based offspring progression and IP3-based offspring creation are summarized in Table 7.1.

It may be noted that these worst-case complexities, both for IP2 and IP3, correspond to their underlying ML methods, i.e., RF and *k*NN, respectively. The use of a different ML method may affect the corresponding complexities in Table 7.1. However, since the focus of this book is to provide a proof-of-concept that ML methods could be utilized for such enhancements related to convergence and diversity in RV-EMâOAs, the comparison of alternative ML methods is not within the scope of this book.

7.3 Comparison with Existing EMâO Enhancements

The concept of the UIP operator (and the underlying IP2 and IP3 operators), as in this book, may seem similar to certain existing enhancements used in the EMâO domain. In this section, some of these practices, including: (a) a local-search method and (b) a surrogate-modeling method, are highlighted, and their key differences from the operators introduced here are discussed. In this discussion, the IP2, IP3, and UIP operators are collectively referred to as the IP operators.

Table 7.1 Time- and space complexities of different modules of the UIP operator

Module	Time-complexity	Space-complexity
IP2-based offspring progression	$\mathcal{O}(N^3 t_{\text{past}}^3 n \log(N t_{\text{past}}))$	$\mathcal{O}(N^2 t_{\text{past}}^2 n)$
IP3-based offspring creation	$\mathcal{O}(MNn \log(N))$	$\mathcal{O}(MNn)$

7.3.1 A Local Search Method

The IP operators (IP2, IP3, and UIP) attempt to create the offspring solutions through their ML-based progression, in any intermediate generation of an RV-EMâOA run. This operation should not be confused with a local search method [2, 10, 15] that aims to improve the local convergence of solutions in any generation. Key differences are highlighted below.

- Usually, in a multi- or many-objective optimization problem (MOP/MaOP), multiple conflicting objectives are present. This poses a challenge to local search, as the local search usually relies on a single objective function for improvement, defining which may be a non-trivial task. On the contrary, multiple objectives do not pose any additional challenge to any of the IP operators.
- Implementing a local search requires additional solution evaluations beyond the default solution evaluations of any EMâOA, which is not the case with any of the IP operators.
- A local search usually means searching for the best solution in a local neighborhood, while the progression of any solution's X vector using any of the IP operators may be substantial and not necessarily be local.

7.3.2 A Surrogate-Modeling Method

In EMâOAs, a surrogate model is often constructed and used to find solutions closer to PF. In real-world problems in which solution evaluations are computationally very expensive, such an approach helps reduce the number of actual solution evaluations needed to converge, thus saving computational effort and run-time (Chap. 3). Existing studies on surrogate modeling have used several ML methods, including Random Forests (RF). Therefore, it is important to clarify the distinction between ML-based surrogate modeling and ML-based IP operators, which Table 7.2 seeks to make clearer.

7.4 Experimental Setup

This section lays the foundation for the experimental investigations by highlighting the: (a) test suite considered, (b) performance indicators used and related statistical analysis, and (c) parameters pertaining to the RV-EMâOA(s) and the UIP operator.

Table 7.2 Fundamental differences between an EMâOA coupled with surrogate modeling and with one of the IP operators (IP2, IP3, or UIP)

Point of difference	EMaOAs with	
	Surrogate modeling	IP operators
1. ML model training	X-F mapping	X-X mapping
2. ML model application	Does not alter a solution's X vector directly	Alters a solution's X vector directly
3. Evolution of Offspring	Performed by variation operators only	Performed by the variation operators and/or the learnt ML model(s)
4. Fitness evaluation	Guided by both approximate and actual objective values	Guided by actual objective values only
5. Quality of PF-approximation	At best equivalent to base EMÔA but in fewer actual solution evaluations	Better of equivalent PF-approximation in same number of solution evaluations

7.4.1 Test Suite

To demonstrate the efficacy of the UIP operator, several MOPs have been used, as previously used in Chaps. 5 and 6. These include: (i) convergence-hard ($\tilde{Z}$DT, DTLZ and MaF); and (ii) diversity-hard (CIBN, DASCMOP, and MW) problems.

Once the efficacy of the UIP operator is established on MOPs, several MaOPs with $M = 5$, 8, and 10 are also considered, including the DTLZ [7] and MaF [3] problems. In these problems, the distance variables have been set to $k = 20$, wherever applicable.

7.4.2 Performance Indicators and Statistical Analysis

The choice of performance indicators is the same as that adopted earlier in Chaps. 5 and 6. Key details are reiterated in the following:

- Hypervolume is used as the primary indicator, with the reference point set as $R_{1\times M} = [1 + \frac{1}{p}, \ldots, 1 + \frac{1}{p}]$, where p is the number of gaps set for the Das–Dennis method while generating the RVs for RV-EMâO. Furthermore, for problems where the scales of different objectives are different, the solutions are normalized in the $\mathcal{Z}$ space using the theoretical PF extremes.
- The population mean of the function $g(X)$ is used as a secondary indicator to provide information on the convergence levels in the $\mathcal{X}$ space.

Furthermore, in the context of statistical analysis of performance indicator values,

- When comparing *only two* algorithms at the same time, the Wilcoxon rank sum test [18] is performed on the indicator values reported on multiple independently seeded runs. Here, the threshold value of $p = 0.05$ (95% confidence interval) is used.

- When comparing *more than two* algorithms at the same time, the Kruskal–Wallis test [11] with the threshold value $p = 0.05$ is used to infer whether their overall differences are statistically significant or not. If not, the Wilcoxon rank sum test is used for their pairwise comparisons, treating NSGA-III-UIP as the reference. In that, the threshold value p is adjusted using the standard Bonferroni correction [1], to retain the same overall confidence.

7.4.3 Parameter Settings

In this subsection, the parameters and settings used for (a) the RV-EMâOAs; and (b) the UIP operator are discussed.

7.4.3.1 RV-EMâOA Settings

To obtain a reasonably sized set of RVs using the Das–Dennis method [4], the gap parameter is set as given in Table 7.3. In that table, where two values of p (gaps) are shown, the first value is used to create the boundary RVs and the second value is used to create the interior RVs [6]. For coherence, the population size N is kept the same as the number of RVs corresponding to a particular objective, as given in Table 7.3. Furthermore, the natural variation operators include SBX crossover ($p_c = 0.9$ and $\eta_c = 20$) and polynomial mutation ($p_c = 1/n$ and $\eta_m = 20$) for an n-variable problem.

Further, each EMâOA has been run 31 times, with different random seeds. In that process, for NSGA-III-UIP, termination generation $t_{\max}$ has been determined on-the-fly through the stabilization tracking algorithm, using $\psi_{\text{term}} \equiv \{3, 50\}$. For all other EMâOAs, the mean $t_{\max}$ determined for NSGA-III-UIP on 31 runs has been used as $t_{\max}$.

Table 7.3 Parameter settings for the Das–Dennis method

Setting	$M = 2$	$M = 3$	$M = 5$	$M = 8$	$M = 10$
p (gaps)	99	13	5, 4	3, 3	3, 3
N	100	105	196	240	440

7.4.3.2 UIP Operator Settings

The UIP operator does not involve any additional parameters, other than those specific to IP2 or IP3, for which details are provided below.

- IP2-specific: the IP2 operator involves three parameters—t_{past}, $t_{\text{freq}}^{\text{IP2}}$ and η. Here: (a) $t_{\text{past}} = 5$ is used, since performance is not very sensitive to t_{past} [13], (b) $t_{\text{freq}}^{\text{IP2}}$ is adapted on-the-fly based on the survival of IP2-based offspring, while its initial value is set as 1, and (c) η assumes a random value in the range $[1, 1.5]$.
- IP3-specific: the IP3 operator involves two parameters—$t_{\text{freq}}^{\text{IP3}}$ and ψ_{mild}. In that operator, $t_{\text{freq}}^{\text{IP3}}$ is adapted on-the-fly based on the survival of IP3-based offspring, while its initial value is set as 1. In addition, ψ_{mild} governs the degree of stabilization required to trigger IP3's first invocation. While it is intuitive that ψ_{mild} should correspond to a lower degree of stabilization than ψ_{TM} that is used to terminate the RV-EMâOA-UIP runs, its exact setting is borrowed from the suggestion made in [16, 17]. According to that, $\psi_{\text{mild}} = \{2, 20\}$.

7.5 Results and Discussion

This section first presents the evaluation of: (a) UIP vis-à-vis IP2, on convergence-hard problems; and (b) UIP vis-à-vis IP3, on diversity-hard problems, to collectively establish the efficacy of the UIP operator on MOPs, after which the evaluation of the UIP operator on MaOPs is presented.

7.5.1 UIP Vis-à-Vis IP2 Operator

In this subsection, the performance comparison of UIP and IP2 operators has been made through a direct comparison of NSGA-III-UIP and NSGA-III-IP2 on convergence-hard MOPs. Notably, NSGA-III has also been included as reference, to assess whether UIP's integration into NSGA-III leads to deteriorated performance in any test instance.

Table 7.4 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of the $t_{\max}$ generations. In that, $t_{\max}$ has been determined on-the-fly for NSGA-III-UIP, and the same has been used for NSGA-III and NSGA-III-IP2. From Table 7.4, the following can be observed:

- In the context of hypervolume: compared to NSGA-III-IP2, NSGA-III-UIP performs statistically better, equivalent, and worse in 4, 16, and 1 instances, respectively, out of 21 instances.
- In the context of $g(X)$ values: compared to NSGA-III-IP2, NSGA-III-UIP performs statistically better, equivalent, and worse in 3, 12, and 3 instances,

Table 7.4 Hypervolume and $g(X)$ based comparison of NSGA-III-UIP with NSGA-III and NSGA-III-IP2, on convergence-hard (Z̃DT, DTLZ and MaF) problems. The termination generation ($t_{\max}$) has been determined on-the-fly for NSGA-III-UIP. The entries are formatted as $median$ indicator values from 31 runs followed by '+', '=' or '−'. $+/=/-$ inform that NSGA-III-UIP performs statistically worse than, equivalent to, or better than the underlying algorithm, respectively

Problem		$t_{\max}$	Hypervolume			$g(X)$		
			NSGA-III	NSGA-III-IP2	NSGA-III-UIP	NSGA-III	NSGA-III-IP2	NSGA-III-UIP
$M = 2$	Z̃DT1	1198	0.681860=	0.681860=	0.681859	1.0007E+00–	1.0004E+00=	1.0004E+00
	Z̃DT2	1267	0.348794=	0.348794=	0.348794	1.0005E+00–	1.0003E+00=	1.0003E+00
	Z̃DT3	1007	1.068445=	1.068408=	1.068492	1.0096E+00=	1.0061E+00=	1.0070E+00
	Z̃DT4	1767	0.681859=	0.681860=	0.681860	1.0007E+00–	1.0004E+00=	1.0002E+00
	Z̃DT6	1836	0.312752–	0.324640–	0.336501	1.4277E+00–	1.3161E+00–	1.1387E+00
$M = 3$	DTLZ1	1497	1.221639=	1.222229=	1.222575	1.9427E-02=	1.1336E-02=	6.2710E-03
	DTLZ2	978	0.667327=	0.667309=	0.667323	5.3477E-06=	4.4180E-06=	4.8873E-06
	DTLZ3	1750	0.652251–	0.656486–	0.662523	8.9193E-03–	6.3068E-03–	2.3456E-03
	DTLZ4	1449	0.667309–	0.667331=	0.667346	1.6300E-07=	5.9762E-08+	4.4408E-07
	MaF1	608	0.235973=	0.234828=	0.235578	4.6384E-04=	5.7130E-04=	1.0697E-03
	MaF2	486	0.396887=	0.396552=	0.396720	1.5266E-01=	5.5675E-02=	6.0477E-02
	MaF3	2135	1.193651=	1.193381=	1.194214	3.7947E-03=	3.8814E-03=	2.4265E-03
	MaF4	1214	0.612050–	0.621803–	0.634728	2.8012E-02–	1.5481E-02–	3.3689E-03
	MaF5	1345	1.227613=	1.227604=	1.227609	5.7435E-07=	2.5409E-06=	8.4903E-07
	MaF7	1201	0.375791=	0.376011=	0.375603	1.0000E+00=	1.0003E+00=	1.0003E+00
	MaF8	1244	0.463948–	0.464591=	0.466427	—	—	—
	MaF9	1160	0.626751=	0.626751=	0.626726	—	—	—
	MaF10	996	0.528291=	0.520969=	0.521247	1.8401E-02=	1.1232E-02+	2.0249E-02
	MaF11	993	0.980497=	0.980142=	0.980780	9.5841E-04=	1.2933E-03=	1.2323E-03
	MaF12	725	0.599802–	0.614806+	0.612866	2.3092E-01=	1.7244E-01+	1.9415E-01
	MaF13	1065	0.365532–	0.371873–	0.377158	—	—	—
Total (+/=/–) $\longrightarrow$			00/14/07	01/16/04	of 21 probs.	00/12/06	03/12/03	of 18 probs.

Note (—) implies that the concerned problem does not have a $g(X)$ function

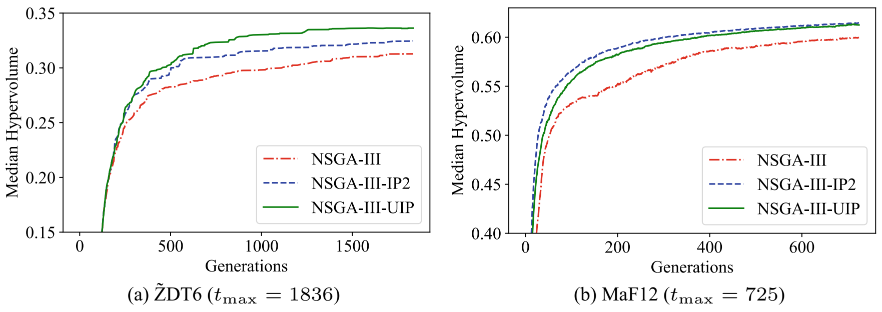


Fig. 7.4 Generation-wise median hypervolume plots from among 31 randomly seeded runs of NSGA-III, NSGA-III-IP2 and NSGA-III-UIP, on two sample convergence-hard problems

respectively, out of 18 instances (problems where $g(X)$ function is not existent, are marked by '–').

Overall, the above results endorse that on convergence-hard problems, NSGA-III-UIP performs statistically better than or equivalent to NSGA-III-IP2 in most cases, and worse in a few cases. These results could be interpreted in light of the interplay of—the problem features and the algorithmic features, as explained below:

1. *Problem features*: convergence-hard problems may belong to two broad categories: (i) where supplemental diversity may be desired or essential for good convergence—for example, problems with multiple local optima, where population diversification may help in overcoming the traps of local optima, and (ii) where supplemental diversity may not facilitate better convergence.
2. *Algorithmic features*: for convergence-hard problems, the algorithmic features contributing to the selection pressure for convergence are noteworthy. For example, compared to NSGA-III-IP2, NSGA-III-UIP potentially provides a diminished selection pressure for convergence[3] and a supplemental selection pressure for diversity. Hence, in problems where supplementary diversity may help improve convergence (as generally observed for EMâOAs), NSGA-III-UIP may potentially perform better than or equivalent to NSGA-III-IP2. However, in problems where supplemental diversity may be inconsequential, NSGA-III-UIP may potentially be marginally worse than NSGA-III-IP2.

The interplay of the problem and algorithmic features cited above appears to manifest in a contrasting manner for the $\tilde{\text{Z}}$DT6 and MaF12 problems, where NSGA-III-UIP offers a better and marginally worse hypervolume, respectively, than NSGA-III-IP2 (Fig. 7.4).

[3] In case of NSGA-III-IP2: $\lfloor 0.5N \rfloor$ pro-convergence offspring are produced whenever IP2 is invoked, with $t^{\text{IP2}}_{\text{freq}} \geq 1$. However, in case of NSGA-III-UIP: whenever IP2 and IP3 are invoked simultaneously, pro-convergence $\lfloor 0.5N \rfloor$ offspring are produced, but with $t^{\text{IP2}}_{\text{freq}} \geq 2$. The latter causes fewer invocations of pro-convergence IP2 in the case of NSGA-III-UIP than in NSGA-III-IP2, leading to a potentially diminished selection pressure for convergence in NSGA-III-UIP.

7.5.2 UIP Vis-à-Vis IP3 Operator

In this subsection, the performance comparison of UIP and IP3 operators has been realized through a direct comparison of NSGA-III-UIP and NSGA-III-IP3 on diversity-hard MOPs. Notably, NSGA-III has also been included as a reference, to assess whether UIP's integration into NSGA-III leads to deteriorated performance in any test instance.

Table 7.5 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of the $t_{\max}$ generations. In those runs, $t_{\max}$ has been determined on-the-fly for NSGA-III-UIP, and the same has been used for NSGA-III and NSGA-III-IP3. From Table 7.5, the following can be observed:

- In the context of hypervolume: compared to NSGA-III-IP3, NSGA-III-UIP performs statistically better, equivalent, and worse in 6, 21, and 1 instances, respectively, out of 28 instances.
- In the context of $g(X)$ values: compared to NSGA-III-IP3, NSGA-III-UIP performs statistically better, equivalent, and worse in 8, 20, and 0 instances.

Overall, the above results endorse that, on diversity-hard problems, NSGA-III-UIP performs statistically better than or equivalent to NSGA-III-IP3 in most cases, and worse in a few cases. These results could also be interpreted in light of the interplay of—the problem features and algorithmic features, as explained below.

1. *Problem features*: diversity-hard problems may belong to two broad categories: (i) where supplemental convergence during the earlier generations may be desired or essential for achieving good diversity, eventually. and (ii) where supplemental convergence during the earlier generations may be inconsequential for the final diversity.
2. *Algorithmic features*: for diversity-hard problems, the algorithmic features contributing to the selection pressure for diversity are noteworthy. For example, compared to NSGA-III-IP3, NSGA-III-UIP potentially provides a diminished selection pressure for diversity[4] and a supplemental selection pressure for convergence. Hence, in problems where supplemental convergence may help improve diversity, NSGA-III-UIP may potentially perform better than or equivalent to NSGA-III-IP3. However, in problems where supplemental convergence may be inconsequential, NSGA-III-UIP may potentially be marginally worse than NSGA-III-IP3.

Interestingly, the interplay of the problem and algorithmic features cited above appears to manifest itself in favor of NSGA-III-UIP than NSGA-III-IP3, in all the problems considered, with the exception of the CIBN5 problem. For a sample illustration of the broader trend, the DASCMOP1 problem has been considered. Figure 7.5a

[4] In case of NSGA-III-IP3: $\lfloor 0.5N \rfloor$ pro-diversity offspring are produced whenever IP3 is invoked, with $t_{\text{freq}}^{\text{IP3}} \geq 1$. However, in the case of NSGA-III-UIP: whenever IP2 and IP3 are invoked simultaneously, $\lfloor 0.5N \rfloor$ pro-diversity offspring are produced, but with $t_{\text{freq}}^{\text{IP3}} \geq 2$. The latter causes fewer invocations of pro-diversity IP3 in the case of NSGA-III-UIP than in NSGA-III-IP3, leading to a potentially diminished selection pressure for diversity in NSGA-III-UIP.

Table 7.5 Hypervolume and $g(X)$ based comparison of NSGA-III-UIP with NSGA-III and NSGA-III-IP3, on diversity-hard (CIBN, DASCMOP, and MW) problems. The termination generation ($t_{\max}$) has been determined on-the-fly for NSGA-III-UIP. The entries are formatted as *median* indicator values from 31 runs followed by '+', '=', or '–'. $+/=/-$ inform that NSGA-III-UIP performs statistically worse than, equivalent to, or better than the underlying algorithm, respectively

Problem		$t_{\max}$	Hypervolume			$g(X)$		
			NSGA-III	NSGA-III-IP3	NSGA-III-UIP	NSGA-III	NSGA-III-IP3	NSGA-III-UIP
$M = 2$	CIBN1	1365	0.327867–	0.482560–	0.509220	0.000594+	0.001758–	0.001178
	CIBN2	789	0.658712–	0.669122=	0.669612	0.003744–	0.002480–	0.001512
	CIBN3	960	0.213584–	0.219614–	0.227093	0.003216–	0.002899–	0.001622
	DASCMOP1	2101	0.089766–	0.320434–	0.332936	0.000258+	0.004292–	0.001781
	DASCMOP2	1944	0.414610–	0.645024–	0.667395	0.000284+	0.010297–	0.002657
	DASCMOP3	1658	0.391774–	0.396188=	0.399092	0.000530=	0.000120=	0.000125
	DASCMOP4	1993	0.336866=	0.336810=	0.336960	0.000146=	0.000139=	0.000101
	DASCMOP5	2113	0.672667=	0.672740=	0.672744	0.000137–	0.000126=	0.000100
	DASCMOP6	2432	0.549906–	0.574880=	0.574864	0.000093=	0.000074=	0.000063
	MW1	1042	0.415304=	0.415293=	0.415323	1.000066=	1.000061=	1.000058
	MW2	848	0.483094=	0.482937=	0.491429	1.020607=	1.020729=	1.013901
	MW3	868	0.469880=	0.469740=	0.469597	1.041617+	1.044934=	1.044984
	MW5	1783	0.083010–	0.196301=	0.198921	1.000027+	1.000176=	1.000246
	MW6	1244	0.298309–	0.298408–	0.317611	1.026723–	1.026687–	1.013420
	MW7	906	0.366397=	0.366483=	0.366303	1.095000–	1.096852–	1.092561
	MW9	1063	0.293673–	0.296315=	0.296237	1.452037–	1.427431=	1.428838
	MW10	1071	0.247084=	0.247373=	0.268710	1.049229=	1.050259=	1.041663
	MW11	976	0.268232=	0.264602=	0.261213	1.278576=	1.246254=	1.245009
	MW12	1055	0.570528–	0.570794=	0.570739	1.246357+	1.249184=	1.249410
	MW13	1001	0.328966–	0.328578–	0.346489	1.070329–	1.070999–	1.054297

(continued)

Table 7.5 (continued)

Problem		$t_{\max}$	Hypervolume			$g(X)$		
			NSGA-III	NSGA-III-IP3	NSGA-III-UIP	NSGA-III	NSGA-III-IP3	NSGA-III-UIP
$M = 3$	CIBN4	439	0.912600–	0.920534=	0.923339	0.012217+	0.016751=	0.015673
	CIBN5	294	0.629637+	0.629868+	0.628532	0.008626–	0.008349=	0.007897
	DASCMOP7	1731	1.027245=	1.026493=	1.025842	0.000909=	0.000977=	0.001066
	DASCMOP8	1675	0.630175=	0.659697=	0.638794	0.010419=	0.001257=	0.008113
	DASCMOP9	1537	0.346493–	0.647141=	0.647739	0.004972=	0.005465=	0.005233
	MW4	743	1.041376=	1.041260=	1.041385	1.000239=	1.000372=	1.000201
	MW8	732	0.626663=	0.626711=	0.627444	1.014399=	1.014274=	1.013526
	MW14	848	0.152752=	0.151446=	0.154971	1.018127=	1.017369=	1.019706
		Total (+/=/–) $\longrightarrow$	01/13/14	01/21/06	of 28 probs.	07/13/08	00/20/08	of 28 probs.

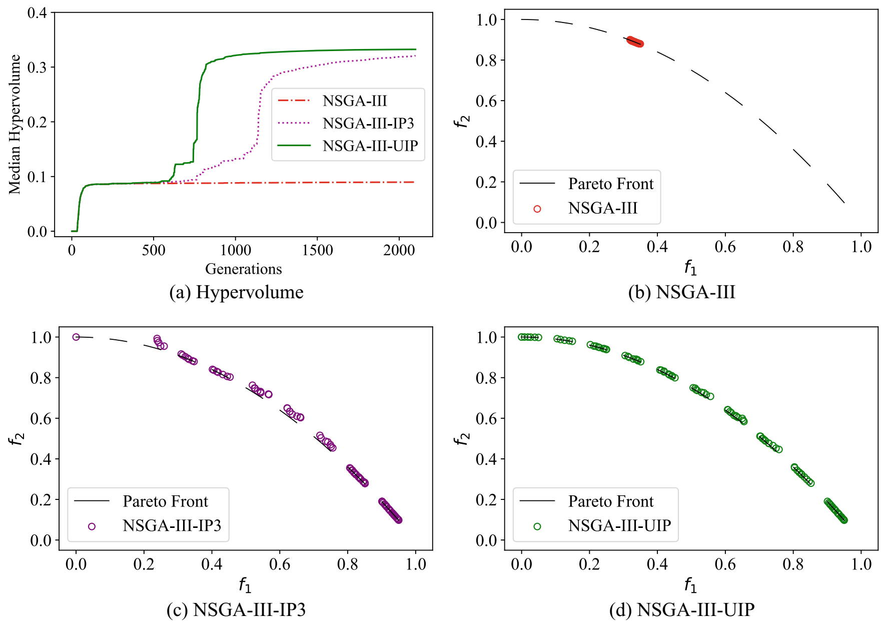


Fig. 7.5 Generation-wise hypervolume trend and solutions obtained in the respective median runs of NSGA-III, NSGA-III-IP3, and NSGA-III-UIP at $t_{\max} = 2101$ generations, on DASCMOP1

shows the generation-wise median hypervolume plot among the 31 randomly seeded runs of NSGA-III, NSGA-III-IP3, and NSGA-III-UIP. In addition, their respective PF-approximations at the end of $t_{\max} = 2101$ generations are shown in Figs. 7.5b to d. Clearly, the superiority of NSGA-III-UIP is evident in terms of both the hypervolume measure and the quality of the PF-approximation. While the above results form the broader trend, the only aberration to it related to the CIBN5 problem is captured in Fig. 7.6. While the quality of PF-approximations offered by NSGA-III, NSGA-III-IP3, and NSGA-III-UIP presented in Figs. 7.6b to d, are hard to distinguish visually, their characterization in terms of hypervolume values presented in Fig. 7.6a reveals the differences in their performance.

7.5.3 *UIP Operator on Many-Objective Problems*

This subsection aims to assess whether the efficacy of the UIP operator, established above on MOPs, can be extended to MaOPs as well. Notably, since the UIP operator clearly outperformed the IP2 and IP3 operators, they have been excluded from further investigation on MaOPs. To this end, Table 7.6 reports the median hypervolume values from among the 31 randomly seeded runs of NSGA-III and NSGA-III-UIP, at the end of the $t_{\max}$ generations. In this, $t_{\max}$ has been determined on-the-fly for

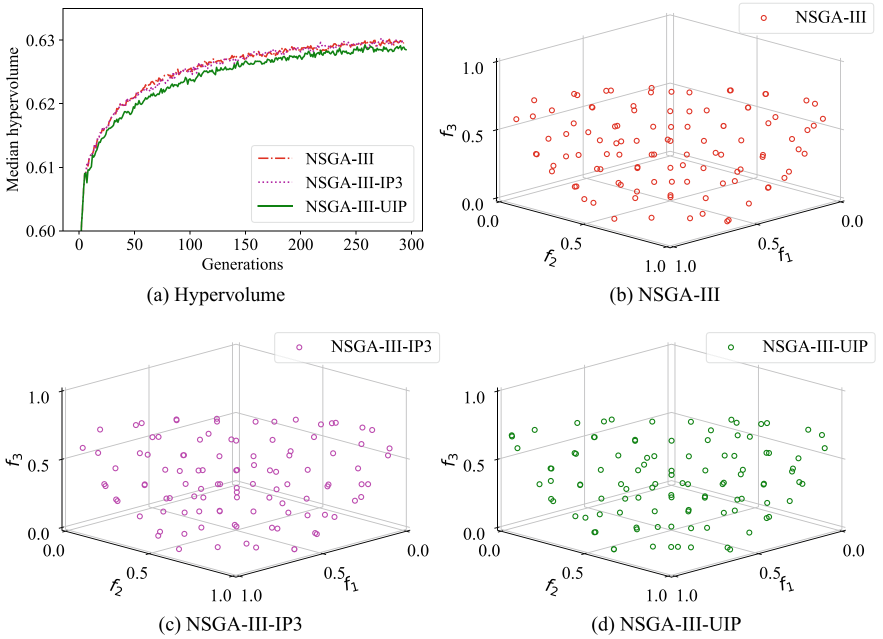


Fig. 7.6 Generation-wise hypervolume trend and solutions obtained in the respective median runs of NSGA-III, NSGA-III-IP3, and NSGA-III-UIP at $t_{\max} = 294$ generations, on CIBN5

NSGA-III-UIP, and the same has been used for NSGA-III. From Table 7.6, it can be observed that compared to NSGA-III, NSGA-III-UIP performs statistically better, equivalent, and worse in 15, 28, and 5 instances, respectively, out of 48 instances (spread over $M = 5$, 8 and 10). From the above, it is fair to infer that NSGA-III-UIP offers overall better performance than NSGA-III on the considered MaOPs, ranging from 5 to 10 objectives. This serves as a proof-of-concept that the UIP operator is scalable in terms of objectives. Hence, the UIP operator is capable of improving the performance of NSGA-III, not only on MOPs, but also on MaOPs.

7.6 Comparison with Other RV-EMâOAs

To demonstrate how the UIP operator can improve the search efficacy of other RV-EMâOAs, the performance of θ-DEA-UIP, MOEA/DD-UIP, and LHFiD-UIP has been investigated vis-à-vis their respective base variants, on both MOPs and MaOPs.

Table 7.6 Hypervolume based comparison of NSGA-III-UIP with NSGA-III on many-objective (DTLZ and MaF) problems, with $M = 5$, 8 and 10. The termination generation ($t_{\max}$) has been determined on-the-fly for NSGA-III-UIP. The entries are formatted as *median* hypervolume values from 31 runs followed by '+', '=', or '−'. $+/=/-$ inform that NSGA-III-UIP performs statistically worse than, equivalent to, or better than the underlying algorithm, respectively

Problem	$M = 5$			$M = 8$			$M = 10$		
	$t_{\max}$	NSGA-III	NSGA-III-UIP	$t_{\max}$	NSGA-III	NSGA-III-UIP	$t_{\max}$	NSGA-III	NSGA-III-UIP
DTLZ1	1367	2.486820=	2.487010	1604	9.988500=	9.988620	2007	17.757700=	17.757700
DTLZ2	860	2.172040=	2.172090	777	9.816830–	9.818580	793	17.665800–	17.667500
DTLZ3	1071	1.938460+	0.000000	1537	9.741440=	9.759030	1718	17.650300–	17.664100
DTLZ4	912	2.173240=	2.173190	825	9.826530+	9.826270	820	17.677500=	17.677200
MaF1	843	0.059222=	0.059753	989	0.015552=	0.015751	830	0.002220=	0.002241
MaF2	359	0.997132=	0.998834	422	4.352760–	4.377980	487	7.846630=	7.856730
MaF3	2238	2.488190–	2.488200	3333	9.988720–	9.988720	3765	17.757700–	17.757727
MaF4	604	0.117321=	0.083836	874	0.118441=	0.153381	1069	0.064231–	0.077138
MaF5	1109	2.487950=	2.487940	1160	9.988720–	9.988720	1196	17.757700–	17.757746
MaF7	1087	0.970398=	0.969063	912	4.402240=	4.390560	849	7.602060=	7.595800
MaF8	1546	0.475295=	0.476568	1315	0.709058+	0.686091	1214	0.577736=	0.572205
MaF9	985	0.027410=	0.027633	716	0.000518–	0.002425	774	0.000106=	0.000136
MaF10	1077	0.939567–	0.947997	853	3.240900–	3.456380	855	6.178800+	5.927050
MaF11	968	2.437800=	2.437950	1369	9.807760–	9.847950	1606	17.526500–	17.613100
MaF12	573	1.775930=	1.776530	448	7.256190=	7.254460	483	13.127900=	13.199400
MaF13	820	0.225160+	0.206183	872	0.234528=	0.224742	867	0.172316=	0.165359
Total (+/=/–) ⟶		02/12/02			02/07/07			01/09/06	of 16 probs.

7.6.1 Multi-objective Problems

In this subsection, for each of the algorithms: θ-DEA, MOEA/DD and LHFiD, its base version is compared with its respective UIP variants. The median HV values are shown in Table 7.7, where one can see:

- θ-DEA-UIP performs statistically better in 23 (out of 49) instances, and statistically better or equivalently in 48 (out of 49) instances, compared to θ-DEA.
- MOEA/DD-UIP performs statistically better in 20 (out of 49) instances, and statistically better or equivalently in 48 (out of 49) instances, compared to MOEA/DD.
- LHFiD-UIP performs statistically better in 16 (out of 49) instances, and statistically better or equivalently in 45 (out of 49) instances, compared to LHFiD.

In general, the UIP variants are statistically better in about 42% of the instances, and statistically better or equivalent in about 96% of the instances, compared to their respective base variants (including the NSGA-III-UIP results presented in Sects. 7.5.1 and 7.5.2). This performance enhancement could be directly attributed to the search efficacy infused by the UIP operator when integrated with an RV-EMâOA.

7.6.2 Many-objective Problems

In this subsection, for each of the algorithms: θ-DEA, MOEA/DD, and LHFiD, its base version is compared with its respective UIP variants. The median HV values are shown in Table 7.8, where one can see:

- θ-DEA-UIP performs statistically better in 13 (out of 48) instances, and statistically better or equivalently in 45 (out of 48) instances, compared to θ-DEA.
- MOEA/DD-UIP performs statistically better in 14 (out of 48) instances, and statistically better or equivalently in 44 (out of 48) instances, than MOEA/DD.
- LHFiD-UIP performs statistically better in 14 (out of 48) instances, and statistically better or equivalently in 38 (out of 48) instances, than LHFiD.

In general, the UIP variants are statistically better in about 28% of the instances, and statistically better or equivalent in about 89% of the instances (including NSGA-III-UIP results presented in Sect. 7.5.3). This performance enhancement could be directly attributed to the search efficacy infused by the UIP operator, when integrated with an RV-EMâOA.

7.7 Run-Time Analysis of UIP Operator

For real-world problems, quite often the time spent on solution evaluations constitutes a dominant fraction of the overall run-time of the optimization process. Moreover, these solution evaluations are costly in terms of the resources (experimental or computational solvers, other than the optimizer) needed for evaluation.

Table 7.7 HV based performance comparison of θ-DEA-UIP, MOEA/DD-UIP and LHFiD-UIP with their respective base variants, i.e., θ-DEA, MOEA/DD and LHFiD, respectively, on both convergence-hard ($\tilde{\text{Z}}$DT, DTLZ, MaF) and diversity-hard (CIBN, DASCMOP, MW) test suites. Each column shows the median HV (from 31 runs) at the end of $t_{\max}$ generations, determined on-the-fly for the respective UIP variants. The symbols '+', '=', or '–' against each variant highlight where these are statistically better than, comparable to, or worse than the corresponding UIP variant, respectively (taken from [14])

Problem	M	$t_{\max}$	θ-DEA	θ-DEA- UIP	$t_{\max}$	MOEA/DD	MOEA/DD- UIP	$t_{\max}$	LHFiD	LHFiD- UIP
$\tilde{\text{Z}}$DT1	2	1161	0.681280=	0.681299	1155	0.681215=	0.681193	683	0.681874+	0.681871
$\tilde{\text{Z}}$DT2	2	1237	0.347854=	0.347861	1247	0.347746=	0.347800	774	0.348792–	0.348793
$\tilde{\text{Z}}$DT3	2	977	1.067375=	1.067561	1005	1.067135=	1.067364	664	0.987924–	1.022636
$\tilde{\text{Z}}$DT4	2	1744	0.681257=	0.681255	1675	0.681196=	0.681192	1481	0.681873+	0.681782
$\tilde{\text{Z}}$DT6	2	1822	0.313645–	0.334052	1771	0.313384–	0.332646	1209	0.309905–	0.333011
DTLZ1	3	1118	1.213503+	0.873703	1136	1.216124=	1.202253	633	0.012268=	0.002838
DTLZ2	3	963	0.651611=	0.651758	912	0.653244=	0.652884	499	0.659976=	0.660592
DTLZ3	3	1237	0.611978=	0.085917	981	0.551322+	0.000000	635	0.000000=	0.000000
DTLZ4	3	1935	0.652918=	0.654825	1226	0.654143–	0.655937	908	0.661133=	0.661058
MaF1	3	618	0.228233=	0.229025	625	0.229522=	0.227796	546	0.208156–	0.209422
MaF2	3	515	0.390904=	0.391698	513	0.393007=	0.393700	488	0.394240=	0.394093
MaF3	3	2123	1.193223–	1.194434	2133	1.191562–	1.194840	1527	1.191826=	1.194385
MaF4	3	1300	0.610459–	0.620879	1295	0.613593–	0.617812	736	0.568126=	0.013983
MaF5	3	1831	1.228640=	1.228654	1503	1.228484=	1.228521	1119	1.222987–	1.228257
MaF7	3	1166	0.371659=	0.372502	1118	0.371731=	0.371782	1049	0.363208–	0.371374
MaF8	3	1532	0.000331=	0.000170	1608	0.000109=	0.000117	1552	0.458162=	0.458045
MaF9	3	1155	0.610583–	0.614858	1227	0.610000–	0.613124	777	1.160416=	1.160459
MaF10	3	1019	0.477226–	0.492448	1025	0.508591=	0.499053	944	0.568694=	0.571757
MaF11	3	1071	1.144277=	1.138350	998	1.142115=	1.142002	800	1.146045–	1.159070
MaF12	3	743	0.534650–	0.590104	761	0.535199–	0.592456	626	0.533782–	0.543327
MaF13	3	1021	0.366427–	0.369516	1174	0.501206–	0.524071	946	0.401748–	0.456261
CIBN1	2	1367	0.335619–	0.508695	1325	0.334598–	0.488058	518	0.381507–	0.506510
CIBN2	2	811	0.667394–	0.670158	766	0.668834–	0.669695	406	0.670678–	0.677235

(continued)

Table 7.7 (continued)

Problem	M	$t_{\max}$	θ-DEA	θ-DEA- UIP	$t_{\max}$	MOEA/DD	MOEA/DD- UIP	$t_{\max}$	LHFiD	LHFiD- UIP
CIBN3	2	1013	0.213996–	0.225995	914	0.214547–	0.226747	426	0.213834–	0.222645
CIBN4	3	505	0.895468–	0.898464	480	0.896755=	0.896848	393	0.992818–	1.028556
CIBN5	3	299	0.619556=	0.619234	305	0.622079=	0.621281	359	0.622345=	0.622445
DASCMOP1	2	1889	0.087565–	0.332968	1944	0.132878–	0.333200	1929	0.082801–	0.329137
DASCMOP2	2	1851	0.415182–	0.664793	1963	0.472963–	0.669519	1876	0.410152–	0.663136
DASCMOP3	2	1836	0.398531–	0.489664	1866	0.408387–	0.483395	2973	0.402701=	0.392024
DASCMOP4	2	2006	0.335362–	0.335532	2183	0.335677=	0.335616	1741	0.267030=	0.250359
DASCMOP5	2	2056	0.671229=	0.671456	2293	0.671290=	0.671308	1863	0.646158=	0.629977
DASCMOP6	2	2536	0.574568–	0.574866	2024	0.575065–	0.575132	1887	0.549128=	0.532582
DASCMOP7	3	1767	1.014610=	1.011353	1888	1.016219=	1.017115	2257	1.028625=	1.028417
DASCMOP8	3	1751	0.648435=	0.632479	1850	0.651319=	0.651529	2115	0.655370=	0.655050
DASCMOP9	3	1524	0.451661–	0.643508	1511	0.638729–	0.647314	2201	0.641902–	0.642864
MW1	2	1022	0.414481=	0.414439	1074	0.413709–	0.414543	1268	0.415326+	0.415005
MW2	2	833	0.482597–	0.489976	891	0.482312–	0.490426	813	0.327672=	0.330055
MW3	2	1050	0.469747=	0.469411	971	0.469695=	0.469723	1619	0.417336=	0.000000
MW4	3	756	1.024349=	1.026601	755	1.024167–	1.028355	879	1.039593=	1.039663
MW5	2	1632	0.100401–	0.200033	1453	0.195836=	0.200113	1125	0.082853=	0.010235
MW6	2	1212	0.297263–	0.310665	1272	0.296733–	0.309847	814	0.119352=	0.119553
MW7	2	905	0.365294=	0.365080	890	0.365752=	0.366039	814	0.363979=	0.364093
MW8	3	732	0.609678=	0.612839	668	0.615115=	0.624875	1223	0.391247=	0.472479
MW9	2	986	0.293277–	0.295267	1128	0.293964=	0.293913	764	0.295782=	0.295258
MW10	2	1032	0.246414=	0.263219	1054	0.261012=	0.289881	1680	0.000000=	0.000000
MW11	2	987	0.265164=	0.208145	1087	0.265863=	0.254258	681	0.268410=	0.268541
MW12	2	988	0.570107–	0.570262	1085	0.570166–	0.570440	732	0.570348=	0.570297
MW13	2	990	0.322834–	0.336527	945	0.322475=	0.334442	723	0.299718=	0.301507
MW14	3	894	0.156222=	0.155941	960	0.164248=	0.168807	679	0.213535+	0.209878
Total $(+/=/-) \longrightarrow$			01/25/23			01/28/20			04/29/16	of 49 probs.

Table 7.8 HV-based performance comparison of θ-DEA-UIP, MOEA/DD-UIP, and LHFiD-UIP with their respective base variants, i.e., θ-DEA, MOEA/DD, and LHFiD, respectively. For each test instance of $M = 5$, 8, and 10, median hypervolume is shown at the end of $t_{\max}$ generations determined on-the-fly for the respective UIP variants. The symbols '+', '=', or '−' against each base variant highlight where these are statistically better than, comparable to, or worse than the corresponding UIP variant, respectively (taken from [14])

Problem	M	$t_{\max}$	θ-DEA	θ-DEA- UIP	$t_{\max}$	MOEA/DD	MOEA/DD-UIP	$t_{\max}$	LHFiD	LHFiD- UIP
DTLZ1	5	748	2.157168=	2.155684	883	2.670244=	2.456018	859	4.207928=	4.052889
	8	875	9.981730=	9.979700	853	6.960960+	5.976893	869	9.988074=	9.982169
	10	833	0.000000=	0.000000	808	16.139124+	15.307207	1223	17.757704=	17.757702
DTLZ2	5	941	2.121200=	2.117510	894	2.121840=	2.120590	578	3.891166=	3.890823
	8	905	9.763760+	9.757400	839	9.758620=	9.755060	644	9.847959=	9.847937
	10	877	17.591500=	17.592100	834	17.574200=	17.566900	676	17.694772=	17.694515
DTLZ3	5	765	2.082338=	2.012005	904	0.260120=	0.066750	776	0.783335+	0.000000
	8	847	8.546232+	7.622917	803	8.319002=	8.116630	863	9.245672=	0.014920
	10	819	0.000000=	0.000000	814	9.385090+	6.920623	881	17.633675=	17.105152
DTLZ4	5	969	2.129440=	2.128680	960	2.133190=	2.130790	868	3.895547=	3.895753
	8	857	9.731910+	9.707990	784	9.713380=	9.714600	841	9.849535=	9.849960
	10	763	17.219400=	17.165100	754	17.296200=	17.248200	857	17.696040=	17.696277
MaF1	5	844	0.056371=	0.056379	856	0.055339=	0.056772	828	0.037931=	0.037928
	8	928	0.013510=	0.012873	839	0.013111=	0.013142	886	0.000128=	0.000129
	10	773	0.001562–	0.001647	757	0.001787–	0.001837	862	0.000724=	0.000724
MaF2	5	366	0.988566–	0.995460	376	0.981635=	0.984138	468	0.867556+	0.843094
	8	454	4.122040–	4.177970	455	4.039790–	4.096930	580	1.290279+	1.175563
	10	521	7.475370–	7.613500	535	7.321450–	7.494960	594	8.161162+	7.354750
MaF3	5	1209	1.977794=	1.866041	1842	2.483580=	2.378010	1729	2.160181=	2.160109
	8	612	0.000000=	0.000007	1504	0.000000=	0.000045	1453	3.432204+	3.432000
	10	461	0.000000=	0.000000	568	0.000000=	0.000237	1401	17.757727+	17.757130
MaF4	5	863	0.192197=	0.220878	820	0.154150=	0.168207	1101	0.112496=	0.108572
	8	963	0.032738=	0.045256	833	8.957987=	8.748836	1172	0.000273–	0.000533
	10	1222	0.026055–	0.034063	975	0.015330=	0.019916	1146	0.000000=	0.000000

(continued)

Table 7.8 (continued)

Problem	M	$t_{\max}$	θ-DEA	θ-DEA- UIP	$t_{\max}$	MOEA/DD	MOEA/DD-UIP	$t_{\max}$	LHFiD	LHFiD- UIP
MaF5	5	1157	2.487720=	2.487770	1173	2.487690–	2.487740	1039	2.160346–	2.161071
	8	1254	9.988720=	9.988720	1167	9.988720–	9.988720	970	3.431163–	3.432212
	10	1292	17.757700=	17.757700	1090	17.757700=	17.757700	1020	15.021723+	6.174841
MaF7	5	1090	0.956168=	0.955427	1126	0.968569+	0.964265	1244	0.657965–	0.716870
	8	1022	4.222140–	4.262750	959	4.291200=	4.295080	1562	0.453748–	0.562075
	10	766	6.468070=	6.712380	514	4.935440–	5.351250	1715	0.000000=	0.000000
MaF8	5	1767	0.000233–	0.000340	1853	0.000094–	0.000208	1675	0.000392+	0.000390
	8	1429	0.000517–	0.000570	1508	0.000390=	0.000382	1647	0.000003=	0.000003
	10	1317	0.000075=	0.000081	1438	0.000063=	0.000061	1595	0.000117=	0.000117
MaF9	5	2037	0.028336=	0.028005	1607	0.026104=	0.027067	5632	0.022343–	0.024079
	8	849	0.000833=	0.001078	745	0.000000=	0.000004	1132	0.000054=	0.000054
	10	786	0.000014=	0.000042	912	0.000022–	0.000071	1151	0.000504=	0.000504
MaF10	5	1049	0.830748–	0.877855	975	0.828694–	0.879025	1180	0.967104–	0.991590
	8	888	2.709280–	3.293280	931	2.599670–	3.326120	1284	1.498149–	1.578252
	10	962	5.286690–	5.609290	938	4.797120–	5.624200	1475	0.689354+	0.254470
MaF11	5	1048	2.376760=	2.370580	1020	2.361810=	2.364970	963	2.143115–	2.147966
	8	1105	9.448400=	9.465670	1073	9.368320=	9.367990	1027	3.395103=	3.396399
	10	1193	16.919800–	17.053500	1149	16.695000–	16.906100	1055	15.582983+	14.025161
MaF12	5	558	1.671030=	1.647670	505	1.637250=	1.622540	663	1.615504–	1.668499
	8	431	6.450900=	6.438770	462	5.856190=	5.954880	718	2.724894–	2.840152
	10	490	11.461000=	11.612800	496	10.449200=	10.762600	750	0.000000=	0.000000
MaF13	5	857	0.248616=	0.513728	944	0.409027–	0.515826	1815	0.204676–	0.521277
	8	966	0.269744=	0.297863	1132	0.621752=	1.007820	2091	0.057134–	0.057623
	10	1211	0.160839–	0.213371	1164	1.268000–	1.918400	2414	0.256875–	0.474652
Total $(+/=/-) \longrightarrow$			03/32/13			04/30/14			10/24/14	of 48 probs.

Therefore, any methodological intervention that could help ensure a reasonably good PF-approximation in fewer solution evaluations could have immense utility. However, it cannot be ignored that any such intervention may require additional solution evaluations and/or additional computational time for the underlying methodology to be exercised. Hence, considering the collective requirements of the original optimizer and the added methodology, both the total solution evaluations and the total run-time need to be taken into account. In this context, it is notable that for a desired or pre-fixed quality of PF-approximation, any methodological intervention may pose two promising scenarios vis-à-vis the base case (without any intervention), where:

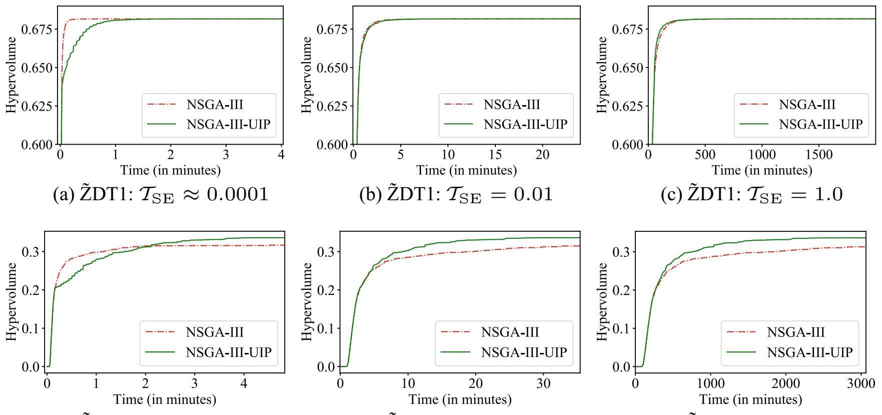
- Fewer total solution evaluations may be needed, and also a lower run-time.
- Fewer total solution evaluations may be needed, but a higher run-time.

In the context of the UIP operator, the results discussed above testify to its promise of fewer total solution evaluations to achieve the desired quality of the PF-approximation. These include:

- The fact that no new solution evaluations are required, and
- The fact that when integrated with an RV-EMâOA, it promises a better or equivalent PF-approximation than the stand-alone RV-EMâOA, *in any given generation* (as in Figs. 7.4 and 7.5a).

Critically, any specific generation of an RV-EMâOA-UIP run, where either or both of IP2 and IP3 are invoked, will take more time than any generation of the base RV-EMâOA (without UIP), which may be attributed to the construction of the underlying training-dataset and subsequent time-consuming training of ML model(s). Hence, *a better PF-approximation after a fixed number of generations (equivalently, after a fixed number of solution evaluations)*, as observed with respect to Figs. 7.4 and 7.5a, *may not necessarily translate to a better PF-approximation in lower total run-time*. This sets the motivation for investigating the UIP operator with regard to its associated run-time. To this end, a sample analysis is presented here, focusing on the performance of NSGA-III and NSGA-III-UIP on $\tilde{Z}$DT1 and $\tilde{Z}$DT6 problems. Let $\mathcal{T}_{\text{SE}}$ denote the time, in *seconds*, required for the evaluation of a solution. Under the computational setup employed and the experimental settings highlighted earlier, it turns out that $\mathcal{T}_{\text{SE}} = 1.07\text{e-}04$ (0.107 milliseconds) and 1.05e–04 (0.105 milliseconds) for $\tilde{Z}$DT1 and $\tilde{Z}$DT6, respectively. For both problems, NSGA-III-UIP has been run until the respective $t_{\max}$ generations (determined on-the-fly); and the average run-time required (among 31 randomly seeded runs) has been recorded for both problems. Subsequently, the base NSGA-III has been run on both problems, until the generation where the corresponding run-time matches with that of NSGA-III-UIP. The median hypervolume plots for both problems, with run-time on the horizontal axis, are shown in Fig. 7.7a, d. In that:

- Figure 7.7a shows the plot for $\tilde{Z}$DT1, where both NSGA-III and NSGA-III-UIP achieve a similar hypervolume toward the end. Hence, $\tilde{Z}$DT1 serves as an example where the base NSGA-III could arrive at a reasonable PF-approximation on its own. However, the intermediate trend suggests that NSGA-III could achieve the same hypervolume faster (in terms of run-time) than NSGA-III-UIP.



(a) $\tilde{Z}$DT1: $\mathcal{T}_{\mathrm{SE}} \approx 0.0001$ (b) $\tilde{Z}$DT1: $\mathcal{T}_{\mathrm{SE}} = 0.01$ (c) $\tilde{Z}$DT1: $\mathcal{T}_{\mathrm{SE}} = 1.0$

(d) $\tilde{Z}$DT6: $\mathcal{T}_{\mathrm{SE}} \approx 0.0001$ (e) $\tilde{Z}$DT6: $\mathcal{T}_{\mathrm{SE}} = 0.01$ (f) $\tilde{Z}$DT6: $\mathcal{T}_{\mathrm{SE}} = 1.0$

Fig. 7.7 Generation-wise hypervolume trend in the respective median runs of NSGA-III and NSGA-III-UIP. In that, NSGA-III-UIP has been run till termination generations ($t_{\max} = 1198$ and 1836 for $\tilde{Z}$DT1 and $\tilde{Z}$DT6, respectively); and NSGA-III has been run till the generation where the corresponding run-time matches to that of NSGA-III-UIP

- Figure 7.7d shows the plot for $\tilde{Z}$DT6, where NSGA-III-UIP achieves a better hypervolume than the base NSGA-III, toward the end. Hence, $\tilde{Z}$DT6 serves as an example where the base NSGA-III could *not* arrive at a reasonable *PF*-approximation on its own. Although the initial performance was better for base NSGA-III, NSGA-III-UIP eventually performed better.

The specific instance of $\tilde{Z}$DT1 above could *mistakenly* lead to the inference that *if base NSGA-III can arrive at a reasonable PF-approximation on its own, it would always require a lower run-time, even though NSGA-III-UIP may require fewer generations or equivalently fewer solution evaluations*. The basis for such a misconception has been countered below, through variation in $\mathcal{T}_{\mathrm{SE}}$. To symbolically emulate real-world scenarios where each solution evaluation can take a significantly longer time, two hypothetical values of $\mathcal{T}_{\mathrm{SE}} = 0.01$ and $\mathcal{T}_{\mathrm{SE}} = 1.0$ have been considered. Although, in the case of $\tilde{Z}$DT1, the solution evaluations, actual $\mathcal{T}_{\mathrm{SE}} = 1.07\text{e-}04$, have been padded by 0.01 and 1.0 s, respectively (using the *sleep* function available in the computational setup used), to emulate the two scenarios. The corresponding median hypervolume plots, with respect to run-time, are shown in Fig. 7.7b, c. Notably:

- In Fig. 7.7a with $\mathcal{T}_{\mathrm{SE}} \approx 0.0001$, NSGA-III achieved a better hypervolume than NSGA-III-UIP, initially, while both achieved a similar hypervolume toward the end.
- In Fig. 7.7b with $\mathcal{T}_{\mathrm{SE}} = 0.01$, NSGA-III achieved a hypervolume similar to NSGA-III-UIP, both initially and toward the end.
- In Fig. 7.7c with $\mathcal{T}_{\mathrm{SE}} = 1.0$, NSGA-III achieved a worse hypervolume than NSGA-III-UIP, initially, while both achieved a similar hypervolume toward the end.

The above analysis demonstrates that, if $\mathcal{T}_{\text{SE}}$ is relatively higher, say 1.0 s (plausible in many real-world scenarios), even in problems where base NSGA-III can arrive at a reasonable PF-approximation on its own, NSGA-III-UIP may require a lower run-time, compared to NSGA-III. For completeness, similar plots for ZDT6 have also been presented in Fig. 7.7e, f. Based on the above investigation, the following can be inferred:

- In problems where base NSGA-III can arrive at a reasonable PF-approximation on its own, NSGA-III-UIP may require a lower run-time, if the time required for each solution evaluation is sufficiently high.
- In problems where the base NSGA-III cannot arrive at a reasonable PF-approximation on its own, given the same run-time, NSGA-III-UIP may arrive at a better PF-approximation, regardless of the time required for each solution evaluation.

Clearly, the utility of incorporating the UIP operator may be even more significant in problems where each solution evaluation requires significant time.

7.8 Summary

In this chapter, the UIP operator for simultaneous and adaptive enhancement of convergence and diversity has been presented. The UIP operator relies on independent invocations IP2 and IP3 operators for the creation of both pro-convergence and pro-diversity offspring. Notably, in any generation of an RV-EMâOA-UIP run, either or both of IP2 and IP3 operators may be invoked. Furthermore, the same criteria for the first and subsequent invocations of IP2 and IP3 operators have been retained, as presented originally in Chaps. 5 and 6. The hallmark of the UIP operator is that its design and usage is guided by the overarching criteria to—avoid additional solution evaluations beyond those required by the base RV-EMâOA; favorably manage the convergence–diversity balance and ML-based risk-reward trade-off; and minimize ad-hoc parameter fixes. The efficacy of the UIP operator has been established: (i) with respect to IP2 and IP3 operators for NSGA-III, and (ii) in general for NSGA-III, θ-DEA, MOEA/DD, and LHFiD. Notably, NSGA-III-UIP performed better than: (i) NSGA-III-IP2 in about 19% of convergence-hard MOPs, and (ii) NSGA-III-IP3 in about 21% of diversity-hard MOPs. This supports the claim that the use of pro-convergence and pro-diversity offspring solutions simultaneously, regardless of the underlying convergence- or diversity-hard problem characteristics, offers a better PF-approximation than is possible without them. In general, the UIP variants of the four RV-EMâOAs collectively performed: (i) better in about 36% of the instances, and (ii) either better or equivalently in about 93% of the instances, compared to their respective base variants, on the MOPs and MaOPs considered.

Appendix

7.9 UIP Operator's Integration with Other RV-EMâOAs

As mentioned earlier, the UIP operator is generic and can ideally be integrated with any RV-EMâOA. To facilitate its implementation, three RV-EMâOAs (in addition to NSGA-III) have been selected, including, θ-DEA, MOEA/DD, and LHFiD. The algorithmic details of the resulting θ-DEA-UIP, MOEA/DD-UIP, and LHFiD-UIP algorithms are provided below.

7.9.1 θ-DEA-UIP

The algorithmic description of any generation t of θ-DEA-UIP is summarized in Algorithm 7.4. In that, the target-archive T_t as required by the IP2 function is updated first (line 1, Algorithm 7.4). Then the prerequisite conditions for the invocation of IP2 and IP3 are checked, and if fulfilled, the flags $startIP2$ and $startIP3$ are activated (lines 2–5, Algorithm 7.4). In the subsequent generations:

- If the IP2 operator is invoked, then the trained ML model is used for the progression of 50% of the (randomly chosen) natural offspring, leading to Q_t^{IP2} (lines 6–7, Algorithm 7.4).
- If IP3 operator is invoked, then the trained ML model is used for progression of 50% of the (judiciously chosen) parent solutions, leading to Q_t^{IP3} (lines 8–9, Algorithm 7.4).
- If none or only one of IP2 and IP3 is invoked, the number of offspring shall be less than N. Hence, the remaining offspring are created using the natural variation operators, denoted by Q_t^{V} (lines 10–11, Algorithm 7.4).
- All offspring, namely, Q_t^{IP2}, Q_t^{IP3} and Q_t^{V} are merged into Q_t (sized N), and evaluated (lines 12–13, Algorithm 7.4).
- Update the input-archive A_{t+1} as required by the `IP2` function (line 14, Algorithm 7.4).
- The steps in lines 15–27 (Algorithm 7.4) relate to the steps of the survival selection procedure of θ-DEA [19].
- $t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$ are adapted, if the respective operators were invoked in the current generation t (lines 28–31, Algorithm 7.4). This adaptation is based on the survival of the respective offspring vis-a-vis the natural offspring, as described in Sect. 7.1.6.
- Finally, if both IP2 and IP3 are invoked in the current generation, it is ensured that neither $t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$ have a value lesser than two (lines 32–34, Algorithm 7.4), as explained in Sect. 7.1.4.

Algorithm 7.4: Generation t of θ-DEA-UIP

```
Input: RV set $\mathcal{R}$, variable bounds $[X^{(L)}, X^{(U)}]$, parent population $P_t$, offspring survived
       $N_{t-1}^{\text{surv(V)}}$
       IP2-specific: target archive $T_{t-1}$, input archive $A_t$, frequency $t_{\text{freq}}^{\text{IP2}}$
       IP3-specific: neighborhood radius $r$, frequency $t_{\text{freq}}^{\text{IP3}}$
Output: $P_{t+1}$, $T_t$, $A_{t+1}$, $t_{\text{freq}}^{\text{IP2}}$, $t_{\text{freq}}^{\text{IP3}}$
1  $T_t \leftarrow$ Update the target archive for IP2 operator
2  if population is completely non-dominated then
3      $startIP2 = True$
4  if population has mildly stabilized then
5      $startIP3 = True$
6  if $startIP2$ & $t_{\text{freq}}^{\text{IP2}}$ passed after last invocation then
7      $Q_t^{\text{IP2}} \leftarrow$ Create 50% offspring using IP2 (Algorithm 7.1)
8  if $startIP3$ & $t_{\text{freq}}^{\text{IP3}}$ generations passed after last invocation then
9      $Q_t^{\text{IP3}} \leftarrow$ Create 50% offspring using IP3 (Algorithm 7.2)
10 if offspring created are insufficient ($< N$) then
11     $Q_t^{\text{V}} \leftarrow$ Create rest of offspring using natural variation operators
12 $Q_t \equiv$ Merge $Q_t^{\text{IP2}}$, $Q_t^{\text{IP3}}$ and $Q_t^{\text{V}}$ (total $N$ offspring)
13 Evaluate $Q_t$
14 $A_{t+1} \leftarrow$ Update the input archive for IP2 operator
15 $R_t \leftarrow$ Merge $P_t$ and $Q_t$
16 $S_t \leftarrow$ Perform Pareto Non-dominated Sorting on $R_t$
17 Update the ideal point from solutions of $S_t$
18 Normalize $S_t$ using $Z^{\text{ideal}}$ and $Z^{\text{nadir}}$ as bounds
19 $\mathcal{C} \leftarrow$ Perform clustering on $S_t$ using $\mathcal{R}$
20 $\{F'_1, F'_2, \ldots\} \leftarrow$ Perform $\theta$-non-dominated sorting on $S_t$
21 Initialize $P_{t+1}$ as an empty set
22 $i \leftarrow 1$
23 while $|P_{t+1}| + |F'_i| < N$ do
24     $P_{t+1} \leftarrow P_{t+1} \cup F'_i$
25     $i \leftarrow i + 1$
26 Randomly sort the solutions in $F'_i$
27 $P_{t+1} \leftarrow P_{t+1} \cup F'_i[N - |P_{t+1}|]$
28 if IP2 was invoked in current generation then
29     Update $t_{\text{freq}}^{\text{IP2}}$ (Section 7.1.6)
30 if IP3 was invoked in current generation then
31     Update $t_{\text{freq}}^{\text{IP3}}$ (Section 7.1.6)
32 if both IP2 and IP3 were invoked in current generation then
33     if $t_{\text{freq}}^{\text{IP2}} < 2$ then $t_{\text{freq}}^{\text{IP2}} = 2$
34     if $t_{\text{freq}}^{\text{IP3}} < 2$ then $t_{\text{freq}}^{\text{IP3}} = 2$
```

7.9.2 MOEA/DD-UIP

The algorithmic description of any generation t of MOEA/DD-UIP is summarized in Algorithm 7.5. In that, the target-archive T_t as required by the IP2 function is updated first (line 1, Algorithm 7.5). Then the prerequisite conditions for invocation of IP2 and IP3 are checked, and if fulfilled, appropriate flags ($startIP2$, $startIP3$) which influence whether or not IP2 and IP3 are to be invoked, are triggered as True (lines 2–5, Algorithm 7.5). In the subsequent generations:

Algorithm 7.5: Generation t of MOEA/DD-UIP

```
Input: RV set $\mathcal{R}$, variable bounds $[X^{(L)}, X^{(U)}]$, parent population $P_t$, offspring survived
       $N_{t-1}^{\text{surv(V)}}$
       IP2-specific: target archive $T_{t-1}$, input archive $A_t$, frequency $t_{\text{freq}}^{\text{IP2}}$
       IP3-specific: neighborhood radius $r$, frequency $t_{\text{freq}}^{\text{IP3}}$
Output: $P_{t+1}$, $T_t$, $A_{t+1}$, $t_{\text{freq}}^{\text{IP2}}$, $t_{\text{freq}}^{\text{IP3}}$
1  $T_t \leftarrow$ Update the target-archive for IP2 operator
2  if population is completely non-dominated then
3      $startIP2 = True$
4  if population has mildly stabilized then
5      $startIP3 = True$
6  if $startIP2$ & $t_{\text{freq}}^{\text{IP2}}$ generations passed after last invocation then
7      $Q_t^{\text{IP2}} \leftarrow$ Create 50% offspring using IP2 (Algorithm 7.1)
8  if $startIP3$ & $t_{\text{freq}}^{\text{IP3}}$ generations passed after last invocation then
9      $Q_t^{\text{IP3}} \leftarrow$ Create 50% offspring using IP3 (Algorithm 7.2)
10 if offspring created are insufficient ($< N$) then
11     $\bar{P}_t \leftarrow$ Perform mating selection on $P_t$
12     $Q_t^{\text{V}} \leftarrow$ Create rest of offspring using natural variation operators
13 $Q_t \leftarrow$ Merge $Q_t^{\text{IP2}}$, $Q_t^{\text{IP3}}$ and $Q_t^{\text{V}}$ (total $N$ offspring)
14 Evaluate $Q_t$
15 $A_{t+1} \leftarrow$ Update input-archive for IP2 operator
16 Initialize $P_{t+1}$ as $P_t$
17 for each offspring $q \in Q_t$ do
18     $P_{t+1} \leftarrow$ Update the population $P_{t+1}$ using $q$
19 if IP2 was invoked in current generation then
20     Update $t_{\text{freq}}^{\text{IP2}}$ (Section 7.1.6)
21 if IP3 was invoked in current generation then
22     Update $t_{\text{freq}}^{\text{IP3}}$ (Section 7.1.6)
23 if both IP2 and IP3 were invoked in current generation then
24     if $t_{\text{freq}}^{\text{IP2}} < 2$ then $t_{\text{freq}}^{\text{IP2}} = 2$
25     if $t_{\text{freq}}^{\text{IP3}} < 2$ then $t_{\text{freq}}^{\text{IP3}} = 2$
```

- If the IP2 operator is invoked, then the trained ML model is used for the progression of 50% of the (randomly chosen) natural offspring, leading to Q_t^{IP2} (lines 6–7, Algorithm 7.5).
- If the IP3 operator is invoked, then the trained ML model is used for progression of 50% of the (judiciously chosen) parent solutions, leading to Q_t^{IP3} (lines 8–9, Algorithm 7.5).
- If none or only one of IP2 and IP3 is invoked, the number of offspring shall be less than N. Hence, the remaining offspring are created using the variation operators, denoted by Q_t^{V} (lines 10–12, Algorithm 7.5).
- All offspring, namely, Q_t^{IP2}, Q_t^{IP3} and Q_t^{V} are merged into Q_t (sized N), and evaluated (lines 13–14, Algorithm 7.5).
- Update the input-archive A_{t+1} as required by the `IP2` function (line 15, Algorithm 7.5).
- The steps in lines 16–18 (Algorithm 7.5) relate to the steps of MOEA/DD's survival selection procedure [12].
- $t_{\mathrm{freq}}^{\mathrm{IP2}}$ and $t_{\mathrm{freq}}^{\mathrm{IP3}}$ are adapted, if the respective operators were invoked in the current generation t (lines 19–22, Algorithm 7.5). This adaptation is based on the survival of the respective offspring vis-a-vis the natural offspring, as described in Sect. 7.1.6.
- Finally, if both IP2 and IP3 are invoked in the current generation, it is ensured that neither $t_{\mathrm{freq}}^{\mathrm{IP2}}$ nor $t_{\mathrm{freq}}^{\mathrm{IP3}}$ have a value lower than two (lines 23–25, Algorithm 7.5), as explained in Sect. 7.1.4.

7.9.3 LHFiD-UIP

The algorithmic description of any generation t of LHFiD-UIP is summarized in Algorithm 7.6. In that, first the target-archive T_t as required by the IP2 function is updated (line 1, Algorithm 7.6). Then the prerequisite conditions for invocations of IP2 and IP3 are checked, and if fulfilled, appropriate flags ($startIP2$, $startIP3$), which influence whether or not IP2 and IP3 are to be invoked, are triggered as True (lines 2–5, Algorithm 7.6). In the subsequent generations:

- If the IP2 operator is invoked, then the trained ML model is used for the progression of 50% of the (randomly chosen) natural offspring, leading to Q_t^{IP2} (lines 6–7, Algorithm 7.6).
- If the IP3 operator is invoked, then the trained ML model is used for the progression of 50% of the (judiciously chosen) parent solutions, leading to Q_t^{IP3} (lines 8–9, Algorithm 7.6).
- If none or only one of IP2 and IP3 gets invoked, the number of offspring shall be less than N. Hence, the remaining offspring are created using the variation operators, denoted by Q_t^{V} (lines 10–11, Algorithm 7.6).
- All offspring, namely, Q_t^{IP2}, Q_t^{IP3} and Q_t^{V} are merged into Q_t (sized N), and evaluated (lines 12–13, Algorithm 7.6).

Algorithm 7.6: Generation t of LHFiD-UIP

```
Input: RV set $\mathcal{R}$, variable bounds $[X^{(L)}, X^{(U)}]$, parent population $P_t$, offspring survived
       $N_{t-1}^{\text{surv(V)}}$
       IP2-specific: target archive $T_{t-1}$, input archive $A_t$, frequency $t_{\text{freq}}^{\text{IP2}}$
       IP3-specific: neighborhood radius $r$, frequency $t_{\text{freq}}^{\text{IP3}}$
Output: $P_{t+1}$, $T_t$, $A_{t+1}$, $t_{\text{freq}}^{\text{IP2}}$, $t_{\text{freq}}^{\text{IP3}}$
1  $T_t \leftarrow$ Update target-archive for IP2 operator
2  if population is completely non-dominated then
3      $startIP2 = True$
4  if population has mildly stabilized then
5      $startIP3 = True$
6  if $startIP2$ & $t_{\text{freq}}^{\text{IP2}}$ generations passed after last invocation then
7      $Q_t^{\text{IP2}} \leftarrow$ Create 50% offspring using IP2 (Algorithm 7.1)
8  if $startIP3$ & $t_{\text{freq}}^{\text{IP3}}$ generations passed after last invocation then
9      $Q_t^{\text{IP3}} \leftarrow$ Create 50% offspring using IP3 (Algorithm 7.2)
10 if offspring created are insufficient $(< N)$ then
11     $Q_t^{\text{V}} \leftarrow$ Create rest of offspring using natural variation operators
12 $Q_t \leftarrow$ Merge $Q_t^{\text{IP2}}$, $Q_t^{\text{IP3}}$ and $Q_t^{\text{V}}$ (total $N$ offspring)
13 Evaluate $Q_t$
14 $A_{t+1} \leftarrow$ Update input-archive for IP2 operator
15 $U_t \leftarrow$ Merge $P_t$ and $Q_t$
16 $Z^I \leftarrow$ Compute the ideal point from solutions in $U_t$
17 if $Z^N$ is an empty set then
18     $\tilde{F} \leftarrow$ Translate the solutions in $U_t$ using $Z^I$
19 else
20     $\tilde{F} \leftarrow$ Normalize the solutions in $U_t$ using $Z^I$ & $Z^N$
21 $P_{t+1} \leftarrow$ Perform Survival selection on $U_t$
22 if population has mildly stabilized then
23     Update the nadir point $Z^N$
24 if IP2 was invoked in current generation then
25     Update $t_{\text{freq}}^{\text{IP2}}$ (Section 7.1.6)
26 if IP3 was invoked in current generation then
27     Update $t_{\text{freq}}^{\text{IP3}}$ (Section 7.1.6)
28 if both IP2 and IP3 were invoked in current generation then
29     if $t_{\text{freq}}^{\text{IP2}} < 2$ then $t_{\text{freq}}^{\text{IP2}} = 2$
30     if $t_{\text{freq}}^{\text{IP3}} < 2$ then $t_{\text{freq}}^{\text{IP3}} = 2$
```

- Update the input-archive A_{t+1} as required by the IP2 function (line 14, Algorithm 7.6).
- The steps in lines 15–21 (Algorithm 7.6) relate to the steps of survival selection, as in the base LHFiD proposed originally.

- If mild stabilization of the population is detected, then the nadir point is updated (lines 22–23, Algorithm 7.6).
- $t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$ are adapted, if the respective operators were invoked in the current generation t (lines 24–27, Algorithm 7.6). This adaptation is based on the survival of the respective offspring vis-a-vis the natural offspring, as described in Sect. 7.1.6.
- Finally, if both IP2 and IP3 are invoked in the current generation, it is ensured that neither $t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$ have a value lesser than two (lines 28–30, Algorithm 7.6), as explained in Sect. 7.1.4.

Notably, LHFiD, as originally proposed in [17], does not include constraint handling. Owing to the use of constrained problems here:

- Its random mating selection procedure is replaced with the constraint-based mating selection procedure [9].
- Its survival selection procedure is modified to select feasible solutions first. If the number of feasible solutions is less than N, then the infeasible solutions offering the least constraint violation values are selected to fill the population [9].

Further, there is a minor modification in the first invocation criterion of the IP2 operator when integrated with LHFiD. Instead of ensuring that the entire population is non-dominated, the non-domination is checked *locally* within each cluster (one cluster per RV). Since LHFiD (originally) does not have a non-domination check over the entire population, there is no inherent selection pressure for the entire population to be non-dominated. Hence, without the above modification, it is plausible that the IP2 may never be invoked over the entire LHFiD-UIP run.

References

1. Abdi, H.: Bonferroni and Sidak corrections for multiple comparisons. Encycl. Meas. Stat. 103–107 (2007)
2. Bossek, J., Grimme, C., Meisel, S., Rudolph, G., Trautmann, H.: Local search effects in bi-objective orienteering. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'18, p. 585–592. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3205455.3205548
3. Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., Yao, X.: A benchmark test suite for evolutionary many-objective optimization. Complex Intell. Syst. **3**, 67–81 (2017). https://doi.org/10.1007/s40747-017-0039-7
4. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998)
5. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
6. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535

7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, pp. 105–145. Springer London, London (2005). https://doi.org/10.1007/1-84628-137-7_6
8. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins, G.J.E. (ed.) Foundations of Genetic Algorithms, vol. 1, pp. 69–93. Elsevier (1991). https://doi.org/10.1016/B978-0-08-050684-5.50008-2
9. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. IEEE Trans. Evol. Comput. **18**(4), 602–622 (2014). https://doi.org/10.1109/TEVC.2013.2281534
10. Jaskiewicz, A.: Genetic local search for multiple objective combinatorial optimization. European J. Oper. Res. **371**(1), 50–71 (2002)
11. Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. J. Am. Stat. Assoc. **47**(260), 583–621 (1952). http://www.jstor.org/stable/2280779
12. Li, K., Deb, K., Zhang, Q., Kwong, S.: An evolutionary many-objective optimization algorithm based on dominance and decomposition. IEEE Trans. Evol. Comput. **19**(5), 694–716 (2015). https://doi.org/10.1109/TEVC.2014.2373386
13. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: Enhanced innovized progress operator for evolutionary multi- and many-objective optimization. IEEE Trans. Evol. Comput. **26**(5), 961–975 (2022). https://doi.org/10.1109/TEVC.2021.3131952
14. Mittal, S., Saxena, D.K., Deb, K., Goodman, E.D.: A unified innovized progress operator for performance enhancement in evolutionary multi- and many-objective optimization. IEEE Trans. Evol. Comput. 1 (2023). https://doi.org/10.1109/TEVC.2023.3321603
15. Murata, T., Nozawa, H., Tsujimura, Y., Gen, M., Ishinuchi, H.: Effect of local search only performance of celluar multi-objective genetic algorithms for designing fuzzy rule based classification systems. In: Proceeding of the Congress on Evolutionary Computation (CEC-2002), pp. 663–668 (2002)
16. Saxena, D.K., Kapoor, S.: On timing the nadir-point estimation and/or termination of reference-based multi- and many-objective evolutionary algorithms. In: Deb, K., Goodman, E., Coello Coello, C.A., Klamroth, K., Miettinen, K., Mostaghim, S., Reed P. (eds.) Evolutionary Multi-criterion Optimization, pp. 191–202. Springer International Publishing, Cham (2019)
17. Saxena, D.K., Mittal, S., Kapoor, S., Deb, K.: A localized high-fidelity-dominance based many-objective evolutionary algorithm. IEEE Trans. Evol. Comput. 1 (2022). https://doi.org/10.1109/TEVC.2022.3188064
18. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945). http://www.jstor.org/stable/3001968
19. Yuan, Y., Xu, H., Wang, B., Yao, X.: A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. **20**(1), 16–37 (2016). https://doi.org/10.1109/TEVC.2015.2420112

Chapter 8
Investigating Innovized Progress Operators with Different ML Methods

Chapters 5 and 6 have shown how learning efficient search directions from the intermittent generations' solutions could be utilized to create pro-convergence and pro-diversity offspring, enabling better convergence and diversity, respectively. The entailing steps of dataset preparation, training of ML models, and utilization of these models have been encapsulated as Innovized Progress operators, namely IP2 for convergence improvement and IP3 for diversity improvement. In these chapters, the goal was to establish that ML-based operators can potentially enhance the performance of RV-EMâOAs. In doing so, major emphasis was laid on the design of these operators adhering to the key considerations of convergence–diversity balance and ML risk–reward trade-off, and avoiding ad hoc parameter fixations and extra solution evaluations. Noticeably, the impact of the choice of the specific ML methods used in these operators was not discussed. However, to endorse the robustness of the proposed (IP2, IP3, and UIP) operators, it is imperative to investigate how significantly their performance can be influenced when the underlying ML methods are varied.

This chapter presents an exploratory analysis for both IP2 and IP3, based on eight different ML methods, tested against an exhaustive test suite comprising multi- and many-objective test instances. The results suggest that changing the underlying ML methods (Random Forest in the case IP2, and k-Nearest Neighbors in the case of IP3) does not significantly improve or deteriorate the operators' performance in terms of the hypervolume metric, though some gains in run-time can be observed for similar hypervolume measures. The results interestingly also reveal that different ML methods are characterized by different trade-offs between the hypervolume metric offered and the associated run-times. Based on these results, an ML method offering the best balance between hypervolume metric and run-time is identified, and is used as the underlying method for the UIP operator (implying that the chosen method is used in each of IP2 and IP3 operators). The remaining of the chapter is organized as follows: the details of the ML methods considered in this study are presented in Sect. 8.1, followed by the experimental settings in Sect. 8.2. The empirical investigation across ML methods on IP2, IP3, and UIP operators is presented in Sect. 8.3, followed by the conclusion in Sect. 8.4.

D. K. Saxena et al., *Machine Learning Assisted Evolutionary Multi- and Many-Objective Optimization*, Genetic and Evolutionary Computation,
https://doi.org/10.1007/978-981-99-2096-9_8

8.1 Alternative ML Methods for IP2 and IP3 Operators

This chapter considers eight different ML methods, covering linear and nonlinear Modeling, Boosting, and Trees, to investigate their suitability for the IP2 and IP3 operators. The chosen methods are highlighted below.

- From the family of linear methods: standard Linear Regression, Ridge Regression [7], and Elastic Net Regression [9] are included. Linear Regression based on the least squares approach is used as the base model, but it can also be fitted in other ways, such as by minimizing the lack of fit through a penalized version of the least squares cost function as in Ridge Regression (L2-norm penalty) and Lasso (L1-norm penalty). Since Lasso is known to suffer where there are correlated features [8], it has not been considered. Hence, Ridge Regression and Elastic Net Regression based on a linear combination of the L2-norm and L1-norm penalties are included.
- From the family of trees: Extra Trees Regressor and Random Forests are included, owing to their ability to efficiently handle complex, nonlinear, high-dimensional data, with a lower tendency for overfitting [6]. Notably, Random Forests aggregate the results from many decision trees, each generated from a bootstrap sample of the data. Here, at each node, one feature is selected to split on, from a random subset of all features. While in the case of Random Forests, the optimum split is chosen, Extra Trees do it randomly.
- From the family of boosting algorithms: XGBoost is included. It is an implementation of gradient boosting that is scalable and optimized for execution speed and model performance. This is an ensemble technique, where each new model added sequentially learns from the previous models' errors [2].
- From other nonlinear methods: k-Nearest Neighbors Regression and Support Vector Regression are included. The former method identifies the k nearest inputs of the test instance in the original training-dataset and returns the average of their respective targets [3]. The latter method seeks to find the hyperplane with the maximum number of points that lies within a threshold distance from the boundary line (unlike other regression models that try to minimize the error between the real and predicted values).

For ease of subsequent reference, the chosen ML methods have been abbreviated as follows: Linear Regression (LR), Ridge Regression (Ridge), Elastic Net Regression (ENet), Extra Trees Regressor (ExTree), Random Forests (RF), XGBoost (XGB), k-Nearest Neighbors (kNN), and Support Vector Regression (SVR).

8.2 Experimental Settings

This section lays the foundation for the experimental investigations by highlighting the (a) test suite considered, (b) parameters pertaining to the base RV-EMâOA (NSGA-III), and (c) performance indicators used and related statistical analysis.

8.2.1 Test Suite

The test suite in this chapter comprises a combination of multi-objective (both convergence- and diversity-hard) and many-objective problems, as previously used in Chaps. 5, 6, and 7. These include

- convergence-hard multi-objective problems: $\tilde{\text{Z}}$DT, DTLZ, and MaF.
- diversity-hard multi-objective problems: CIBN, DASCMOP, and MW.
- many-objective problems: DTLZ and MaF ($M = 5$, and 10).

Here, it may be noted that in Chap. 5 dedicated to the IP2 operator for convergence enhancement, only the convergence-hard multi-objective problems were considered, while many-objective problems were not. Similarly in Chap. 6 dedicated to the IP3 operator for diversity enhancement, only the diversity-hard multi-objective problems were considered, while many-objective problems were not. The rationale for the restricted experimentation in the above chapters relates to the fact that the focus in these chapters was more on the development of the respective operators and proof-of-concept that these operators bear the potential for performance enhancement of RV-EMâOAs. However, since the goal in this chapter is more exploratory, even many-objective problems are included for experiments pertaining to the IP2 and IP3 operators. In the same light, the UIP operator is tested for both multi- and many-objective optimization problems.

8.2.2 RV-EMâOA Settings

To obtain a reasonably sized set of RVs using the Das–Dennis method [4], the gap parameter is set as given in Table 8.1. In that table, where two values of p (gaps) are shown, the first value is used to create the boundary RVs and the second value is used to create the interior RVs [5]. For coherence, the population size N is kept the same as the number of RVs corresponding to a particular objective, as given in Table 8.1. Furthermore, the natural variation operators include SBX crossover ($p_c = 0.9$ and $\eta_c = 20$) and polynomial mutation ($p_c = 1/n$ and $\eta_m = 20$) for an n-variable problem.

Further, each RV-EMâOA has been run 31 times, with different random seeds. In that process, for the reference RV-EMâOA (explained below), the termination generation $t_{\max}$ has been determined on-the-fly through the stabilization tracking

Table 8.1 Parameter settings for the Das–Dennis method

Setting	$M = 2$	$M = 3$	$M = 5$	$M = 10$
p (gaps)	99	13	5, 4	3, 3
N	100	105	196	440

algorithm, using $\psi_{\text{term}} \equiv \{3, 50\}$. For all other RV-EMâOAs, the mean $t_{\max}$ determined for reference RV-EMâOA on 31 runs has been used as $t_{\max}$.

Reference RV-EMâOA is the algorithm against which the performance of other RV-EMâOAs is compared. For instance, the performance of NSGA-III-IP2 is to be compared amidst its eight variants corresponding to eight different ML methods (identified in Sect. 8.1). Since RF is the base method for the existing IP2 operator, NSGA-III-IP2-RF can be distinguished from the other NSGA-III-IP2-ML variants. Similarly NSGA-III-IP3-*k*NN can be distinguished from the remaining NSGA-III-IP3-ML variants. Since in this chapter, NSGA-III is the sole RV-EMâOA, its name can be avoided for brevity, and the task in this chapter translates to comparing the performance of (a) IP2-RF with all other IP2-ML, and (b) IP3-*k*NN with all other IP3-ML. Hence, NSGA-III-IP2-RF and NSGA-III-IP3-*k*NN are the reference RV-EMâOAs for the exploratory analysis on IP2 and IP3 operators, respectively.

8.2.3 Performance Indicators and Statistical Analysis

In this chapter, hypervolume (HV) has been used as the primary performance indicator. In that:

- the used reference points are given by $R_{1\times M} = [1 + \frac{1}{p}, \ldots, 1 + \frac{1}{p}]$, where p is the number of gaps set for the RV generating Das–Dennis method.
- the hypervolume values (31 in count) obtained from the final solutions of 31 randomly seeded runs IP2-RF are subjected to Wilcoxon ranksum test (for statistical significance) with a p-value threshold of 0.05, against the 31 hypervolume values available for each IP2-ML. This test only infers if the difference between IP2-RF and any IP2-ML is statistically insignificant (denoted by =). If not, then their respective median values are compared, and the better/worse performing method is denoted by a $+/-$ sign (as in Tables 8.2 and 8.3). Similar procedure is adopted for comparison of IP3-*k*NN with each IP3-ML.

8.3 Experimental Results

This section first presents the exploratory analysis of (a) IP2-ML vis-à-vis IP2-RF and (b) IP3-ML vis-à-vis IP3-*k*NN, followed by the underlying trade-off analysis. Subsequently, the best performing ML method(s) have been applied on the UIP operator to examine its performance, against the original UIP operator described in Chap. 7.

Table 8.2 HV-based comparison of IP2 with eight different ML methods on convergence-hard multi-objective problems, where RF is the base for pairwise comparisons. The symbols '+', '=', and '–' indicate whether the corresponding ML method is statistically better, equivalent, or worse, than RF. Here, $\hat{t}_{\mathrm{TM}}$ denotes the average of the termination generations for 31 runs of the base RF

Problem	M	$\hat{t}_{\mathrm{TM}}$	kNN	ExTree	SVR	ENet	Ridge	LR	XGB	**RF**
Z̃DT1	2	1187	0.681860=	0.681859=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681860
Z̃DT2	2	1289	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794=	0.348794
Z̃DT3	2	1013	1.068370=	1.068457=	1.068402=	1.068368=	1.068490=	1.068559=	1.068454=	1.068502
Z̃DT4	2	1770	0.681860=	0.681860=	0.681860=	0.681860=	0.681860=	0.681859=	0.681859=	0.681860
Z̃DT6	2	1779	0.326889+	0.321376=	0.327059+	0.317623=	0.319643=	0.323048=	0.317781=	0.320871
DTLZ1	3	834	1.222051=	1.221686=	1.221625=	1.221376=	1.221096=	1.221702=	1.220976=	1.221882
DTLZ2	3	953	0.667314–	0.667295–	0.667332=	0.667333=	0.667319–	0.66732–	0.667310–	0.667337
DTLZ3	3	980	0.651196=	0.651735=	0.655384=	0.643967–	0.65159=	0.655007=	0.661276+	0.656469
DTLZ4	3	1256	0.667330=	0.667310=	0.667319=	0.667326=	0.667315=	0.667311=	0.667327=	0.667285
MaF1	3	584	0.234406–	0.235446=	0.238137+	0.236298=	0.236871+	0.236791+	0.235838=	0.235178
MaF2	3	466	0.39697=	0.396911=	0.396957=	0.396724=	0.396598=	0.396991=	0.396655=	0.396592
MaF3	3	1409	1.248976=	1.248976=	1.248976=	1.248976=	1.248976=	1.248976=	1.248976=	1.248976
MaF4	3	676	1.248837=	1.248815=	1.248857+	1.24883=	1.248839=	1.248831=	1.248879+	1.248827
MaF5	3	1217	1.227618=	1.227598=	1.227607=	1.227616=	1.227611=	1.227605=	1.22761=	1.227605
MaF7	3	1161	0.376137=	0.376049=	0.375816=	0.375900=	0.375471=	0.375787=	0.375781=	0.375643
MaF8	3	1344	0.000000=	0.000000=	0.000000=	0.000000=	0.000000=	0.000000=	0.000000=	0.000000
MaF9	3	1289	0.626817=	0.626817=	0.626817=	0.626816=	0.626816=	0.626818=	0.626817=	0.626816
MaF10	3	904	0.519790=	0.512255=	0.519678=	0.510216=	0.519227=	0.520822=	0.509222=	0.514095
MaF11	3	938	0.981043=	0.980234=	0.980781=	1.147040=	1.145995=	0.979225=	0.980813=	0.979666
MaF12	3	709	0.613310=	0.613925=	0.613773=	0.613332=	0.614259=	0.613322=	0.613556=	0.612813
MaF13	3	907	0.372156=	0.372329=	0.365152–	0.366430–	0.375474=	0.376024+	0.373147=	0.371255
Total (+/=/–)			01/18/02	00/20/01	03/17/01	00/19/02	01/19/01	02/18/01	02/18/01 of 21 probs.	

Table 8.3 HV-based comparison of IP2 with eight different ML methods on many-objective problems, where RF is the base for pairwise comparisons. The symbols '+', '=', and '–' indicate whether the corresponding ML method is statistically better, equivalent, or worse, than RF. Here, $\hat{t}_{\text{TM}}$ denotes the average of the termination generations for 31 runs of the base RF (taken from [1])

Problem	M	$\hat{t}_{\text{TM}}$	kNN	ExTree	SVR	ENet	Ridge	LR	XGB	**RF**
DTLZ1	5	1206	2.485633=	2.486744=	2.486437=	2.486802=	2.486921=	2.486951=	2.486755=	2.479388
	10	2076	17.757704=	17.757704=	17.757704=	17.757707=	17.757711+	17.757708=	17.75771+	17.757703
DTLZ2	5	903	2.172478=	2.172430=	2.172397=	2.172605=	2.172648=	2.17262=	2.172606=	2.172439
	10	797	17.667911=	17.668915=	17.668607=	17.668416=	17.670702+	17.670357+	17.670246+	17.668409
DTLZ3	5	1196	2.111525=	2.110038=	2.086116=	2.103797=	2.102734=	2.109105=	2.083618=	0.975718
	10	1914	17.665819=	17.663740=	17.661898–	17.659894–	17.667285=	17.666322=	17.667097=	17.667027
DTLZ4	5	901	2.173324=	2.173321=	2.173259=	2.173090=	2.173194=	2.173259=	2.173285=	2.173402
	10	873	17.679661–	17.679996=	17.679292–	17.677214–	17.679034–	17.678961–	17.679912=	17.679992
MaF1	5	852	0.060541=	0.060107=	0.059505=	0.059808=	0.060784=	0.059723=	0.060170=	0.061855
	10	938	0.001951=	0.001928=	0.001968=	0.001973=	0.002021+	0.002039=	0.001882–	0.001984
MaF2	5	362	1.002621=	0.998740=	1.002171=	0.999238=	1.003607=	1.001332=	1.000969=	1.000973
	10	492	7.823111=	7.836465=	7.814538=	7.829009=	7.850499+	7.810496=	7.837603=	7.808070
MaF3	5	2509	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320
	10	1912	17.757727+	17.757727+	17.757727+	17.757727+	17.757727+	17.757727+	17.757727+	0.000000
MaF4	5	646	2.475006=	2.474046=	2.483075+	2.472548=	2.466242=	2.477218=	2.480569=	2.462381
	10	1106	17.481657=	17.487127=	17.487090=	17.488453=	17.483530=	17.484489=	17.476629–	17.482483

(continued)

Table 8.3 (continued)

Problem	M	$\hat{t}_{\mathrm{TM}}$	kNN	ExTree	SVR	ENet	Ridge	LR	XGB	**RF**
MaF5	5	1117	2.487940=	2.487937=	2.487936=	2.487937−	2.487939=	2.487938=	2.487937=	2.487940
	10	1180	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727
MaF7	5	1095	0.968628=	0.970901=	0.969541=	0.969176=	0.969783=	0.972155=	0.970013=	0.971900
	10	746	7.634033=	7.626016=	7.620365=	7.579781=	7.637727=	7.631401=	7.619364=	7.611451
MaF8	5	1361	0.000504=	0.000456=	0.000491=	0.000495=	0.000475=	0.000506=	0.000481=	0.000509
	10	1041	0.000087=	0.000087=	0.000082=	0.000082=	0.000081=	0.000080=	0.000084=	0.000085
MaF9	5	2996	0.025761=	0.027336=	0.026762=	0.027222=	0.026980=	0.026736=	0.028677=	0.027412
	10	689	0.000303=	0.000282=	0.000317=	0.000324=	0.000322=	0.000319=	0.000332+	0.000296
MaF10	5	909	0.933306=	0.901095=	0.901880=	0.896384=	0.889783=	0.926323=	0.891255=	0.930609
	10	753	6.298122=	6.102056=	6.318167=	6.310587=	6.247213=	6.255246=	6.269511=	6.225195
MaF11	5	1006	2.452302+	2.442642=	2.449517+	2.437614=	2.421567−	2.420182−	2.448075+	2.439245
	10	1066	17.482878=	17.484652=	17.642898+	17.590201+	17.552494+	17.534068=	17.512623=	17.418106
MaF12	5	539	1.790848=	1.791929=	1.790169=	1.836252=	1.808378=	1.817218=	1.793968=	1.785985
	10	478	13.241563=	13.288255=	13.235779=	13.19385=	13.303345=	13.313089=	13.346762=	13.366043
MaF13	5	871	0.218946=	0.218760=	0.225984=	0.233687+	0.216365=	0.223561=	0.221912=	0.220197
	10	921	0.164408=	0.162449=	0.161884=	0.155932=	0.095460−	0.121185−	0.151681=	0.189759
Total (+/=/−)			02/29/01	01/31/00	04/26/02	03/26/03	06/23/03	02/27/03	05/25/02 of 32 probs.	

8.3.1 Analysis of IP2 Operator

This subsection entails the comparative performance of seven IP2-ML variants versus IP2-RF, on both multi-objective (convergence-hard) and many-objective test instances. The corresponding hypervolume comparison is presented in Tables 8.2 and 8.3, respectively. The dominant trend in both, Tables 8.2 and 8.3, is that even if the ML method of the underlying IP2 operator is changed, the performance (in terms of solution quality) largely remains statistically equivalent (with a few exceptions, discussed later). This indicates that the originally proposed IP2 operator is reasonable, robust, and its performance may neither drastically deteriorate nor improve with a mere change in the underlying ML method.

For further insights into the results reported in Tables 8.2 and 8.3, the notion of Hypervolume factor is proposed for each ML method in conjunction with the multi-objective and many-objective test suite. For instance, in Table 8.2, intersection of Column 4 and last row with the $+/=/-$ entries as 01/18/02 suggests that for the multi-objective test suite comprising 21 problem instances, kNN performed statistically better in one, equivalent in 18, and worse in two. In general, for both Tables 8.2 and 8.3, if the performance $(+/=/-)$ of any alternative ML method vis-$\grave{a}$-vis the base method, for a multi or many-objective problem suite, is given by $B/E/W$, then it is proposed that the Hypervolume factor be defined as

$$\text{Hypervolume factor} = \frac{B - W}{B + E + W}. \tag{8.1}$$

Hence, for any alternative ML method and a specific problem suite: a positive/negative Hypervolume factor would represent the percentage instances, where it is relatively better/worse than the base method.

Furthermore, the notion of Hypervolume factor is extended to computation of Run-time factor. As a precursor, it may be noted that the run-time for only the seed underlying the median HV run for each ML method was recorded.[1] Their relative summary formatted as H/L in Table 8.4 indicates the number of times an ML method has a higher/lower run-time compared to the base method. Given this, it is proposed that the Run-time factor be defined as

$$\text{Run-time factor} = \frac{H - L}{H + L}. \tag{8.2}$$

Hence, for any alternative ML method and a specific problem suite: a positive/negative Run-time factor would represent the percentage instances, where it is relatively worse/better than the base method in terms of run-time.

[1] For this chapter, the 31 seed runs were executed in parallel to save the overall run-time, given which the exact run-time for each seed was not traceable. Hence, for run-time estimate, only the seed corresponding to the median hypervolume was executed again.

Table 8.4 Run-time comparison for IP2 operator, with eight different ML methods, on multi- and many-objective problems. Here, RF is the base method for the pairwise comparisons. The entries in the format H/L indicate the number of times an ML method has a higher/lower run-time compared to the base method (RF)

	ExTree	SVR	ENet	Ridge	LR	XGB	*k*NN
Multi-objective	07/14	06/15	06/15	05/16	05/16	20/01	05/16
Many-objective	06/26	05/27	04/28	04/28	08/24	14/18	08/24

8.3.2 Analysis of IP3 Operator

This subsection entails the comparative performance of seven IP2-ML variants versus IP3-kNN, on multi-objective (diversity-hard) and many-objective test instances. The corresponding hypervolume comparison is presented in Tables 8.5 and 8.6, respectively. The dominant trend in both, Tables 8.5 and 8.6, is that even if the ML method of the underlying IP3 operator is changed, the performance (in terms of solution quality) largely remains statistically equivalent (with a few exceptions, discussed later). This indicates that the originally proposed IP3 operator is reasonably robust, and its performance may neither drastically deteriorate nor improve with a mere change in the underlying ML method. Furthermore, the run-time summary for the IP3 operator has been presented in Table 8.7.

8.3.3 Trade-off Analysis on IP2/IP3 Operator

In the wake of the results presented in Sects. 8.3.1 and 8.3.2, the performance of different ML methods at the level of multi- and many-objective problem suites is captured in Figs. 8.1 and 8.2. Understandably, RF being the base method for IP2 occupies the origin in Fig. 8.1. Similarly, kNN being the base method for IP3 occupies the origin in Fig. 8.2. Importantly, if any alternative ML method was to dominate the base method in case of IP2 or IP3, in terms of both HV and run-time, then it should occupy the fourth quadrant. However, such occurrences are not frequent, as highlighted below

- For the IP2 operator applied to multi-objective suite: SVR and LR outperform the base RF, and seem to offer slightly better HV in reasonably lower run-time.
- For the IP2 operator applied to many-objective suite: XGB, kNN, ExTree, SVR, and Ridge seem to offer only a marginally better HV than the base RF, but in reducing order of run-time.
- For the IP3 operator applied to multi-objective suite: the base kNN seems to be the best choice, as it outperforms all other alternative ML methods.
- For the IP3 operator applied to many-objective suite: Ridge, ExTree, LR, and ENet offer insignificantly better HV, but in reducing order of run-time.

Table 8.5 HV-based comparison of IP3 with eight different ML methods on diversity-hard multi-objective problems, where *k*NN is the base for pairwise comparisons. The symbols '+', '=', and '–' indicate whether the corresponding ML method is statistically better, equivalent, or worse, than *k*NN. Here, $\hat{t}_{\text{TM}}$ denotes the average of the termination generations for 31 runs of the base *k*NN

Problem	M	$\hat{t}_{\text{TM}}$	RF	ExTree	SVR	ENet	Ridge	LR	XGB	**kNN**
CIBN1	2	1367	0.436266–	0.371431–	0.461983–	0.358032–	0.503585+	0.505845+	0.400948–	0.477499
CIBN2	2	737	0.667002–	0.663127–	0.667823–	0.668287–	0.650507–	0.668336–	0.662662–	0.669136
CIBN3	2	924	0.213839–	0.214003–	0.216013–	0.217643–	0.21647–	0.220906+	0.213819–	0.219070
CIBN4	3	429	0.922104=	0.921348=	0.914271=	0.965414+	0.935523+	0.990024+	0.914102=	0.917953
CIBN5	3	293	0.629754=	0.629755=	0.629446=	0.62982=	0.630226=	0.629558=	0.629682=	0.629438
DASCMOP1	2	1993	0.091597–	0.091364–	0.092969–	0.311539–	0.310390–	0.093046–	0.092180–	0.318945
DASCMOP2	2	1737	0.419335–	0.417992–	0.425151–	0.553743–	0.481398–	0.424107–	0.418823–	0.632949
DASCMOP3	2	1592	0.389035=	0.391774=	0.390234=	0.396553=	0.392023=	0.402481=	0.393318=	0.391774
DASCMOP4	2	2004	0.337011=	0.336996=	0.337023=	0.336977=	0.337006=	0.336929=	0.337031=	0.336944
DASCMOP5	2	2098	0.672736=	0.672932+	0.672803+	0.672756+	0.672795+	0.672842+	0.672729+	0.672536
DASCMOP6	2	2364	0.574657=	0.549858=	0.574768=	0.574817=	0.574812=	0.574867=	0.574839=	0.574711
DASCMOP7	3	1725	1.025381=	1.027783=	1.027030=	1.027350=	1.028054=	1.027964=	1.026924=	1.025903
DASCMOP8	3	1676	0.659817=	0.647717=	0.645259=	0.65963=	0.63037=	0.658968=	0.634491=	0.661457
DASCMOP9	3	1524	0.645847–	0.641343–	0.643265–	0.645213–	0.64766=	0.645021–	0.634619–	0.647879

(continued)

Table 8.5 (continued)

Problem	M	$\hat{t}_{\text{TM}}$	RF	ExTree	SVR	ENet	Ridge	LR	XGB	**kNN**
MW1	2	1086	0.415419=	0.415482=	0.415328=	0.415469=	0.415332=	0.415345=	0.415356=	0.415435
MW2	2	832	0.482916=	0.483006=	0.482914=	0.482974=	0.482776=	0.482936=	0.482986=	0.483077
MW3	2	862	0.470025=	0.469952=	0.469801=	0.469976=	0.470081=	0.469904=	0.469868=	0.470079
MW4	3	751	1.041446=	1.041393=	1.041398=	1.041431=	1.041442=	1.041366=	1.041479=	1.041415
MW5	2	1682	0.197842=	0.199818=	0.197719=	0.198334=	0.198288=	0.198584=	0.199581=	0.196036
MW6	2	1211	0.066170=	0.066186=	0.066179=	0.066162=	0.066157=	0.066187=	0.066089=	0.066154
MW7	2	914	0.148623=	0.148983=	0.148656=	0.148355=	0.148806=	0.148719=	0.148764=	0.148740
MW8	3	688	0.626837=	0.626173=	0.626974=	0.626700=	0.627159=	0.626572=	0.627297=	0.627103
MW9	2	1031	0.295729=	0.295494=	0.295335=	0.295310=	0.296098=	0.295089=	0.295587=	0.295640
MW10	2	1072	0.263741=	0.263107=	0.263266=	0.263692=	0.263353=	0.263541=	0.263203=	0.263450
MW11	2	964	0.257090+	0.266341+	0.265250+	0.266780+	0.266616+	0.208405+	0.265514+	0.000000
MW12	2	1100	0.570850+	0.570827+	0.570720+	0.570823+	0.570840+	0.570745+	0.570859+	0.430485
MW13	2	935	0.329000+	0.329813+	0.329233+	0.329400+	0.329011+	0.329942+	0.328646+	0.000000
MW14	3	902	0.152975+	0.151108+	0.153208+	0.155175+	0.161203+	0.156183+	0.151444+	0.000000
Total (+/=/−) ⟶			04/18/06	05/17/06	05/17/06	06/16/06	07/17/04	08/16/04	05/17/06	of 28 probs.

Table 8.6 HV-based comparison of IP3 with eight different ML methods on many-objective problems, where *k*NN is the base for pairwise comparisons. The symbols '+', '=', and '–' indicate whether the corresponding ML method is statistically better, equivalent, or worse, than *k*NN. Here, $\hat{t}_{\mathrm{TM}}$ denotes the average of the termination generations for 31 runs of the base *k*NN (taken from [1])

Problem	M	$\hat{t}_{\mathrm{TM}}$	RF	ExTree	SVR	ENet	Ridge	LR	XGB	**kNN**
DTLZ1	5	1402	2.486749=	2.486892=	2.486839=	2.486717=	2.486795=	2.486833=	2.486964=	2.486885
	10	2023	17.757703=	17.757702=	17.757693=	17.757704=	17.757690=	17.757684=	17.757683=	17.757696
DTLZ2	5	866	2.171945=	2.172083+	2.171872=	2.172031+	2.172102+	2.172082+	2.172134+	2.171782
	10	757	17.664176=	17.664258=	17.664655=	17.664885=	17.664505=	17.664448=	17.664356=	17.664525
DTLZ3	5	1198	2.083931=	2.079724=	2.049543–	2.057957–	2.033053–	2.055357–	2.099911=	2.09651
	10	1851	17.659759=	17.660024=	17.658199–	17.658355=	17.656276–	17.659139=	17.663084=	17.660877
DTLZ4	5	915	2.173070–	2.173172=	2.173212=	2.173206=	2.173125=	2.173185=	2.173211=	2.173235
	10	800	17.676593–	17.67686=	17.67699=	17.676891=	17.676909=	17.676767=	17.677003=	17.676834
MaF1	5	853	0.059759=	0.059340=	0.059473=	0.059903=	0.059967=	0.059876=	0.05956=	0.060380
	10	1059	0.002074+	0.002155+	0.001959=	0.001957=	0.002004=	0.002149+	0.002159+	0.002000
MaF2	5	361	0.999336=	0.997776=	0.999999=	1.000740+	1.002444+	0.998881=	0.999796=	0.995748
	10	484	7.877194=	7.865008=	7.850738=	7.883851+	7.827113=	7.833383=	7.827931=	7.830099
MaF3	5	2416	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320=	2.488320
	10	3658	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727
MaF4	5	649	2.480958=	2.469060=	2.482613=	2.481608=	2.475904=	2.479125=	2.482338=	2.477745
	10	1094	17.529054=	17.534092=	17.541146=	17.497915–	17.565631=	17.535149=	17.542442=	17.557395
MaF5	5	1113	2.487942=	2.487944=	2.487942=	2.487946=	2.487946=	2.487943=	2.487947=	2.487944
	10	1197	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727=	17.757727

(continued)

Table 8.6 (continued)

Problem	M	$\hat{t}_{\text{TM}}$	RF	ExTree	SVR	ENet	Ridge	LR	XGB	**kNN**
MaF7	5	1074	0.968845=	0.972045=	0.967674=	0.969739=	0.968382=	0.968788=	0.969065=	0.969353
	10	734	7.598982=	7.578182=	7.590612=	7.585287=	7.615941=	7.594704=	7.592734=	7.584186
MaF8	5	1361	0.000551=	0.000477=	0.000426–	0.000450–	0.000410-	0.000470=	0.000449–	0.000495
	10	1046	0.000065=	0.000067=	0.000076+	0.000077+	0.000074+	0.000069=	0.000076+	0.000068
MaF9	5	1820	0.028026=	0.028181=	0.025498–	0.026383–	0.027629=	0.028588=	0.026612–	0.028363
	10	725	0.000065=	0.000080=	0.000099=	0.000056=	0.000035=	0.000061=	0.000072=	0.000080
MaF10	5	1069	0.964888=	0.957266=	0.967990+	0.965795=	0.968337+	0.959917=	0.964241+	0.961056
	10	783	6.060485=	5.912519=	5.959022=	6.296865+	5.875247=	5.879183=	6.002479=	5.908347
MaF11	5	960	2.448849=	2.449593=	2.450035=	2.449182=	2.449345=	2.449839=	2.450457=	2.446909
	10	1231	17.561243=	17.571667=	17.567886=	17.551676=	17.550847=	17.558191=	17.549476=	17.557456
MaF12	5	546	1.782436=	1.784627=	1.785285=	1.782553=	1.783997=	1.78398=	1.785365=	1.784192
	10	481	13.118814=	13.118862=	13.130149=	13.165706=	13.122503=	13.077448=	13.243644+	13.041259
MaF13	5	911	0.246188=	0.246621=	0.237909=	0.242349=	0.236794=	0.238086=	0.241078=	0.236727
	10	904	0.237637=	0.236821=	0.230929=	0.192100=	0.225026=	0.229397=	0.235592=	0.226286
Total (+/=/–) $\longrightarrow$			01/29/02	02/30/00	02/26/04	05/23/04	04/25/03	02/29/01	05/25/02	of 32 probs.

Table 8.7 Run-time comparison for IP3 operator, with eight different ML methods, on multi- and many-objective problems. Here, *k*NN is the base method for the pairwise comparisons. The entries in the format H/L indicate the number of times an ML method has a higher/lower run-time compared to the base method (*k*NN)

	RF	ExTree	SVR	ENet	Ridge	LR	XGB
Multi-objective	27/01	27/01	23/05	24/04	26/02	25/03	28/00
Many-objective	27/05	12/20	17/15	09/23	14/18	10/22	29/03

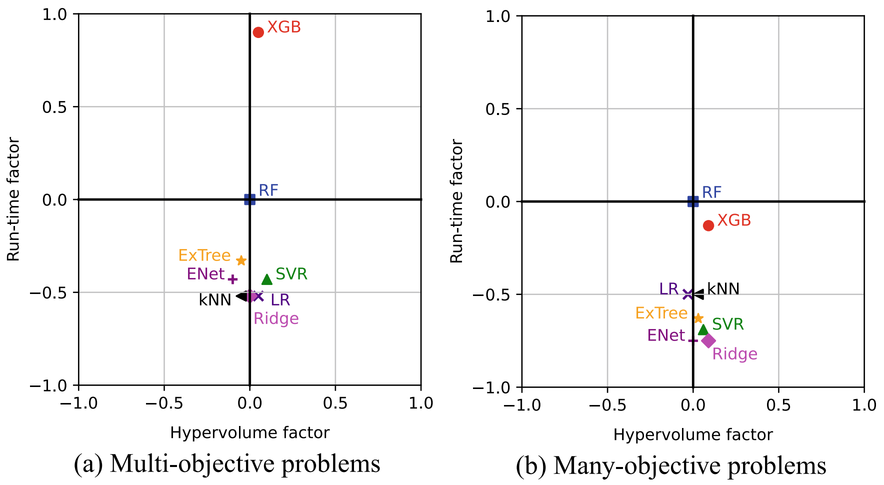


Fig. 8.1 Plots comparing HV factor and Run-time factor across the multi-objective and many-objective problem suite for the IP2 operator

Overall, if one winner has to be picked across all scenarios, then *k*NN can be inferred as the one.

The presented results also endorse the known characteristics of some of the considered ML methods, as highlighted below. It can be observed that the considered linear ML models, namely LR, Ridge, and ENet have a comparable performance, particularly in terms of the HV measures. Within the family of trees, RF has a higher run-time than ExTree. This is consistent with the expectation, since RF chooses the optimal split, while ExTree relies on random split, saving some time. Overall, the boosting algorithm, namely XGB, is seen to have a higher run-time compared to the other ML methods. This could be attributed to the fact that during each split finding process, XGB iterates over all entries in the input data, making the process slower.

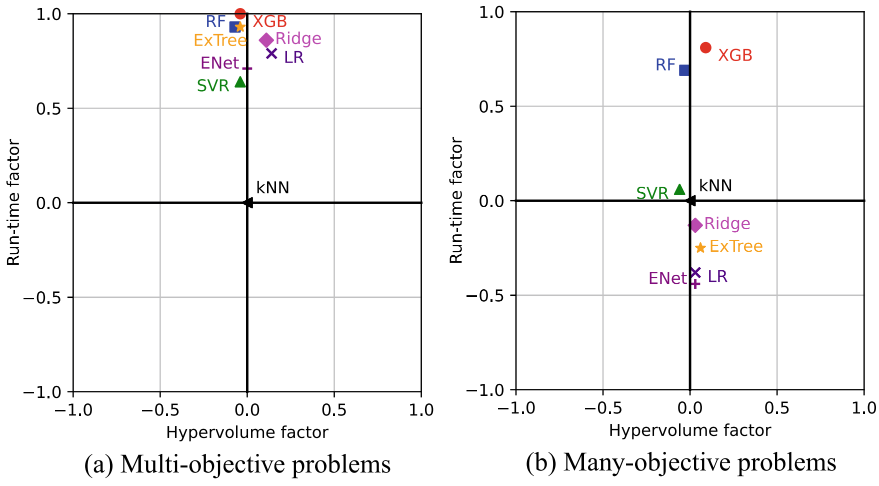


Fig. 8.2 Plots comparing HV factor and Run-time factor across the multi-objective and many-objective problem suite for the IP3 operator

8.3.4 *Results with UIP Operator*

As concluded above in Sect. 8.3.3, if a single ML method had to be selected for both IP2 and IP3 operators, *k*NN is the choice. Since the UIP operator employs both IP2 and IP3 operators, it is imperative to investigate how the use of *k*NN for both IP2 and IP3 operators affects the performance of UIP operator, given that the UIP operator (as discussed in Chap. 7), uses RF for IP2 and *k*NN for IP3. Hence, this section compares the performance of the original and new UIP operators, given as

- UIP (original): RF for the IP2 and *k*NN for the IP3 operator.
- UIP (new): *k*NN for both IP2 and IP3 operators.

The median hypervolume values for both original and new UIP variants, along with their statistical comparison, are provided in Tables 8.8, 8.9, and 8.10, for multi- and many-objective problems. As can be observed from these tables:

- The new UIP performs statistically better in 1, and statistically equivalent in the rest 20 convergence-hard multi-objective problems, compared to the original UIP.
- The new UIP performs statistically equivalent in all 28 diversity-hard multi-objective problems, compared to the original UIP.
- The new UIP performs statistically better in 1, statistically equivalent in 30, and statistically worse in 1 many-objective problems, compared to the original UIP.

Table 8.8 HV-based comparison of UIP (original, with RF for IP2 and *k*NN for IP3), with the UIP (new, with *k*NN for both IP2 and IP3) on convergence-hard multi-objective problems. The symbols '+', '=', and '–' indicate whether the corresponding method is statistically better, equivalent, or worse. Here, $\hat{t}_{\text{TM}}$ denotes the average of the termination generations for 31 runs of UIP (new)

Problem	M	t_{TM}	UIP (original)	UIP (new)
ZDT1	2	1200	0.681859=	0.681859
ZDT2	2	1270	0.348794=	0.348794
ZDT3	2	1002	1.068446=	1.068452
ZDT4	2	1763	0.681860=	0.681860
ZDT6	2	1781	0.334000=	0.333027
DTLZ1	3	1431	1.222117=	1.222344
DTLZ2	3	989	0.667332=	0.667338
DTLZ3	3	1548	0.658096=	0.661404
DTLZ4	3	2262	0.667356=	0.667354
MaF1	3	601	0.235747–	0.236504
MaF2	3	468	0.396482=	0.396862
MaF3	3	2154	1.239854=	1.239862
MaF4	3	1297	0.628690=	0.631397
MaF5	3	2027	1.227609=	1.227626
MaF7	3	1218	0.376031=	0.375771
MaF8	3	1515	0.466266=	0.466419
MaF9	3	1163	0.626760=	0.626764
MaF10	3	935	0.535141=	0.534181
MaF11	3	992	0.980350=	0.980128
MaF12	3	741	0.605585=	0.607628
MaF13	3	1008	0.369809=	0.369025
Total (+/=/–) $\longrightarrow$			00/20/01 of 21 probs.	

Overall, it is fair to infer that the new UIP performs almost equivalent to the original UIP, in terms of performance (hypervolume metric). Hence, changing the underlying ML method of the IP2 operator (from RF to *k*NN) has an insignificant effect on the performance of the UIP operator. This indicates that the originally proposed UIP operator is reasonably robust, and its performance may neither drastically deteriorate nor improve with a mere change in the underlying ML method. Notably, the use of *k*NN still has its edge over the use of RF, in terms of run-time, as can be observed from Table 8.4.

Table 8.9 HV-based comparison of UIP (original, with RF for IP2 and *k*NN for IP3), with the UIP (new, with *k*NN for both IP2 and IP3) on diversity-hard multi-objective problems. The symbols '+', '=', and '–' indicate whether the corresponding method is statistically better, equivalent, or worse. Here, $\hat{t}_{\mathrm{TM}}$ denotes the average of the termination generations for 31 runs of UIP (new)

Problem	M	t_{TM}	UIP (original)	UIP (new)
CIBN1	2	1421	0.480243=	0.486227
CIBN2	2	748	0.669060=	0.669101
CIBN3	2	943	0.224022=	0.223506
CIBN4	3	427	0.916890=	0.919279
CIBN5	3	296	0.630038=	0.629572
DASCMOP1	2	2028	0.321026=	0.320843
DASCMOP2	2	1962	0.651950=	0.654912
DASCMOP3	2	1567	0.391939=	0.392005
DASCMOP4	2	2033	0.336920=	0.336910
DASCMOP5	2	2115	0.672642=	0.672626
DASCMOP6	2	2473	0.574932=	0.574911
DASCMOP7	3	1811	1.028027=	1.028283
DASCMOP8	3	1711	0.661391=	0.653839
DASCMOP9	3	1574	0.647210=	0.647831
MW1	2	1061	0.415373=	0.415303
MW2	2	850	0.483088=	0.482936
MW3	2	966	0.470011=	0.470043
MW4	3	742	1.041414=	1.041342
MW5	2	1658	0.199164=	0.197663
MW6	2	1225	0.302555=	0.302521
MW7	2	897	0.366408=	0.366482
MW8	3	724	0.626689=	0.625977
MW9	2	1046	0.295656=	0.295583
MW10	2	1069	0.247448=	0.247404
MW11	2	974	0.264916=	0.255534
MW12	2	1079	0.570711=	0.570743
MW13	2	960	0.329176=	0.327977
MW14	3	838	0.154104=	0.156726
Total (+/=/–) $\longrightarrow$			00/28/00	of 28 probs.

Table 8.10 HV-based comparison of UIP (original, with RF for IP2 and *k*NN for IP3), with the UIP (new, with *k*NN for both IP2 and IP3) on many-objective problems. The symbols '+', '=', and '–' indicate whether the corresponding method is statistically better, equivalent, or worse. Here, t_{TM} denotes the average of the termination generations for 31 runs of UIP (new)

Problem	$M = 5$			$M = 10$		
	t_{TM}	UIP (original)	UIP (new)	t_{TM}	UIP (original)	UIP (new)
DTLZ1	1476	2.488320=	2.488320	2011	17.757727=	17.757727
DTLZ2	866	2.172711=	2.172758	769	17.670674=	17.670376
DTLZ3	1136	2.488320=	2.488320	1806	17.757727=	17.757727
DTLZ4	1097	2.173476=	2.173154	804	17.676855=	17.676941
MaF1	830	0.058752=	0.058755	762	0.002110–	0.002252
MaF2	356	1.000493+	0.997510	487	7.869492=	7.844157
MaF3	1971	2.488320=	2.488320	3654	17.757727=	17.757727
MaF4	650	0.214196=	0.154689	971	0.061585=	0.061113
MaF5	1655	2.487940=	2.487940	1178	17.757700=	17.757700
MaF7	1079	0.969948=	0.971369	834	7.688220=	7.669310
MaF8	1741	7.196114=	7.210096	1295	67.994300=	66.559200
MaF9	1015	0.027374=	0.027867	710	0.000115=	0.000137
MaF10	997	0.977029=	0.973406	903	6.008000=	6.068800
MaF11	1012	2.440631=	2.436644	1554	17.613900=	17.620000
MaF12	547	1.772635=	1.775227	489	13.038200=	13.004600
MaF13	752	0.233392=	0.231456	1092	0.173370=	0.170714
Total (+/=/–)		01/15/00			00/15/01	

8.4 Summary

This chapter has sought to analyze as to how sensitive the existing Innovized Progress operators (IP2 for convergence improvement and IP3 for diversity improvement) are to the choice of the underlying ML method. In that, the key concern was to investigate if by changing the underlying ML methods, the performance of these operators may drastically deteriorate, or if it could be significantly improved. Exhaustive experiments based on eight ML methods, tested against 49 multi-objective and 32 many-objective test instances, suggest that both the existing IP2 and IP3 operators are reasonably robust, and not too sensitive to the choice of underlying ML method. However, if one ML method is to be recommended for use, within the gambit of the existing IP2 and IP3 operators, considered test suite, and chosen ML methods, then *k*NN could be considered as the best performing method. The justification is that even though the hypervolume offered by it may be comparable or just marginally better than the alternative ML methods, its run-time is relatively lower. Hence, when *k*NN is used as the underlying ML method for the IP2 and IP3 operators jointly leading up to the UIP operator, the advantage in terms of the gain in hypervolume is insignificant compared to the the base UIP presented in Chap. 7 (with RF underlying the IP2 operator, and kNN underlying the IP3 operator). In accomplishing the above investigation, this chapter relied on a systematic methodology to investigate the trade-off associated with different ML methods in terms of their potential for performance enhancement of Evolutionary Multi- and Many-objective Optimization algorithms vis-à-vis the associated computational cost.

References

1. Bhasin, D., Swami, S., Sharma, S., Sah, S., Saxena, D.K., Deb, K.: Investigating innovized progress operators with different machine learning methods. In: Emmerich, M., Deutz, A., Wang, H., Kononova, A.V., Naujoks, B., Li, K., Miettinen, K., Yevseyeva, I. (eds.) Evolutionary Multi-Criterion Optimization, pp. 134–146. Springer Nature Switzerland, Cham (2023)
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pp. 785–794. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785
3. Cover, T.: Estimation by the nearest neighbor rule. IEEE Trans. Inf. Theory **14**(1), 50–55 (1968). https://doi.org/10.1109/TIT.1968.1054098
4. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998)
5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Trans. Evolut. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
6. Goldstein, B.A., Polley, E.C., Briggs, F.B.S.: Random forests for genetic association studies. Stat. Appl. Genet. Mol. Biol. **10**(1) (2011). https://doi.org/10.2202/1544-6115.1691
7. Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970). https://doi.org/10.1080/00401706.1970.10488634
8. Hussain, J.N.: High dimensional data challenges in estimating multiple linear regression. J. Phys.: Conf. Ser. **1591**(1), 012,035 (2020). https://doi.org/10.1088/1742-6596/1591/1/012035
9. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B: Stat. Methodol. **67**(2), 301–320 (2005). https://doi.org/10.1111/j.1467-9868.2005.00503.x

Chapter 9
Learning to Analyze the Pareto-Optimal Front

As mentioned in the previous chapters, evolutionary multi- and many-objective optimization algorithms (EMâOAs) attempt to find a set of well-converged and well-diversified solutions to approximate the true Pareto front (PF). In general, a uniform distribution of solutions across PF is desired. However, this cannot be guaranteed due to the stochasticity involved in EMâOAs. In a contrasting scenario, even a biased distribution of solutions across PF, with a higher concentration of solutions in specific parts of PF, may be desired by the decision maker for a subsequent multi-criterion decision-making (MCDM) task. indexMulti-criterion decision-making (MCDM) To meet such requirements, this chapter presents a machine learning (ML)-based approach, which treats a given PF-approximation as input and trains an ML model to capture the relationship between pseudo-weight vectors derived from the objective vectors in the PF-approximation (F in $\mathcal{Z}$), and their underlying variable vectors (X in $\mathcal{X}$). Subsequently, the trained ML model is applied to predict the solution's X vector for any desired pseudo-weight vector. In other words, the trained ML model is used to create new non-dominated solutions in any desired region of the obtained PF-approximation. Such new solutions could be created to fill apparent gaps in the input PF-approximation toward a more uniform distribution or to enhance the concentration of solutions as desired by the decision maker. The working and usefulness of the above post-optimality analysis basis approach have been demonstrated over several problem instances. However, this approach also has the potential to be integrated within an EMâOA to arrive at the desired distribution.

In principle, the proposed approach, summarized above, conforms with any conventional ML approach in which input–output relationships are learned from the training-dataset, and the resulting ML model is used to predict the output from an unseen given input. In the current context, the training-dataset is based on non-dominated solutions obtained by the EMâOA, in which the input is a unique indicator of a solution (in the objective space or $\mathcal{Z}$ space) and the output is the X vector of the solution. In that scenario, the pseudo-weight vectors [2], denoted by W, have been used as the unique indicator. An advantage of using a pseudo-weight vector is that each component lies in the [0, 1] range and therefore does not require normalization. Once the ML model is trained, any new pseudo-weight vector can be used as input, to find the corresponding solution's X vector. Subsequently, its F vector can be computed using the objective formulations stated in the given multi- or many-objective optimization problem (MOP/MaOP).

The remainder of this chapter is organized as follows. The ML-based procedure for the post-optimal *PF* analysis is described in Sect. 9.1. The results on multi- and many-objective test problems (constrained and unconstrained) and real-world problems are presented in Sect. 9.2. Finally, Sect. 9.4 summarizes the findings.

9.1 ML-Based *PF* Analysis: Proposition and Validation

As discussed in previous chapters, using an EMâOA with N population members to solve an MOP/MaOP can lead to N potentially non-dominated solutions $X^{(k)} \in \mathbb{R}^n$, for $k = 1, 2, \ldots, N$, along with their respective objective vectors $F(X^{(k)}) \in \mathbb{R}^M$, for $k = 1, 2, \ldots, N$. In the $\mathcal{Z}$ space, these objective vectors are referred to as the *PF*-approximation. From the position of each objective vector on the *PF*-approximation, the corresponding pseudo-weight vector $W^{(k)} = \left(w_1^{(k)}, w_2^{(k)}, \ldots, w_M^{(k)}\right)$; $W^{(k)} \in \mathbb{R}^M$ can be computed as follows [2]:

$$w_i^{(k)} = \frac{\left(f_i^{\max} - f_i(X^{(k)})\right) \Big/ \left(f_i^{\max} - f_i^{\min}\right)}{\sum_{j=1}^{M} \left(f_j^{\max} - f_j(X^{(k)})\right) \Big/ \left(f_j^{\max} - f_j^{\min}\right)}, \tag{9.1}$$

where $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values in the ith objective, among all the non-dominated solutions in the *PF*-approximation. The pseudo-weight (W) vector for any solution can be viewed as a representative identity on the obtained *PF*-approximation. For example, given a two-objective problem, the best solution in f_1 would correspond to a vector W of $(1, 0)$, which means that the solution is of 100% importance with respect to f_1, and of no importance with respect to f_2.

Clearly, Eq. 9.1 indicates that the W vectors are linearly derived from the F vectors. If the solutions are sorted according to their F vectors, the respective W vectors also are sorted accordingly. Previous studies on the concept of *innovization* (Chap. 3) have revealed that a set of Pareto-optimal solutions usually possesses certain patterns or constancy with respect to certain variables. Since a W vector is derived from an F vector that bears an exact relationship with its underlying X vector, it is fair to assume a linkage between a W vector and the corresponding X vector. In this context, this study captures the relationships between the derived W vectors and corresponding X vectors from a *PF*-approximation, using an ML method. Once these patterns, if any, are captured, the desired $\bar{W}$ vectors can be used to obtain the $\bar{X}$ vectors of new (potentially non-dominated) solutions. Figure 9.1 illustrates the

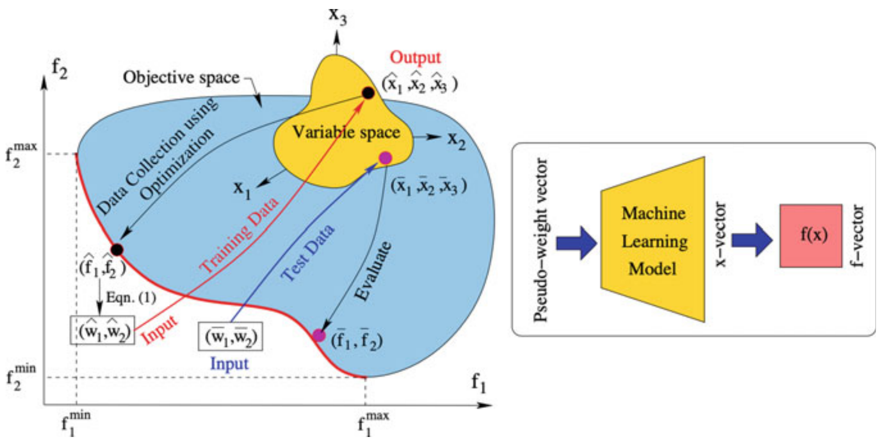


Fig. 9.1 Training data generation and test-data to create a new potentially Pareto-optimal solution (taken from [3])

underlying ML training and testing procedure, as required, which is detailed in the following steps:

Step 1: From each solution in the obtained *PF*-approximation, compute the W vectors using Eq. 9.1.

Step 2: Construct the training-dataset $[W^{(k)}, X^{(k)}]\ \forall\, k = 1, 2, \ldots, N$, with $W^{(k)}$ as input and $X^{(k)}$ as the corresponding target.

Step 3: Train an ML model using the training-dataset.

Step 4: Use the trained ML model to find $\bar{X}$ for any specific desired $\bar{W}$ vector. Then, compute $\bar{F}$ from $\bar{X}$ using the original objectives' formulation, where $\bar{F} = F(\bar{X})$.

The resulting trained ML model can then be used for several different tasks, as summarized below.

Task 1: Gap-filling: The trained ML model can be used to find new and potentially non-dominated solutions, so that the apparent gaps in the obtained *PF*-approximation can be filled. This could be realized by creating new W vectors in those gaps and then applying the trained ML model to create new (non-dominated) solutions.

Task 2: Gap-validation: The trained ML model can be used to determine whether or not an apparent gap in the *PF*-approximation is a natural gap in the true *PF*. This could be realized by first creating new W vectors in the apparent gaps; obtaining the corresponding X vectors through the trained ML model; evaluating these X vectors to obtain the corresponding F vectors; and investigating whether or not these F vectors are dominated by any solution in the existing *PF*-approximation. If any such F vector stands dominated, it implies that the apparent gap for which the W vector was created happens to be a natural gap in the true PF.

Task 3: Preferred solution-density propagation: The trained ML model can be used to populate new non-dominated solutions in any preferred region of the PF-approximation, toward ease of visualization and decision-making. This could be achieved by creating suitable W vectors in the region of interest and then using the trained ML model to find new and potentially non-dominated solutions in that region.

Task 4: Efficient offspring creation: The trained ML model can be used within an EMâOA run, to create new offspring solutions. This could be realized by first training an ML model using the non-dominated solutions from the parent population; selecting the W vectors in the less dense areas; and applying the trained ML model to create new offspring solutions (one solution per W vector).

The scope of this proof-of-concept study has been restricted to the first three tasks mentioned above since the last task requires the development of a new EMâOA.

9.1.1 Chosen ML Methods

In this section, details of the ML methods utilized in this study are discussed. The problem statement at hand is the task of predicting an n-dimensional X vector from a given M-dimensional W vector. Since, in general, $M \ll n$, developing an accurate ML model with few input parameters and a large number of output parameters is a challenging task. Two different ML methods have been used for this task, namely deep neural network (DNN) [6] and Gaussian process regression (GPR) [13]. In both the DNN-based and the GPR-based approaches, the X vectors of the solutions are normalized to zero mean and unit variance a priori, as required by the ML methods. As mentioned at the beginning of this chapter, the W vectors are already within the [0, 1] range and therefore do not require normalization.

For each problem considered here, PF is approximated using NSGA-III, with the population size given by $N = (110M + 10)$. These population sizes were selected based on a trial-and-error study. From these N solutions, randomly selected $100M$ solutions are used as the training-dataset, and the other $10M$ solutions are used as the test-dataset. Subsequently, the remaining 10 solutions are used as the validation-dataset as required by the DNN, but are discarded in the case of GPR, since GPR does not require a validation step. The validation-dataset is uniformly sampled from the PF-approximation for appropriate model selection in the case of DNN. Notably, the same population size has been used for both to maintain the similarity of the training-dataset size. Furthermore, this approach is not conditional on the source of the training-dataset. Therefore, the training-dataset obtained from the non-dominated solutions of the PF-approximation can be replaced by the non-dominated solutions at the end of any intermediate generation of an EMâOA run.

9.1.1.1 DNN-Based Approach

For the DNN-based approach, multi-layer perceptrons have been implemented using the PyTorch library [12], with W vectors as inputs and their underlying X vectors as respective targets, and the corresponding hyper-parameters are tuned using Optuna [1]. Due to the proof-of-concept nature of the study, DNNs with hidden layers ranging from 1 to 6 are used with 'ReLU' as activation function, and 'Adam' [9] as optimizer. The complexity of the DNN architecture and granularity of hyper-parameters can be further increased for handling more complicated problems.

9.1.1.2 GPR-Based Approach

An approach similar to surrogate modeling is used to train the GPR model. In that approach, each component of the target—that is, $x_i \in X$—is modeled independently. A diverse set of kernels, mean functions, and other hyper-parameters is considered in a grid-search for finding the most suitable setting.

9.1.2 Handling Variable Bounds and Constraints

In any MOP/MaOP, the variables are usually restricted within the pre-specified lower and upper bounds. Since a DNN or GPR model usually does not restrict its output values (X vector) within any bounds by default, the resulting output values for any test-dataset can be outside of those bounds, if specific steps are not taken. Hence, the target values in the training-dataset have also been normalized. Subsequently, the output X vector from the trained ML model is denormalized and clipped to within the specified lower and upper bounds.

Furthermore, constraint satisfaction is also a strict requirement in solving a constrained MOP/MaOP. In such a scenario, the resulting X vectors from the trained ML model may not guarantee the constraint satisfaction by default. Hence, the infeasible solutions (if created by the ML model) are simply discarded; but a more sophisticated constraint handling method can be used during the ML training. For instance, the constraint value of each constraint can be included as an additional output in the training-dataset. During testing, if any W vector (input) produces a positive constraint value (meaning a constraint violation), the solution can simply be ignored. To implement such constraint handling, the training-dataset must contain some infeasible solutions that are non-dominated to the feasible non-dominated solution set, to effect a better learning. In this study, such a tailored constraint handling approach has not been included.

9.1.3 Test Suite

For validation of ML models and analysis of different tasks (mentioned earlier in Sect. 9.1), several test and real-world problems have been used, including the following:

- Two-objective unconstrained ZDT [16], and constrained OSY [11] and BNH [15] problems.
- Three-objective unconstrained DTLZ [4] and WFG [7] problems.
- Many-objective unconstrained DTLZ [4] and constrained C-DTLZ [8] problems.
- Real-world Carside [5] and Crashworthiness [10] problems.

9.2 Validation of ML Models

This section focuses on the validation of the trained ML model. Recall that given a PF-approximation with $N = 110M + 10$ solutions, the training-data was constituted by randomly chosen $100M$ solutions. Of the remaining ones, the randomly picked $10M$ solutions constitute the test-data and are referred to as *Random* test-data. Once the W vectors for these test solutions are subjected to the trained ML model (DNN or GPR-based), it yields the X vectors (referred to as model-based X vectors) which on evaluation offer the corresponding model-based F vectors. Since the original X and F vectors of these test solutions are available from the PF-approximation, the mean absolute error (MAE) between the original and model-based X vectors (normalized using the variable bounds), and the original and model-based F vectors (normalized using ideal and nadir points) can be computed and referred to as MAE_X and MAE_F, respectively.

For validation of the ML model, a range of problems is considered, and 11 independently seeded runs of NSGA-III are performed for each problem. The MAE_X resulting from each of the 11 runs can be processed, leading to their respective mean and standard deviation, namely $\mu(MAE_X)$ and $\sigma(MAE_X)$, respectively. Similarly, using the MAE_F values over the 11 runs, the corresponding $\mu(MAE_F)$ and $\sigma(MAE_F)$ can be obtained. Table 9.1 presents the above indicators for a range of two-objective problems, for both DNN- and GPR-based approaches. Evidently:

- Low values of $\mu(MAE_X)$ and $\sigma(MAE_X)$ indicate that the actual and the ML model-based approximated X vectors are reasonably close to each other. The same applies to the actual and approximated F vectors. These observations validate the underlying ML models. A visual evidence for this claim is presented for a sample problem, namely ZDT1, in Fig. 9.2.
- Compared to the DNN-based approach, the GPR-based approach offers lower values for each of $\mu(MAE_X)$ and $\sigma(MAE_X)$, also $\mu(MAE_F)$ and $\sigma(MAE_F)$. Hence, the GPR-based approach is seemingly better than the DNN-based approach.

Table 9.1 Comparison of DNN and GPR, with *Random* test-data on two-objective problems

Problem	M	Model	$\mu(MAE_X)$	$\mu(MAE_F)$	$\sigma(MAE_X)$	$\sigma(MAE_F)$
ZDT1	2	DNN	4.606E-04	6.918E-03	2.575E-04	3.689E-03
		GPR	5.525E-09	3.462E-06	3.064E-09	7.511E-06
ZDT2	2	DNN	7.260E-04	1.387E-02	4.065E-04	6.786E-03
		GPR	1.398E-05	4.200E-04	1.946E-05	6.093E-04
ZDT3	2	DNN	4.619E-04	1.757E-02	2.591E-04	1.434E-02
		GPR	3.659E-06	3.098E-04	4.985E-06	8.355E-04
OSY	2	DNN	7.833E-03	3.607E-02	3.323E-02	3.521E+00
		GPR	1.491E-03	1.273E-03	1.146E-02	3.407E-01

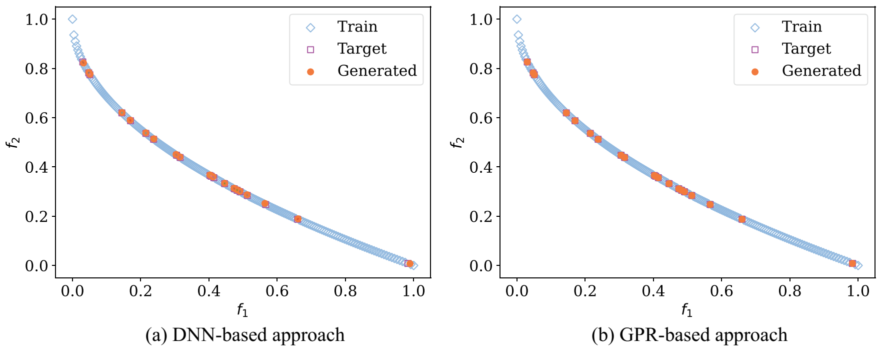


Fig. 9.2 DNN- and GPR-based approaches on *Random* test-dataset for the ZDT1 problem (taken from [3])

The scope of ML-model validation is expanded by including some two-objective (constrained), three-objective, and many-objective problems. However, for experiments, only the GPR-based approach which performed better on the unconstrained two-objective problems has been considered. The results are presented in Table 9.2 along side some self-explanatory representative plots in Figs. 9.3 and 9.4.

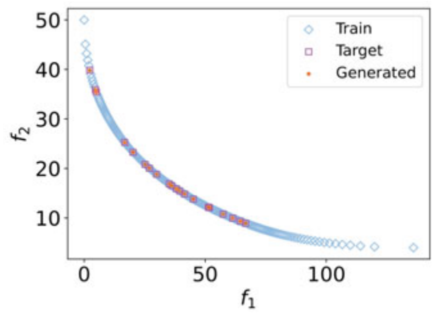
9.3 ML-Based *PF* Analysis for Different Tasks

This section presents the application of the ML-based *PF* analysis toward the first three tasks highlighted in Sect. 9.1. As a pre-requisite, the use of the different test-data types as defined below must be noted.

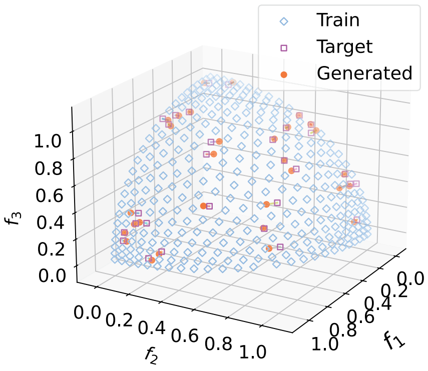
- *Edge* test-data: It implies that the test solutions representing the gap belong to one of the extreme regions of the *PF*-approximation.
- *Continuous* test-data: It implies that the test solutions representing the gap are bounded by continuous regions in the *PF*-approximation.

Table 9.2 Performance of GPR-based approach on two-objective (constrained), three-objective, and many-objective problems, for *Random* test-dataset

Problem	M	$\mu(MAE_X)$	$\mu(MAE_F)$	$\sigma(MAE_X)$	$\mu(MAE_F)$
BNH	2	1.890E-03	2.073E-04	3.004E-03	9.670E-03
DTLZ2	3	3.419E-03	1.153E-02	2.494E-03	7.168E-03
WFG2	3	4.773E-02	4.922E-02	1.220E-01	1.732E-01
Carside	3	1.171E-02	3.957E-03	9.080E-03	1.471E-02
DTLZ2	5	9.887E-03	1.382E-02	9.155E-03	8.668E-03
	10	4.582E-02	1.418E-02	3.716E-02	8.500E-03
C2-DTLZ2	5	1.020E-02	1.323E-02	1.028E-02	7.448E-03
	10	4.532E-02	1.418E-02	3.552E-02	9.490E-03



(a) BNH



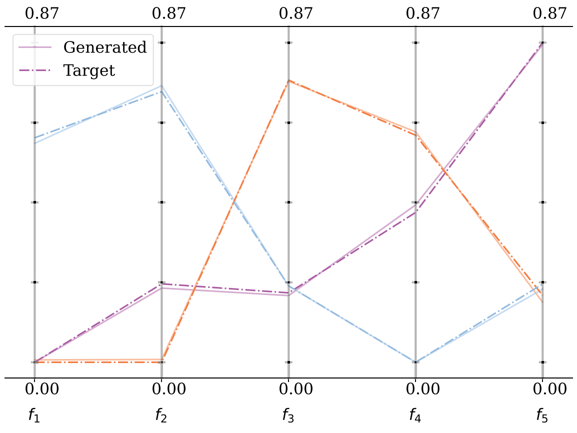
(b) DTLZ2

Fig. 9.3 GPR-based approach on *Random* test-dataset constrained two-objective (BNH) and three-objective (DTLZ2) problems (taken from [3])

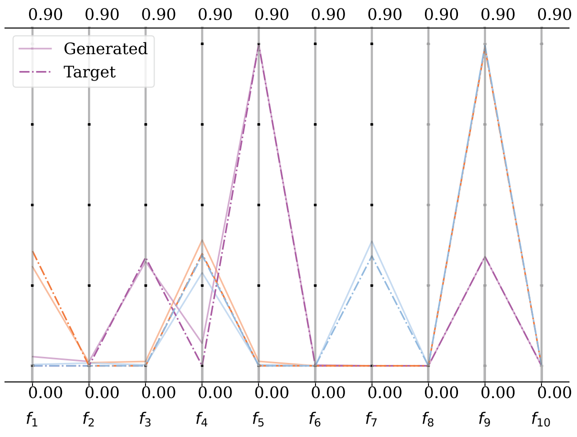
9.3.1 Task 1: Gap-Filling

This section focuses on the validation of Task 1, i.e., the gap-filling task. Unlike in Sect. 9.2 where *Random* test-data was used, here *Edge* test-data has first been used, followed by the *Continuous* test-data. The rest of the procedure is the same as used for validation of the ML model in Sect. 9.2.

Table 9.3 shows the MAE_X and MAE_F indicators on some two-objective problems with *Edge* test-data, for both DNN- and GPR-based approaches. In that, compared to the DNN-based approach, the GPR-based approach offers lower values for each of $\mu(MAE_X)$ and $\sigma(MAE_X)$, and also $\mu(MAE_F)$ and $\sigma(MAE_F)$. This implies that the GPR-based approach is better for finding edge gap solutions. For a visual representation, Fig. 9.5 shows the edge gap solutions produced using DNN- and GPR-based approaches on the ZDT1 problem. As can be observed, while both approaches seem



(a) 5-objective DTLZ2



(b) 10-objective C2-DTLZ2

Fig. 9.4 Parallel Coordinate Plots showing that the GPR-based F vectors conform with the F vectors in the *PF*-approximation (*Randomly* picked), for five- and 10-objective DTLZ2 problems (taken from [3])

Table 9.3 Comparison of DNN and GPR, with *Edge* test-data on two-objective problems

Problem	M	Model	$\mu(MAE_X)$	$\mu(MAE_F)$	$\sigma(MAE_X)$	$\sigma(MAE_F)$
ZDT1	2	DNN	2.012E-03	1.864E-02	4.350E-04	4.332E-03
		GPR	1.086E-06	2.447E-05	8.081E-07	1.820E-05
ZDT2	2	DNN	1.313E-03	3.986E-02	1.207E-04	3.680E-03
		GPR	8.651E-04	3.811E-02	2.308E-04	1.023E-02

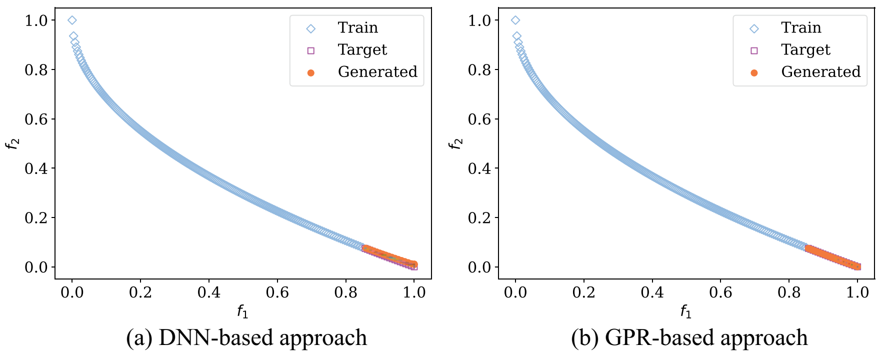


Fig. 9.5 DNN- and GPR-based approaches on *Edge* test-dataset for the ZDT1 problem (taken from [3])

Table 9.4 Comparison of DNN and GPR, with *Continuous* test-data on two-objective problems

Problem	M	Model	$\mu(MAE_X)$	$\mu(MAE_F)$	$\sigma(MAE_X)$	$\sigma(MAE_F)$
ZDT1	2	DNN	6.281E-04	9.264E-03	1.051E-04	1.374E-03
		GPR	2.625E-08	8.466E-08	5.694E-08	1.806E-07
ZDT2	2	DNN	7.387E-04	1.321E-02	1.242E-04	2.357E-03
		GPR	1.083E-04	3.633E-03	3.903E-05	1.320E-03
ZDT3	2	DNN	1.026E-03	3.948E-02	2.030E-04	1.994E-02
		GPR	3.182E-05	3.365E-03	9.706E-06	2.015E-03
BNH	2	DNN	3.701E-03	2.760E-03	4.185E-03	5.378E-02
		GPR	2.343E-03	1.274E-04	3.054E-03	2.224E-03
OSY	2	DNN	2.172E-03	2.720E-02	3.685E-03	6.518E-01
		GPR	1.471E-03	1.243E-03	4.562E-03	5.371E-02

efficient in finding edge gap solutions, the GPR-based approach produces slightly better (more converged) solutions.

Extending the above analysis to *Continuous* test-data, Table 9.4 shows the MAE_X and MAE_F indicators on some two-objective problems, for both DNN- and GPR-based approaches. As evident, compared to the DNN-based approach, the GPR-based approach offers lower values for each of $\mu(MAE_X)$ and $\sigma(MAE_X)$, and also $\mu(MAE_F)$ and $\sigma(MAE_F)$. This implies that the GPR-based approach is better for finding continuous gap solutions as well. For a visual representation, Fig. 9.6 shows the continuous gap solutions produced using the better performing GPR-based approach.

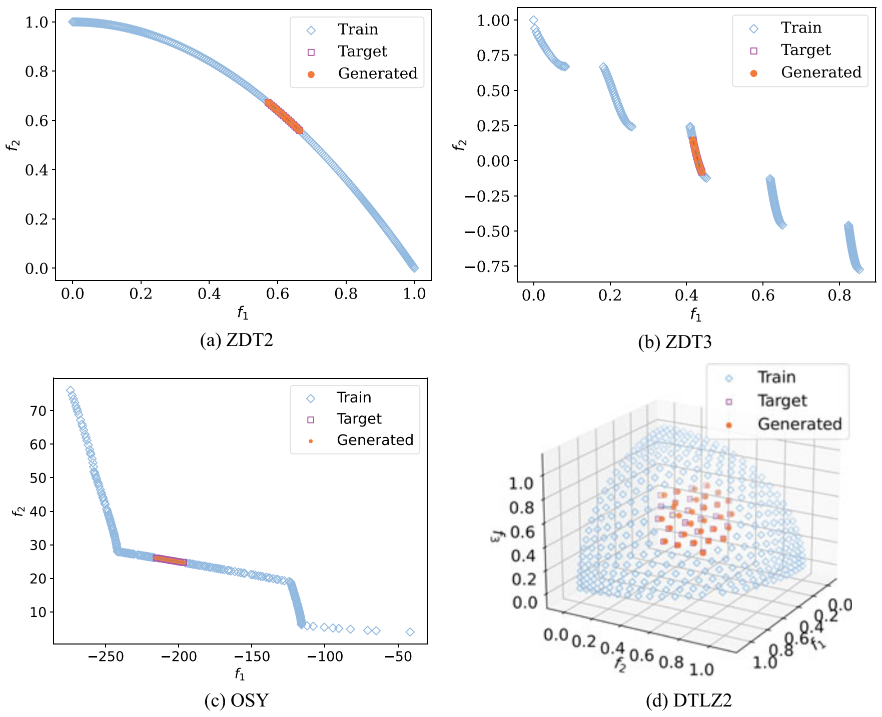
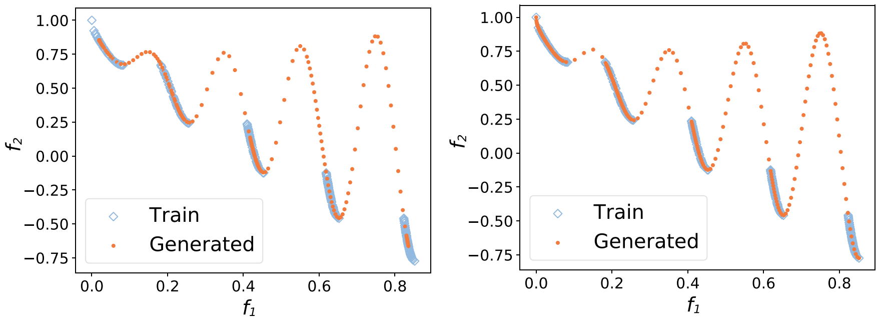


Fig. 9.6 GPR-based approach on *Continuous* test-dataset two-objective (ZDT2 and ZDT3), constrained two-objective (OSY) and three-objective (DTLZ2) problems (taken from [3])

9.3.2 Task 2: Gap-Validation

This section focuses on the validation of Task 2, i.e., the gap-validation task. For this task, it is imperative to select a problem which has natural gaps in its true *PF*, so that it can be validated. Hence, the ZDT3 problem has been used for validating this task. Toward it, a sample set of 200 uniformly distributed *W* vectors has been created across the obtained *PF*-approximation; the trained ML models have been applied to find the respective *X* vectors; and their corresponding *F* vectors are computed. The resulting *F* vectors are shown in Fig. 9.7 (in red color), along with the originally obtained *PF* approximation (in blue color), for both DNN- and GPR-based approaches. It is evident that the *W* vectors corresponding to the true gaps produce dominated solutions, but those corresponding to parts of the true *PF* result in non-dominated solutions. Therefore, this procedure can also be used to testify whether the apparent gaps in the *PF*-approximation offered by an EMâOA are natural gaps or not.



(a) DNN on ZDT3 (b) GPR on ZDT3

Fig. 9.7 Pseudo-weights on true gaps produce dominated solutions, but pseudo-weights on true *PF* produce non-dominated solutions, demonstrating Task 2 (taken from [3])

9.3.3 Task 3: Preferred Solution-Density Propagation

This section focuses on validation of Task 3, i.e., preferred solution-density propagation. In other words, the aim of this task is to improve the density of low-density regions of a given *PF*-approximation. In order to simulate a scenario in which an EMâOA produces a non-uniformly distributed *PF*-approximation, some solutions are removed from a specific region of a uniformly distributed *PF*-approximation. In such a scenario, a GPR model can still be trained, and W vectors from the low-density region of the *PF*-approximation can be used to produce new solutions in that region. Figure 9.8 reveals that the GPR-based approach can provide additional solutions in such low-density regions for three-objective test (DTLZ2) and real-world (crashworthiness) problems.

9.4 Summary

In this chapter, it has been demonstrated that ML models can be used to learn patterns between pseudo-weight vectors and corresponding variable vectors and generate new points on the *PF*-approximation without doing any additional optimization tasks. Furthermore, it has also been demonstrated that this approach can scale up well to handle many-objective test problems, and also real-world problems. This proof-of-concept study and a recent extended study [14] have paved the way for encapsulation of this approach as an additional operator in an EMâOA, to achieve a better distributed *PF*-approximation. Owing to the fact that the ML models are conditioned on the pseudo-weight vectors, this approach can be readily used for decision-making by the user without the need for further optimization.

In the future, it would be interesting to compare this approach with optimization-based gap-filling methods, including preference-based EMâOAs. Applications to

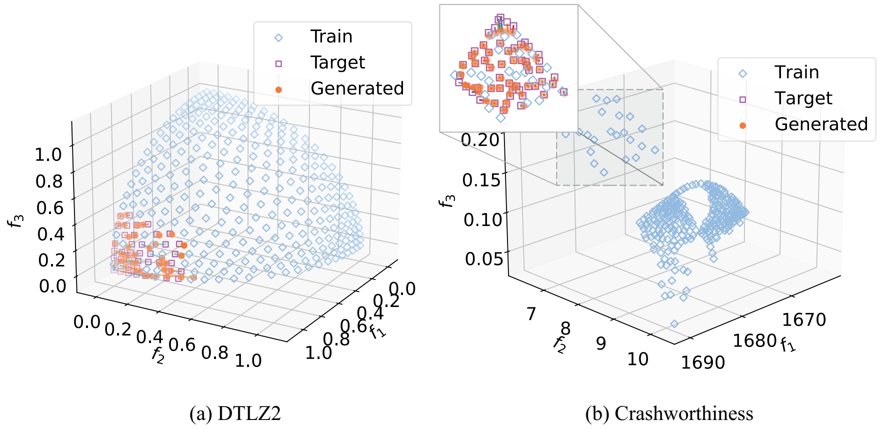


Fig. 9.8 Additional solutions are supplied by the GPR-based approach for two three-objective problems in regions of low density of solutions, demonstrating Task 3. A few blue points were found by EMâOA on a part of the *PF*-approximation, but this ML approach has nicely replenished them (taken from [3])

more complex real-world problems will further shed light on the potential of this approach. Furthermore, such modeling approaches can be improved to include learning of constraint violation and variable bounds during the training process itself. For example, specific activation functions (such as ReLU) can be used for the output layer of DNNs to restrict the output within the variable bounds. Although constraint satisfaction has always occurred in the results here, it would be a challenging task to include constraint satisfaction in the training process, since all obtained solutions (training-dataset) are expected to be feasible. Nevertheless, this proof-of-concept study opens up a unique use of ML methods in assisting multi- and many-objective optimization.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, pp. 2623–2631. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330701
2. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
3. Deb, K., Gondkar, A., Anirudh, S.: Learning to predict Pareto-optimal solutions from pseudo-weights. In: Emmerich, M., Deutz, A., Wang, H., Kononova, A.V., Naujoks, B., Li, K., Miettinen, K., Yevseyeva, I. (eds.) Evolutionary Multi-criterion Optimization, pp. 191–204. Springer Nature Switzerland, Cham (2023)
4. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances

and Applications, pp. 105–145. Springer London, London (2005). https://doi.org/10.1007/1-84628-137-7_6

5. Gu, L., Yang, R.J., Tho, C.H., Makowski, L., Faruque, O., Li, Y.: Optimization and robustness for crashworthiness of side impact. Int. J. Veh. Des. **26**(4) (2001)
6. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006). https://doi.org/10.1162/neco.2006.18.7.1527
7. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
8. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: handling constraints and extending to an adaptive approach. IEEE Trans. Evol. Comput. **18**(4), 602–622 (2014). https://doi.org/10.1109/TEVC.2013.2281534
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015). arxiv:abs/1412.6980
10. Liao, X., Li, Q., Zhang, W., Yang, X.: Multiobjective optimization for crash safety design of vehicle using stepwise regression model. Struct. Multidiscip. Optim. **35**, 561–569 (2008)
11. Osyczka, A., Kundu, S.: A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Struct. Optim. **10**, 94–99 (1995). https://doi.org/10.1007/BF01743536
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019)
13. Rasmussen, C.E.: Gaussian Processes in Machine Learning, pp. 63–71. Springer, Berlin Heidelberg (2004). https://doi.org/10.1007/978-3-540-28650-9_4
14. Suresh, A., Deb, K.: Machine learning based prediction of new Pareto-optimal solutions from pseudo-weights. IEEE Trans. Evol. Comput. (in press)
15. To, T.B., Korn, B.: Mobes: A multiobjective evolution strategy for constrained optimization problems. In: Proceedings of the Third International Conference on Genetic Algorithms (MENDEL97), pp. 176–182 (1997)
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000). https://doi.org/10.1162/106365600568202

Chapter 10
Conclusion and Future Research Directions

This chapter shares the authors' concluding perspectives and points out some potential research directions that could help consolidate the emerging theme of machine learning (ML)-assisted evolutionary multi- and many-objective optimization (EMâO).

10.1 Concluding Perspectives

Any research domain has its own growth trajectory, and it becomes important to continue enriching it through novel interventions, often by employing concepts and ideas from synergistic domains. The domain of evolutionary optimization, seeded in the coming together of computer science and evolutionary principles, is no different. Notably, the availability of multiple sets of solutions over successive generations of EMâO algorithms, or EMâOAs, makes them amenable to the application of ML for various pursuits. However, while the ML domain has seen tremendous growth in the last 15–20 years, efforts toward ML-assisted EMâO have only been sporadic [14, 20], despite a growth of literature in 'evolutionary machine learning' which deals with using evolutionary computation methods for improving the performance of machine learning methods [13, 21, 24]. Recognizing the immense potential for ML-based enhancements in the EMâO domain, this book intends to serve as a single resource that can inform and educate experienced researchers and practitioners on the existing literature, starting with the foundational work on *innovization* (2003) and objective reduction (2006), up to the most recently proposed innovized progress operators (2021–23). This book also seeks to appeal to the novice in the EMâO domain, toward which the fundamental concepts associated with optimization (problem and algorithm types) are sufficiently covered.

In terms of content, this book caters to all three phases of EMâOAs, as highlighted below

- The *search* phase: to enhance the search capabilities of EMâOAs, the innovized progress operators, namely IP2, IP3, and UIP, are presented. These operators enable better Pareto front (PF)-approximation, by producing pro-convergence and/or pro-diversity offspring.
- The *post-optimality* phase: to highlight how the lessons learned from the obtained PF-approximation can be utilized, the notions of automated innovization and objective reduction are presented. These approaches help to extract interpretable knowledge for the user and to better understand the multi- or many-objective optimization problem (MOP/MaOP) at hand. In particular, the objective reduction approach, by revealing redundant objectives, if any, can help simplify the given MOP/MaOP model, promising better search efficiency for the EMâOAs if the reduced representation of the problem is used.
- The *decision-making* phase: for this phase, the δ-MOSS and k-EMOSS analyses based on the objective reduction approach and the pseudo-weight analysis are discussed.

The authors believe that the existing literature presented in this book has only scratched the surface of what may turn out to be one of the dominant research themes of ML-assisted EMâOA. Here, the authors wish to reiterate that the success of any ML-based intervention to modify the creation of natural offspring, or adapt the original search mechanism of an EMâOA would strongly depend on whether the proposed modification/adaptation can (i) avoid disruption of the convergence–diversity balance (otherwise ensured by the base EMâOA), (ii) favorably manage the ML-based risk–reward trade-off, (iii) avoid extra solution evaluations, and (iv) be executed within reasonable run-time.

10.2 Future Research Directions

The focus of the innovized progress operators has been the creation of pro-convergence and/or pro-diversity offspring, toward a better PF-approximation. In essence, these operators seek to complement the natural variation operators in a reference vector (RV)-based EMâOA, referred to as RV-EMâOA. The utility of ML-based interventions could also be explored with respect to the other phases, including, but not limited to, RV *creation* and *update*; *initialization* of solutions; *normalization* of the objective space ($\mathcal{Z}$ space); and algorithm *termination*. Toward a wider gamut of ML-assisted evolutionary optimization, the interventions discussed in the context of RV-EMâOAs (IP2, IP3, and UIP operators) could also be extended to other EMâOAs that do not rely on the use of reference vectors. The utility of the discussed ML-based interventions could also be tested for wider real-world applications, particularly combinatorial optimization problems, where the curse of dimensionality is known to pose a major challenge. Some preliminary ideas with regard to some of the above possibilities are highlighted below.

10.2.1 ML-Assisted Reference Vector Creation and Update

In RV-EMâOAs, each RV ideally leads to a specific solution on the true PF. Therefore, the distribution of RVs, either before or during the optimization run, influences the final distribution of solutions in the approximated PF. In general, a uniform distribution of solutions across the PF is desired. Toward it, structured methods such as the Das–Dennis method and Riesz-energy method (already discussed earlier) exist, which provide for a uniform sampling of RVs in the positive orthant of the $\mathcal{Z}$ space.

However, in special situations, a biased distribution of RVs may be required or desired, and in such situations, the ML intervention could be useful. For example, in the case of constrained problems, not all RVs in uniform distribution may pass through feasible regions. If ML methods could be used to predict infeasible regions, based on the trajectory of the evolving population, advanced RV-update methods could be developed. Such methods can guide the elimination of RVs passing through the infeasible region and the creation of new RVs in the feasible regions, so that the same number of solutions as the RVs can approximate PF. Similarly, in situations where some preference information from decision makers is available, more dense RVs can be created in the preferred region of the PF. This could be achieved by analyzing the preference information from decision makers through an ML method; mapping out the preferred region of the PF; and creating denser RVs in that region using an existing method, such as Das–Dennis's method [4].

Several non-ML-based methods for an adaptive update of RVs over the RV-EMâOA generations already exist [11, 12]. These methods continuously monitor the distribution of solutions with respect to RVs across generations. Subsequently, new RVs are created in the regions where two or more solutions are attached to the same RV (crowded). With ML-based interventions, efficient predictive models can be developed to identify crowded regions, so that RVs that are in the less crowded regions can be relocated to the crowded regions.

10.2.2 ML-Assisted Initialization

Conventionally, EMâOAs are initialized with a population, composed of a predefined number of solutions (N). These solutions are created randomly or by utilizing some uniform sampling methods in the variable space ($\mathcal{X}$ space), such as the Latin hypercube. Although these N solutions may represent a diverse set in the $\mathcal{X}$ space, their representation in the $\mathcal{Z}$ space remains unknown until they are evaluated. Intuitively, if the population is so initialized that it has a good diversity in the $\mathcal{Z}$ space, then it may eventually lead to a better PF-approximation in fewer solution evaluations. Toward this, an ML method may be utilized to develop an inverse mapping model ($F \in \mathcal{Z} \rightarrow X \in \mathcal{X}$), so that a diverse set of solutions can originally be sampled in the $\mathcal{Z}$ space and then translated into the $\mathcal{X}$ space using the learned inverse mapping model.

10.2.3 ML-Assisted Normalization

Normalization of objectives is a crucial part of any EMâOA, since any distance (or crowding metric) computation between two or more objective vectors makes sense only when each objective is brought to the same scale. EMâOAs have a niche for this matter, as for each objective, the current minimum and maximum values can be obtained in each generation. These minimum and maximum objective values can be used to normalize all objective vectors of the population, so that each normalized objective value lies within the range [0, 1]. Often, the minimum and maximum objective values are computed only for the non-dominated solutions in the population, and not the entire population. Although the lower normalization bound is mostly computed as the minimum objective value in the population, some studies have proposed more sophisticated methods to compute the upper normalization bound. For example, the method in [5] identifies the extreme solutions first; then determines an $(M-1)$-dimensional hyperplane passing through these extreme solutions; and finally, the upper bound for each objective is given by the hyperplane's intersection with the respective objective axis. Even this method needs to be modified to handle some specific scenarios—when there is only one non-dominated solution; when the hyperplane's intersection with any objective axis gives a negative intercept; or when the hyperplane degenerates into a lower dimensional plane.

Despite the use of sophisticated methods, it has been observed that the upper normalization bound undergoes significant changes, especially in the initial generations of an EMâOA run, which adversely affects performance [18]. Significant fluctuations in normalization bound over consecutive generations of an EMâOA is undesirable, since it may render the elite solutions found in the former generations meaningless in the latter generations. In this context, it is intuitive that an efficient normalization process should maintain the continuity of normalization bounds between two consecutive generations. ML methods could be utilized to learn how the normalization bounds change over generations, and predict the bounds that could be used for normalizing the objective vectors. It has been clearly observed that the sooner the normalization bounds stabilize, the faster the convergence [2]. Notably, as soon as an MOP/MaOP is formulated, its PF in the M-dimensional $\mathcal{Z}$ is defined, with fixed (theoretical) normalization bounds. EMâOAs attempt to discover these (theoretical) bounds along generations through their evolving population members. Therefore, if sophisticated ML-assisted methods can help discover these bounds sooner, the EMâOAs can converge faster to the true PF.

10.2.4 ML-Assisted Termination

In some of the studies presented in this book, a stabilization tracking algorithm [18] has been utilized, to detect an appropriate timing for terminating an EMâOA run without specifying the maximum generations for termination a priori. In particular, this algorithm requires a set of two parameters that govern the degree of stabilization to be detected to suggest termination. Although these parameters could be set intuitively, it would be better to have a stabilization tracking algorithm that does not require a priori parameter fixation. To achieve this, an ML method can be used to (i) detect the performance trend of the underlying EMâOA on-the-fly, through a performance indicator; (ii) reliably predict the additional generations required by the population to stabilize; and (iii) suggest an appropriate timing for terminating the EMâOA run.

10.2.5 ML-Assisted Hyper-Parameter Learning

Hyper-parameter learning has become an important topic in evolutionary optimization (EO) research, in its own right. An EO algorithm (including EMâOAs) has a number of basic parameters, including population size, variation operators' parameters, termination-related parameters, etc. Although researchers have been trying hard not to introduce any new parameters when proposing any new extensions to these algorithms, the performance significantly depends on the choice of basic parameters mentioned above. Hyper-parameter studies construct an outer optimization loop in which the basic algorithmic parameters are the variables, while the inner loop attempts to solve the given optimization problem by keeping the basic parameters fixed to the outer loop variables [1, 15]. It is clear that, overall, such a methodology attempts to solve the given optimization problem with different basic parameter settings, in search of an optimal parameter set (by optimizing for a given performance indicator). Any effort to make the outer loop more efficient will lead to fewer iterations of the outer loop, significantly saving both computational resource and time. This is where ML-based interventions can help, by learning more promising parameter combinations from the past iterations; predicting to what extent the inner-level optimization should be continued for a specific parameter combination; identifying parameters that are more sensitive for the given problem, etc.

10.2.6 ML-Assisted Innovized Progress Operators for User-Preferred PF Concentration

Often, experienced DMs have a preference toward a specific region of the *PF* that can be dictated by an aspiration point [23] or a reference direction [6], or other means [16]. The innovized progress operators discussed in this book, to find the complete *PF* approximation, can be modified to develop better-focused EMâOAs, than the existing R-NSGA-II [8] or R-NSGA-III [22] algorithms. In such studies, where diversity across the true *PF* is not desired, the pro-diversity IP3 operator can receive a lower preference.

10.2.7 ML-Assisted Transfer Learning

Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, the knowledge gained while learning to recognize cars could apply when trying to recognize trucks. This notion of reusing or transferring information from previously learned tasks for the learning of new tasks could bear multi-pronged utility for the EMâO domain, as highlighted below

- **Design of Problem class-specific EMâOAs**: it is known that the performance of EMâOAs strongly depends on their underlying design aspects, including the representation of a solution, the initialization procedure, the selection, recombination and mutation operators, and the termination criteria. Different combinations of these aspects may work best on different problems that can be classified based on several features, including variable description, search space complexity, etc. Based on an available history of EMâOA runs, ML model(s) can be trained to capture the best *configurations* of algorithmic-components vis-à-vis the problem features. Given an optimization problem with its underlying features known a priori, such ML model(s) can be invoked to predict promising configurations for an EMâOA design.
- **Knowledge-Based and Interactive EMâO**: EMâOAs have primarily been viewed as computational algorithms that start with the push of a button and are expected to end up producing a set of Pareto-optimal solutions. Most existing EMâOAs do not expect user intervention in the middle of a run. It is imperative to note that users (experts in the subject domain of the problem) often have a certain hidden knowledge (hunch), which can be treated as properties of good solutions. However, this knowledge may not be explicitly available as a mathematical construct at the time of problem formulation. Often, this knowledge is contextual and can be extracted when the user is exposed to sample solutions/patterns. Such nuggets of knowledge can be extremely useful while solving a complex optimization problem. For example, as in the case of *innovized repair—rules* found along the intermediate generations of an EMâOA can be reinforced along the subsequent generations. A recent study [10] has shown that such an interactive EMâOA run works best when an optimal number of patterns is used to modify an optimal proportion of population members. While the optimal patterns could be zeroed in using some broad principles, say, smaller size, etc., a better alternative could be to utilize the user's domain knowledge for preference-ordering of the rules. For complex problems, such user intervention is not trivial and carries immense value. To enhance its value, ML methods can help in modeling user preferences, so this learning could be transferred to solve other complex problems from the same problem class.
- **Multi-criterion decision-making (MCDM)**: indexMulti-criterion decision-making (MCDM) Most EMâO studies focus on finding a set of Pareto-optimal solutions and leave the MCDM task of choosing a single preferred solution to the decision maker (DM). However, many existing MCDM methods can be combined with EMâOAs, such that the DM's preferences can influence *selection*, eventually

leading to a single preferred optimal solution at the end of an EMâOA run. ML methods could possibly assist in transfer learning of DM's preferences, to help avoid repeated DM's intervention over the same class of problems. For example, past practices of chosen solutions for similar problems can be *learnt*, to understand the variable and objective combinations that were chosen. This model can then be applied to pick the preferred solution from a fresh set of Pareto-optimal solutions.

10.2.8 ML-Assisted Combinatorial and Other Optimization Studies

Combinatorial optimization problems are considered to be one of the most challenging classes of optimization problems, due to the discrete nature of variables and the high dimensionality of their search spaces. This explains why the applications of EMâOAs on such problems are relatively sparse in the literature. It would be worth investigating whether the innovized progress operators discussed in this book can be extended to such problems and to what extent the search efficacy of EMâOAs could be improved. In addition to the above possibilities, ML-based interventions can also be explored to improve EMâOA's performance in other classes of optimization problems, such as dynamic problems [9], bilevel or multi-level problems [19], variable-length problems [17], and large-scale problems [3, 7].

To conclude: the body of work presented pertaining to ML-assisted EMâO, supported by a rather exhaustive list of foreseeable possibilities, indicates the significance of this book in making evolutionary optimization—in particular, EMâOAs—more efficient, useful, and timely. Advancements in the ML literature can make such studies possible and bring ML and EMâO communities together. In preparing this book, it has been authors' desire that the book would spur more such ML-assisted EMâO studies in the near future.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, p. 2623–2631. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330701
2. Chen, L., Deb, K., Liu, H.L., Zhang, Q.: Effect of objective normalization and penalty parameter on penalty boundary intersection decomposition-based evolutionary many-objective optimization algorithms. Evol. Comput. **29**(1), 157–186 (2021). https://doi.org/10.1162/evco_a_00276
3. Chikumbo, O., Goodman, E., Deb, K.: Triple bottomline many-objective-based decision making for a land use management problem. J. Multi-criterion Dec. Anal.: Optim. Learn. Dec. Supp. **22**(3–4), 133–159 (2015)
4. Das, I., Dennis, J.: Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998)

5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
6. Deb, K., Kumar, A.: Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, pp. 781–788. Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1276958.1277116
7. Deb, K., Myburgh, C.: A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. European J. Oper. Res. **261**(2), 460–474 (2017). https://doi.org/10.1016/j.ejor.2017.02.015
8. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, pp. 635–642. Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1143997.1144112
9. Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Trans. Evol. Comput. **8**(5), 425–442 (2000)
10. Ghosh, A., Deb, K., Goodman, E., Averill, R.: A user-guided innovization-based evolutionary algorithm framework for practical multi-objective optimization problems. Eng. Optim. 1–13 (2022). https://doi.org/10.1080/0305215X.2022.2144275
11. Jain, H., Deb, K.: An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) Evolutionary Multi-Criterion Optimization, pp. 307–321. Springer, Berlin, Heidelberg (2013)
12. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: handling constraints and extending to an adaptive approach. IEEE Trans. Evol. Comput. **18**(4), 602–622 (2014). https://doi.org/10.1109/TEVC.2013.2281534
13. Jena, J.J., Pandey, M., Rautaray, S.S., Jena, S.: Evolutionary algorithms-based machine learning models. Trends of Data Science and Applications: Theory and Practices, pp. 91–111 (2021)
14. Li, B., Wei, Z., Wu, J., Yu, S., Zhang, T., Zhu, C., Zheng, D., Guo, W., Zhao, C., Zhang, J.: Machine learning-enabled globally guaranteed evolutionary computation. Nature Machine Intelligence, pp. 1–11 (2023)
15. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Persp. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002
16. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
17. Ryerkerk, M., Averill, R., Deb, K., Goodman, E.: A survey of evolutionary algorithms using metameric representations. Genetic Program. Evol. Mach. **20**, 441–478 (2019). https://doi.org/10.1007/s10710-019-09356-2
18. Saxena, D.K., Kapoor, S.: On timing the nadir-point estimation and/or termination of reference-based multi- and many-objective evolutionary algorithms. In: Deb, K., Goodman, E., Coello Coello, C.A., Klamroth, K., Miettinen, K., Mostaghim, S., Reed, P. (eds.) Evolutionary Multi-criterion Optimization, pp. 191–202. Springer International Publishing, Cham (2019)
19. Sinha, A., Malo, P., Deb, K.: A review on bilevel optimization: from classical to evolutionary approaches and applications. IEEE Trans. Evol. Comput. **22**(2), 276–295 (2018). https://doi.org/10.1109/TEVC.2017.2712906
20. Sun, C., Wang, H., Jin, Y.: Data-Driven Evolutionary Optimization: Integrating Evolutionary Computation. Machine Learning and Data Science. Springer (2021)
21. Telikani, A., Tahmassebi, A., Banzhaf, W., Gandomi, A.H.: Evolutionary machine learning: a survey. ACM Comput. Surv. (CSUR) **54**(8), 1–35 (2021)
22. Vesikar, Y., Deb, K., Blank, J.: Reference point based NSGA-III for preferred solutions. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1587–1594 (2018). https://doi.org/10.1109/SSCI.2018.8628819

23. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Application, pp. 468–486. Springer, Berlin, Heidelberg (1980). https://doi.org/10.1007/978-3-642-48782-8_32
24. Zhou, Z.H., Yu, Y., Qian, C.: Evolutionary Learning: Advances in Theories and Algorithms. Springer (2019)

Index