MACHINE LEARNING BASED ENHANCEMENTS IN EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

Ph.D. THESIS

by

SUKRIT MITTAL

DEPARTMENT OF MECHANICAL & INDUSTRIAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
AUGUST, 2022

MACHINE LEARNING BASED ENHANCEMENTS IN EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

A THESIS

Submitted in partial fulfilment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

in

MECHANICAL & INDUSTRIAL ENGINEERING

by

SUKRIT MITTAL

DEPARTMENT OF MECHANICAL & INDUSTRIAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
AUGUST, 2022

©INDIAN INSTITUTE OF TECHNOLOGY ROORKEE, ROORKEE-2022
ALL RIGHTS RESERVED

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

STUDENT'S DECLARATION

I hereby certify that the work presented in this thesis entitled, "MACHINE LEARNING BASED ENHANCMENTS IN EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION" is my own work carried out during a period from July, 2018 to August, 2022 under the supervision of Dr. Dhish Kumar Saxena, Associate Professor, Department of Mechanical and Industrial Engineering, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in the thesis has not been submitted for the award of another degree of this or any other institute.

Dated: August, 2022 **(Sukrit Mittal)**

SUPERVISOR'S DECLARATION

This is to certify that the above mentioned work is carried out under my supervision.

Dated: August, 2022 **(Dhish Kumar Saxena)**

Abstract

Learning of effective problem information from the search space explored along an Evolutionary Multi-objective Optimization (EMO) algorithm's run, and its utilization to improve the convergence of subsequent solutions, have presented important research directions. This concept, referred to as *online innovization*, attempts to extract the inter-variable relationships from the intermediate solutions of an EMO run; and utilize them to *repair* the subsequent offspring solutions for better convergence. While this concept is promising, its generality is marred by the fact that the relationship structures need to be specified a priori.

Inspired by the notion of online innovization, this thesis aims to broaden its scope by factoring in both convergence and diversity; doing away with the need to specify the relationship structures a priori; and avoiding any extra solution evaluations compared to the base EMO algorithm. To this effect, this thesis utilizes the framework of Reference-vector based EMO (RV-EMO) algorithms and proposes three *machine learning based operators*, including: (i) IP2 – Innovized Progress operator 2 – for enhancement of convergence, (ii) IP3 – Innovized Progress operator 3 – for enhancement of diversity, and (iii) UIP – Unified Innovized Progress – for simultaneous enhancement of convergence and diversity, in an adaptive manner. In principle, these operators rely on intermittently learning the efficient *search* directions based on inter-generational and/or intra-generational solutions of an RV-EMO run, and utilizing these for *advancement* and/or *creation* of a fraction of the offspring in the same generation, without requiring any additional solution evaluations. The efficacy of the UIP operator has been established on a wide range of test problems, with characteristics including, convergence-hardness, diversity-hardness, bias, multi-modality, and multi- and many-objectives. This thesis's distinctive contribution lies in setting the foundations for machine learning based evolutionary optimization. The encouraging proof-of-concept results make it worthy of further research.

Acknowledgements

The journey of obtaining a PhD is a life-altering period in one's life. Even though it makes one feel alone occasionally, there has been enormous support from several people that helped me sail through it all.

First and foremost, I would like to thank my supervisor, Dr. Dhish Kumar Saxena, for his guidance and unconditional support. He has been associated with me for over eight years, first as a course instructor, then as a mentor, and eventually, as a supervisor. His focus on the development of fundamental and innovative ideas is something that won over me in our first few interactions itself. He gave me the space to develop new ideas as an independent researcher, keeping a regular check if I was astray. His inclination toward certain specifics and nuances when pitching new ideas helped me significantly in developing my academic writing skills. Since the beginning, I've seen him as a person who would pursue what he believes is right. And in the past four years of working closely with him, I've never seen him do otherwise. Also, I would thank him for believing in me, providing me with this opportunity, and allowing me to be a part of the lineage of Prof. John H. Holland, the father of genetic algorithms.

Secondly, I would like to thank Prof. Kalyanmoy Deb and Prof. Erik D. Goodman for their input and support throughout my PhD tenure. Starting as an amateur who barely had a taste of his research topic, working with these pioneers was nothing short of an exhilarating experience for me. I would extend special gratitude for their hospitality when I was a visiting scholar at Michigan State University, especially how Prof. Deb and his wife took care of me during the COVID-19 pandemic.

I would like to mention some of my close ones, especially the one who had been cheering for me from the sidelines. Most notably, Dr. Divyam Aggarwal, who had been a senior and a friend in my bachelor's as well as in my PhD; Nipun Behl, whose outlook towards life has helped me gain peace in unsettling times, both personally and professionally; and Parvej, whose camaraderie made me push myself further to my goals. Some others include Dr. Aditya Rahul, Ankita Sood, Garima Bhandari, Tejas Naik, Suresh Chintam, Ram Singh Rana, Nikki Rathore, Dr. Subash Prasad Rai and Dr. Sarang Kapoor. They have all been there with me on this roller-coaster ride. I would also thank the COIN lab members for hosting me and accepting me as a part of the lab. Special thanks to Dr. Yashesh Dhebar and my flatmate, Christopher LeBelt, who went the extra mile in supporting me with all my needs in the US.

Even though several names come to my mind in retrospection, some people remain unnamed, yet their support cannot be ignored. I would like to thank the Café Coffee Day staff at IIT Roorkee for hosting me every day I have been on this campus. Their adjustment per my convenience, letting me work for hours at a stretch, is commendable. After the long days and short nights, good coffee worked the best for me to boot up.

Finally, I would like to thank my parents. Their support, compromises and sacrifices have been instrumental, enabling me to reach where I am today. Leaving my job for a full-time PhD was a tough call, but their support made me cross the line, and I never looked back after that. After spending the past few years in academic research, it is nearly impossible for me to imagine what I would have been doing had I not taken this road. Even working from home during the pandemic, they made all possible adjustments, allowing me to work with a free mind.

This acknowledgement would be incomplete without mentioning the sponsors: the Ministry of Human Resource Development (MHRD), India; and Michigan State University, for funding the project P66 titled, "INNOVIZATION: Discovery of Innovative Knowledge through Optimization and Machine Learning" under the SPARC scheme. The work presented in this thesis was developed as a part of the project deliverable. I would also thank Dr. Julian Blank for developing the pymoo framework, based on which this work has been implemented.

I am sure there are others who have been a part of this journey in different ways, yet I am forgetting their names. I am grateful to all of them.

Sukrit Mittal

Dedicated

to

Ma and Pa

and

Nana ji

Contents

List of Figures

List of Tables

List of Abbreviations in Chronological Order

SOP	Single-objective Optimization Problem
MOP	Multi-objective Optimization Problem
PO	Pareto-optimal
EA	Evolutionary Algorithm
PS	Pareto Set
PF	Pareto Front
EMO	Evolutionary Multi-objective Optimization
RV	Reference Vector
RV-EMO	Reference Vector based Evolutionary Multi-objective Optimization
ML	Machine Learning
IP	Innovized Progress
IP2	Innovized Progress 2
IP3	Innovized Progress 3
UIP	Unified Innovized Progress
EDA	Estimation of Distribution Algorithm
RL	Reinforcement Learning
LA	Learning Automaton
PCA	Principle Component Analysis
IR	Innovized Repair
ASF	Achievement Scalarization Function
BWS	Biased Weighted Sum
WS	Weighted Sum
PBI	Penalty-based Boundary Intersection
APS	Adaptive Penalty Scheme
SPS	Subproblem-based Penalty Scheme
APD	Angle Penalized Distance
LWS	Localized Weighted Sum
DE	Differential Evolution
AOS	Adaptive Operator Selection

PD	Perpendicular Distance
RF	Random Forest
ANN	Artifical Neural Network
MSE	Mean Squared Error
TSLD	Two-layered Simple Lattice Design
ILD	Incremental Lattive Design
kNN	k-Nearest Neighbours

Chapter 1

Introduction

Optimization is an iterative process of arriving at one or more optimal solution(s), depending on the number of objectives (M). In the case of $M = 1$ or single-objective optimization problems (SOPs), the goal is usually to find a unique global optimum. However, in the case of $M \geq 2$ or multi-objective optimization problems (MOPs), the notion of a unique global optimum does not exist, and the goal extends to finding a set of best trade-off solutions. These best trade-off solutions are referred to as Pareto optimal (PO) solutions, where one cannot be said better than the other [Deb, 2001]. The above goals can be fulfilled by using either point-based or population-based algorithms. However, in MOPs, the population-based algorithms are more suited since a set of PO solutions can be obtained through a single algorithmic run.

Population-based algorithms, commonly referred to as Evolutionary Algorithms (EAs), constitute a class of nature-inspired algorithms that have demonstrated efficient *search* capabilities in tackling optimization problems, especially MOPs [Coello Coello et al., 2020]. In the case of EAs, the goal is to evolve a finite set of random solutions (sized N) over several iterations, referred to as *generations*, towards the global optimum (in the case of SOPs) or the PO solutions (in case of MOPs). In any generation, EAs rely on: creation of a new *offspring* population using the natural variation operators (Q^{V}, sized N), from the *parent* population (P, sized N); merging of the parent and offspring populations; and *selection* of the best N solutions, that constitute the parent population for the next generation. This process is repeated in each generation until the pre-specified termination criterion is met, and the final obtained population is reported. Notably, different selection criteria can be used in EAs to tackle SOPs and MOPs, as required suitably. Considering the scope of this thesis, the subsequent discussion only encircles MOPs.

MOPs are characterized by two or more conflicting objectives. An MOP, involving M objec-

tives and n variables can be represented as:

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

Here, $\Omega \subseteq \mathbb{R}^n$ signifies the feasible variable space (X-space), implying that any solution in Ω satisfies the constraints in the problem, if any. For each solution $X \in \Omega \subseteq \mathbb{R}^n$, there exists an $F(X) \equiv \{f_1(X), \ldots, f_M(X)\} \in \mathbb{R}^M$. In that, as per [Deb, 2001]:

- given two solutions $X \in \Omega$ and $Y \in \Omega$, X is said to *dominate* Y if X is not worse than Y in any objective, and X is better than Y in at least one objective.
- a solution $X^* \in \Omega$ is called a PO solution if there is no $X \in \Omega$ that dominates X^*. The set of all PO solutions is called the Pareto set (PS), and its corresponding representation in the objective space (F-space) constitutes the *efficient set* or loosely known as the Pareto front, given by $PF = \{F(X) \in \mathbb{R}^M \mid X \in PS\}$.

The extension of EAs towards tackling MOPs is referred to as Evolutionary Multi-objective Optimization (EMO), as detailed in the following section.

1.1 EMO algorithms

In the past two decades, EMO has gained significant attention in performing a variety of search and optimization tasks over several domains, including but not limited to aerodynamic design, molecular structure of drugs, medical decision making, supply chain management, land use planning, document summarization, and data clustering [Anand et al., 2007, Saini et al., 2021, Stewart et al., 2008].

EMO algorithms extend the goal of EAs to finding a set of solutions that approximates well the true PF for a given MOP, in terms of *convergence* (proximity of the true PF), and *diversity* (coverage across the PF with a reasonably uniform distribution). This notion of convergence and diversity is symbolically depicted in Figure 1.1.

Once a set of well-converged and well-diverse PO solutions has been obtained, the user (or the decision-maker) can utilize some *decision-making* techniques to arrive at a single useful solution from the set of PO solutions [Deb, 2001, Rachmawati and Srinivasan, 2009].

One of the significant extensions to EMO algorithms includes reference vector (RV) based EMO algorithms, referred to as RV-EMO algorithms here onward. These algorithms are often

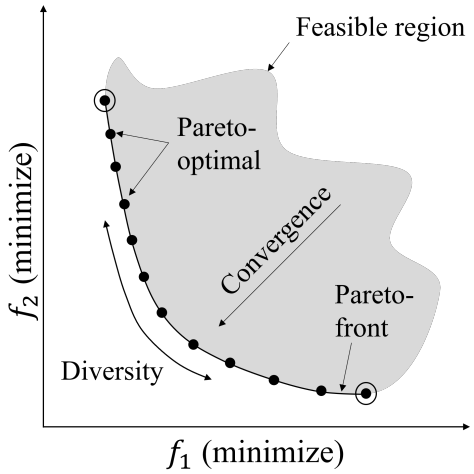


Figure 1.1. A symbolic depiction of convergence and diversity in the PO solutions, in F-space.

also referred to as *decomposition*-based algorithms [Zhang and Li, 2007]. The basic idea in these algorithms is to decompose a given MOP into several scalar optimization subproblems (one per RV) in the F-space and solve these simultaneously to arrive at a reasonable PF-approximation.

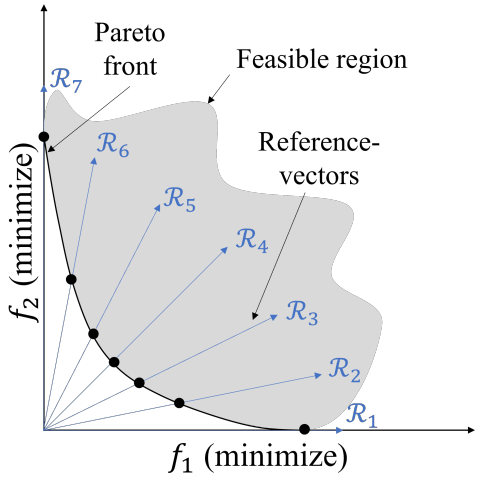


Figure 1.2. A symbolic depiction of a decomposed F-space.

The above decomposition in the F-space is symbolically depicted in Figure 1.2. In that, seven RVs ($\mathcal{R}_1$–$\mathcal{R}_7$) are shown that pass through the true PF. Finding a well-converged solution for each RV, would facilitate a reasonably diverse PF-approximation. It is critical to note that a movement *along* the RVs in the F-space would correspond to a change in the convergence level. In contrast, a movement *across* the RVs in the F-space would correspond to a change in the context of diversity.

In the above background, two key aspects that shed light on this thesis's motivation are highlighted below. While the following discussion in the context of EMO algorithms is generic, the same is also applicable to the RV-EMO algorithms (a subset of EMO algorithms).

1.1.1 Existing Innovized Repair Operator in EMO

It is intuitive to think that the PO solutions may possess some special properties, referred to as *rules*, that set them apart from non-optimal solutions or random solutions. For example, in a mathematically well-behaved problem with convex, continuous, and differentiable functions, the PO solutions must adhere to the *Karush-Kuhn-Tucker* necessary conditions of Pareto-optimality [Bertsekas et al., 2003].

The task of *extracting* these rules from a set of PO solutions was referred to as *innovization* in [Deb, 2003]. It was later exemplified as an *offline* task with utility on a number of real-world engineering problems [Deb and Srinivasan, 2006, Ng et al., 2009]. In that, the first (of the two) step was to obtain a set of near PO solutions, which can be achieved using any EMO algorithm. The second step was to extract the said rules in the form of explicit mathematical relationships among the decision variables, objectives, and constraints, that are *common* to the majority of the solutions. These mathematical relationships, extracted through the two-step innovization task, were referred to as the *innovized relationships*.

While the innovized relationships are interpretable and help reveal problem-features, some studies [Deb and Datta, 2012, Myburgh and Deb, 2018] went a step ahead and utilized these relationships (obtained from some initial EMO runs) towards obtaining faster convergence in the subsequent EMO runs. This demonstrated practical utility, especially when an optimization task needs to be performed repeatedly with minor changes in the problem formulation, such as the scheduling tasks. Motivated by these successful attempts, [Gaur and Deb, 2016, 2017] made the first attempt in extracting and utilizing the innovized relationships within a single EMO run, referred to as *online innovization*. In that, the extracted relationships were used by means of an *innovized repair* operator, to selectively *repair* a fraction of the offspring created by the natural variation operators. While a better convergence was reported on some test problems, the above method needed an a priori specification of the structure of innovized relationships, thus limiting its generality on problems with unknown characteristics.

More recently, a unified automated innovization algorithm was proposed [Mittal et al., 2020]

that could extract useful relationships from problems with variables of both continuous and discrete nature. Further, to reduce the computational time of extracting innovized relationships, increasing its utility in online innovization, a computationally efficient innovization algorithm was proposed [Garg et al., 2020]. However, the limitation discussed above in the context of existing innovization studies, holds for these propositions too.

1.1.2 Offspring Improvement vis-à-vis Dual goals in EMO

Conventionally most EMO algorithms pursue the dual goals of convergence and diversity, through the selection procedure executed at the end of each generation, where the N best solutions are selected for the next generation from the combined parent and offspring populations. Notably, a well-diversified set of solutions has no practical relevance if the solutions have not converged to the PF. Similarly, well-converged solutions that occupy only a small part of the PF is not desirable. Hence, managing the delicate convergence-diversity balance becomes critical for the efficacy of any EMO algorithm.

The above plausibly explains why in most EMO algorithms, offspring creation is left to the natural variation operators, and is not explicitly customized to pursue convergence and/or diversity. In a departure from this trend, a recent study [Seada et al., 2019] provisions for: (i) creating better-converged offspring solutions in poorly converged regions of the obtained front, to effect better convergence, and (ii) creating offspring solutions in empty regions of the obtained front, to effect better diversity, with the help of a gradient-based local search operator. This approach is promising as it does not favour convergence over diversity, or vice-versa. However, an associated pitfall is that using a local search operator requires additional solution evaluations, over and above the default solution evaluations of the underlying EMO algorithm. In a more recent study [Tian et al., 2021], while the offspring solutions are created using the natural variation operators only, an indirect attempt to emphasize convergence and diversity has been made through a judicious choice of the parent solutions for mating. In that: (i) to emphasize convergence, both parents with good convergence degrees are selected, (ii) to emphasize diversity, both parents with good diversity degrees are selected, and (iii) otherwise, a parent with good convergence and another with good diversity degree, are selected.

1.2 Contribution and Structure of the Thesis

This thesis is inspired by the notion of *online innovization*, and aims to broaden its scope by factoring in both convergence and diversity; doing away with the need to specify the relationship structures, a priori; and avoiding any additional solution evaluations compared to the base EMO algorithm.

Towards it, this thesis recognizes that in the existing online innovization studies, it is implicitly assumed that the inter-variable relationships extracted within the scope of pre-specified relationship structures, at any intermediate EMO generation; and propagated through subsequent offspring *repair*, help induce better convergence. Hence, any endeavour to do away with a priori specification of the relationship structures, would require alternative criteria that could guide improvement in convergence and diversity. This thesis further recognizes that if the scope of online innovization could be narrowed down to RV-EMO algorithms, then the underlying RVs could provide the aspired criteria. In that, even though the RVs are sampled in the F-space, and their direct representations are not available in the X-space, they can still provide the criteria for improvement in convergence and diversity. The rationale for this claim is rooted in the recognition that: (i) the solutions associated with the RVs in the F-space, have their representations in the X-space, (ii) mapping of inter-generational solutions in the F-space (from previous to the current generation, implying, improvement in F-space) can facilitate mapping of the solutions' underlying X vectors, and (iii) such mappings in X-space learnt using machine learning (ML) methods, can be treated as efficient search trajectories that could be utilized for improvement in convergence. Similarly, mapping of intra-generational solutions in the F-space, can facilitate learning of efficient search trajectories in the X-space that could be utilized to improve diversity.

In the premise set above, this thesis utilizes the framework of RV-EMO algorithms, and proposes *three ML based Innovized Progress (IP) operators*, including:

1. IP2 (Innovized Progress 2)[1]: for convergence enhancement. It relies on learning the efficient search directions (in X-space), based on mapping of *inter-generational* solutions in F-space, *along* the reference vectors; and utilizing the learning in the same generation, to produce pro-convergence offspring, namely, Q^{IP2} (in X-space).

[1]As a prelude to this thesis, an *Innovized Progress* operator, namely *IP* [Mittal et al., 2021b] was developed. Subsequently, its improved version in IP2 [Mittal et al., 2021a] was developed. Since both cater to a similar goal, the former has been omitted from this thesis.

2. IP3 (Innovized Progress 3): for diversity enhancement. It relies on learning the efficient search directions (in X-space), based on mapping of *intra-generational* solutions in F-space, *across* the reference vectors; and utilizing the learning in the same generation, to produce pro-diversity offspring, namely, Q^{IP3} (in X-space).

3. UIP (Unified Innovized Progress): for convergence and diversity enhancement. It relies on invoking either or both of the IP2 and IP3 operators in the same generation, to produce both pro-convergence and pro-diversity offspring, referred to as Q^{UIP}. In that, if either of IP2 or IP3 are invoked, then $Q^{\mathrm{UIP}} \equiv Q^{\mathrm{IP2}}$ or $Q^{\mathrm{UIP}} \equiv Q^{\mathrm{IP3}}$, as applicable. However, if both IP2 and IP3 are invoked, then $Q^{\mathrm{UIP}} \equiv Q^{\mathrm{IP2}} \cup Q^{\mathrm{IP3}}$.

The above operators, when integrated into an RV-EMO algorithm, are referred to as RV-EMO-IP2, RV-EMO-IP3 and RV-EMO-UIP, respectively. Their efficacy vis-à-vis the base EMO algorithm is bound to be influenced by the following questions:

- In any given generation: what proportion ($\mathcal{P}$, in percentage) of the total offspring solutions (N) should be fetched through the innovized progress operators. In other words, what fraction of N should Q^{IP2} or Q^{IP3} or Q^{UIP} be?

- Across all the generations: how frequently should the innovized progress operators be invoked. In other words, if $t^{\mathrm{IP2}}_{\mathrm{freq}}$ and $t^{\mathrm{IP3}}_{\mathrm{freq}}$ represent[2] the number of generations between two successive invocations of IP2 and IP3, respectively, then what should the values of $t^{\mathrm{IP2}}_{\mathrm{freq}}$ and $t^{\mathrm{IP3}}_{\mathrm{freq}}$ be?

This thesis postulates that the answers to the above questions, ought to cater to the dual considerations of:

- *Convergence-diversity balance:* conventionally, the natural variation operators (for instance, crossover and mutation, in the context of genetic algorithms) utilize the principle of *guided randomness* to produce the offspring solutions Q^{V}, without any explicit consideration for their convergence or diversity characteristics. Such convergence-diversity-neutral offspring, alongside the parent solutions, serve as input to the *selection* operator, which pursues EMO's dual goals of convergence and diversity. Hence, it is imperative that in

[2]Here, t^{UIP}_{freq} is not included, since the invocation of UIP is directly linked to the invocation of either or both of IP2 and IP3, governed by $t^{\mathrm{IP2}}_{\mathrm{freq}}$ and $t^{\mathrm{IP3}}_{\mathrm{freq}}$.

any generation, the offspring resulting through the innovized progress operators are not over-skewed in favour of either convergence or diversity.

- *Risk-rewards tradeoff:* notably, the proposed innovized progress operators rely on *learning* the efficient search directions (in X-space), based on mapping of *inter-* and *intra-generational* solutions in F-space. Clearly, there is a possibility that such a *learning* may not be meaningful due to multiple factors, including the non-linearity in the given problem; training data comprising of intermediate generation solutions which may not be representative of the PF; and choice of ML methods, etc. Hence, it is imperative that the degree of reliance on the innovized progress operators is moderated, considering the risk-rewards tradeoff associated with the accuracy of the underlying ML models.

Table 1.1. Evaluation of RV-EMO-IP2/IP3/UIP vis-à-vis key considerations of convergence-diversity balance and risk-rewards tradeoff associated with reliance on ML based operators.

Algorithm		Nature of Progressed Offspring	Convergence-diversity balance calls for?	Risk-rewards tradeoff to be factored?
RV-EMO-IP2		Pro-convergence: Q^{IP2}	A dominant share of offspring Q^{V} that are convergence-diversity-neutral	Yes
RV-EMO-IP3		Pro-diversity: Q^{IP3}		
RV-EMO-UIP	If only IP2 is invoked	Pro-convergence: Q^{IP2}		
	If only IP3 is invoked	Pro-diversity: Q^{IP3}		
	If both IP2 and IP3 are invoked	Both Pro-convergence and Pro-diversity: $Q^{\mathrm{IP2}} \cup Q^{\mathrm{IP3}}$	No intervention	

In this background, Table 1.1 evaluates each scenario associated with RV-EMO-IP2, RV-EMO-IP3, and RV-EMO-UIP vis-à-vis the dual considerations cited above. It is rather apparent that the risk-rewards tradeoff associated with reliance on ML-based operators is the overarching consideration. Also, in the case of RV-EMO-UIP where both IP2 and IP3 may get invoked in the same generation, the convergence-diversity balance could be implicit, since the IP2 based pro-convergence offspring can balance the IP3 based pro-diversity offspring. In all other instances, the IP2 based convergence-enhancement or IP3 based diversity-enhancement, needs to be balanced, by provisioning for a dominant share of offspring produced by natural variation operators.

This thesis further postulates that both the key considerations of convergence-diversity balance and risk-rewards tradeoff could be addressed by ensuring that over the entire run of RV-EMO-IP2/IP3/UIP (accounting for all the generations till termination), the share of convergence-

diversity-neutral Q^{V} outweighs the contributions of pro-convergence Q^{IP2} and/or pro-diversity Q^{IP3}. While the above could be ensured in multiple ways, this thesis adopts the following settings to realize the same:

1. in the case of RV-EMO-IP2: the proportion of offspring created using IP2 ($\mathcal{P}^{\mathrm{IP2}}$) is kept as 50% in each generation where IP2 is invoked; and $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 1$. The justification is: (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 owing to $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 1$, Q^{V} shall contribute 100% of N. Hence, the overall contribution of Q^{V} across all the generations is guaranteed to be dominant.

2. in the case of RV-EMO-IP3: the proportion of offspring creating using IP3 ($\mathcal{P}^{\mathrm{IP3}}$) is kept as 50% in each generation where IP3 is invoked; and $t^{\mathrm{IP3}}_{\mathrm{freq}} \geq 1$. The justification is: (i) in those generations where IP3 is invoked, Q^{V} shall still contribute 50% of N, and (ii) in generations where IP3 is not invoked owing to $t^{\mathrm{IP3}}_{\mathrm{freq}} \geq 1$, Q^{V} shall contribute 100% of N. Hence, the overall contribution of Q^{V} across all the generations is guaranteed to be dominant.

3. in the case of RV-EMO-UIP: both $\mathcal{P}^{\mathrm{IP2}}$ and $\mathcal{P}^{\mathrm{IP3}}$ are kept as 50% in each generation; and $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 2$ and $t^{\mathrm{IP3}}_{\mathrm{freq}} \geq 2$. This justification is as follows:

   - in generations where only IP2 or IP3 is invoked, Q^{V} shall still contribute 50% of N; and when neither of them is invoked owing to $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 2$ and $t^{\mathrm{IP3}}_{\mathrm{freq}} \geq 2$, Q^{V} shall contribute 100% of N. Hence, the overall contribution of Q^{V} across all the generations is guaranteed to be dominant.
   - in generations where both IP2 and IP3 are invoked, Q^{V} shall be 0% of N. However, $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 2$ and $t^{\mathrm{IP3}}_{\mathrm{freq}} \geq 2$ shall enforce that at least in the next generation neither IP2 nor IP3 is invoked, and Q^{V} shall contribute 100% of N. Hence, the overall contribution of Q^{V} across all the generations is guaranteed to be at least 50%.

The above discussion endorses that under all possible scenarios the share of Q^{V} remains dominant compared to Q^{IP2} and/or Q^{IP3}. This is symbolically endorsed in Figure 1.3. In that, Q^{V} is separated from others through a fuzzy boundary. This could be attributed to the fact that a priori quantification of Q^{V}'s exact share for an entire run of RV-EMO-IP2/IP3/UIP is not

possible, since $t^{\text{IP2}}_{\text{freq}}$ and $t^{\text{IP3}}_{\text{freq}}$ are adapted on-the-fly based on the survival rate of Q^{IP2} and Q^{IP3}, respectively.

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

Figure 1.3. Symbolic depiction of the degree of operators' contribution to convergence and diversity, over an entire run of RV-EMO-IP2, RV-EMO-IP3, and RV-EMO-UIP, respectively. The offspring created through natural variation operators Q^{V} are represented by a different color since they do not impose any explicit preference for either convergence or diversity.

In the wake of this thesis's core contributions summarized above, the chapter-wise layout is highlighted below. In that, while Chapters 1 and 2 set the context, the core contributions are detailed in Chapters 3–5.

- **Chapter 1** introduces multi-objective optimization problems; presents an overview of how EMO algorithms operate in principle; and briefly highlights two key themes that have set the motivation for this thesis.

- **Chapter 2** provides a literature review of the related works. In that, ML-based enhancements in EMO algorithms are reviewed first, followed by other enhancements, referred to as "non-ML-based".

- **Chapter 3** proposes the IP2 operator, designed for convergence-enhancement in the RV-EMO algorithms. Following the description of the IP2 operator, its computational complexity analysis is presented, followed by proof-of-concept results and discussions on convergence-hard multi-objective problems.

- **Chapter 4** proposes the IP3 operator, designed for diversity-enhancement in RV-EMO algorithms. Following the description of the IP3 operator, its computational complexity

analysis is presented, followed by proof-of-concept results and discussions on diversity-hard multi-objective problems.

- **Chapter 5** marks the culmination of the stated aim of this thesis. In that, the IP2 and IP3 operators are combined with relevant adaptations, leading to the UIP operator. Its efficacy has been established on a wide range of test problems, with characteristics including, convergence-hardness, diversity-hardness, bias, multi-modality, and multi- and many-objectives.

- **Chapter 6** concludes the thesis. In that, the contributions of the thesis have been summarized, and some of the future research directions have been highlighted.

Chapter 2

Literature Review

Since the early 1990s, many efficient EMO algorithms have been proposed. For solving complex MOPs, the EMO algorithms purely relying on the natural variation operators might not produce an efficient search [Li et al., 2013, Pelikan et al., 2002]. To effect a better search, several enhancements to these EMO algorithms have been proposed [Coello and Lamont, 2004, Deb and Myburgh, 2017]. Considering the scope of this thesis, these enhancements have been broadly classified as: (i) ML-based enhancements, and (ii) non-ML-based enhancements, as discussed in this chapter. In addition, some other developments, specific to RV-EMO algorithms, have been presented. Finally, some key research gaps are identified in wake of which, the aim and objectives of this thesis are defined.

2.1 ML-based Enhancements in EMO

In EMO algorithms, several ML-based enhancements have been proposed: (i) through surrogate-modelling, (ii) through model-based offspring creation, (iii) through efficient mutation, and (iv) through innovized repair operator. These enhancements have been discussed, in detail, in the following subsections.

2.1.1 Through Surrogate Modelling

ML methods have often been used in the EMO 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). A 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 [Bhattacharjee et al., 2017, Chugh et al., 2018, Dutta and Gandomi, 2020].

While the overarching idea of ML-based surrogate modelling in EMO is highlighted above, it is incomplete without taking constraint function evaluation into account, which might be computationally expensive as well. Given this, the function and constraint evaluations are collectively referred to as *solution evaluations* in this thesis. Several approaches exist, where each objective function and constraint function is modelled *independently* from the already evaluated solutions, using different ML methods, such as Gaussian random-field models [Emmerich et al., 2006]; Gaussian regression models [El-Beltagy et al., 1999]; Kriging [Jones, 2001, Sinha et al., 2018]; radial basis functions [Mullur and Messac, 2006]; random forests [Wang and Jin, 2020]; response surfaces [Lian and Liou, 2005]; and support vector regression [Inapakurthi and Mitra, 2022].

In a slight departure, [Deb et al., 2019] discussed the possibility of building a single *collective* surrogate model for all objective functions using scalarizing functions, such as weighted sum or Tchebychev [Miettinen, 1999]; and separately building a single surrogate model for all constraint functions using a combined normalized constraint violation function [Deb and Datta, 2012]. In a balancing act, [Hussein et al., 2018] proposed a framework that could adaptively switch between independent and collective surrogate models.

Besides facilitating the approximated solution evaluations, surrogate models have also been used to perform other tasks in EMO, including local search [Koçer and Uymaz, 2021, Zhou et al., 2021] and efficient offspring-creation [Li et al., 2020, Mallipeddi and Lee, 2012]. While the notion of local search in EMO has been discussed in detail under the "non-ML-based enhancements" category, later in this chapter, the surrogate model based local search is discussed next, followed by the surrogate based efficient offspring creation.

Performing local search in the intermediate generations of an EMO run, requires additional solution evaluations beyond the usual offspring evaluations in each generation. To reduce its burden, some studies use a surrogate model to perform the local search using approximate solution evaluations [Lima et al., 2006, 2009]. These methods combine a structural hill-climber for local search with BOA [Pelican et al., 1999], in which 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, since different dynamics were observed for different problems [Lima et al., 2006].

Some methods have also used surrogate models for efficient offspring creation. In that, [Mallipeddi and Lee, 2012] relies on generating multiple offspring from the same set of parents (in each generation) by the use of 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 latter's fitness 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 [Li et al., 2020], where besides the parents in any generation, some extra solutions which could serve as potent parents are created. Again, the evaluation of the fitness of such solutions is based on surrogate models instead of the potentially expensive actual evaluations.

2.1.2 Through Model-based Offspring Sampling

EMO algorithms with model-based offspring sampling started with the development of Estimation of Distribution Algorithms (EDAs) in 1996 [Miuhlenbein and Paaß, 1996]. EDAs were designed to directly extract the global search space statistical information 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 [Pelikan et al., 1999], MONEDA [Martí et al., 2008], RM-MEDA [Zhang et al., 2008], EDA-VNS [Du et al., 2021] and HMOBEDA [Martins et al., 2021], have shown a potentially distinctive advantage of exploiting the inter-variable dependencies in creating new offspring solutions. One of the significant extensions to EDAs include FEG-EDA [Xu et al., 2014], that: (i) maps the original search space to a modified search space, (ii) samples new offspring solutions in the modified search space, and (iii) revert 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, that include the choice of: (a) selection strategy for the elite solutions and (b) building strategy for the probability distribution model [Chen et al., 2006].

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 PS of a given MOP. However, there are other MOPs where independent mating through EMO's natural variation operators produces a more efficient search [Mittal

et al., 2021b]. In this background, several EMO algorithms have been proposed that create offspring partially through the natural variation operators and partially through model-based sampling, that include M-MOEA [Zhou et al., 2005], Hybrid [Zhou et al., 2006], IM-MOEA [Cheng et al., 2015] and GMOEA [He et al., 2021].

2.1.3 Through Efficient Mutation

In EMO algorithms, mutation is one of the natural variation operators, that is primarily used as a mechanism for maintaining diversity in the population [Holland, 1975]. Although the mutation operator alone might not constitute an effective search, it plays a crucial role along with a suitable recombination operator, towards making the overall search efficient [Goldberg, 1989]. Traditionally, the mutation operators modify one or more variables of a given solution with a *probability*, while the extent of that modification is controlled through an *index*. To avoid the a priori fixation of these parameters, which may deter an efficient search, some studies have opted for integrating reinforcement learning (RL) into EMO algorithms. One of such studies include NSGA-RL [Bora et al., 2019], where suitable mutation parameters are learnt on-the-fly using RL, individually for each variable, towards a more efficient search. Another such study is RL-NSGA-II [Ren et al., 2019], 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.

Apart from the above, some studies have used learning automaton (LA), a variant of RL, towards guiding the mutation in a more comprehensive manner [Dai et al., 2016, Zhao and Zhang, 2020]. In that, [Dai et al., 2016] learns two probabilities individually for each variable, that correspond to the direction of modification for that variable (towards the lower or upper bound) and the extent of that modification. Alternatively, [Zhao and Zhang, 2020] exploits the RV-based structure of the underlying RV-EMO algorithm, by learning the choice of mutation operator (out of three), individually for each RV.

Unlike the above approaches that attempted learn and adapt the mutation operation, towards a more efficient search, a recent study [Wang et al., 2021] attempted to learn and adapt the search space itself, using principle component analysis (PCA). In that, 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 before their evaluation.

Notably, the PCA-assisted mutation is applied only after a fixed proportion of the maximum allowed generations ($maxgen$).

2.1.4 Through Innovized Repair Operator

The existing innovized repair (IR) operator [Gaur and Deb, 2016, 2017] briefly introduced in Section 1.1.1, is further detailed here. In EMO-IR, EMO algorithm integrated with the IR operator, no learning or repair is executed till 33% of $maxgen$ have passed. In each of the subsequent generations, the mathematical relationships are learnt from the non-dominated solutions of the parent population. The structure of these mathematical relationships is restricted to *power laws*, for example, $x_1^{b_1} x_2^{b_2} = c$, where $\{x_1, x_2\}$ are variables and $\{b_1, b_2, c\}$ are parameters to be learnt. However, these power laws are first converted into linear equations using log-linear modeling, and are then learnt using multi-variate linear regression. If the quality of the learnt relationships is better than a pre-specified threshold value, they are used towards repairing a fraction of the offspring created originally using the natural variation operators of the underlying EMO algorithm. The said repair operation is executed by forcing the offspring's variables to conform to these learnt relationships.

2.2 Non-ML-based Enhancements in EMO

While the ML-based enhancements in EMO algorithms, discussed above, have shown promise towards building efficient EMO algorithms, local search (a non-ML-based enhancement) has proven its effectiveness in solving MOPs with both continuous and discrete search spaces in conjunction with EMO algorithms [Kumar and Singh, 2007, Land and Belew, 1998]. Some such studies, that incorporated a local search into an EMO algorithm, are reviewed below.

Through Local Search

Local search has often been used in conjunction with the EMO algorithms, towards their performance enhancement [Jaskiewicz, 2002, Lara et al., 2010, Murata et al., 2002]. This conjunction has been realized in several ways, including: (i) the use of local search after the EMO run has terminated, and (ii) the use of local search in each generation of the EMO run [Goel and Deb, 2002]. Implementing local search in an EMO algorithm is not straightforward, as the local search usually means searching for the best solution in a local neighbourhood with respect to a single objective

function or the *fitness* function. A number of suggestions have been made in the literature for the fitness function using aggregation functions, including the weighted sum of objectives [Ishibuchi and Narukawa, 2004], achievement scalarizing function (ASF) [Sindhya et al., 2008], and biased weighted sum (BWS) [H. Seada and Deb, 2017].

While the above studies employed a local search to primarily emphasize on convergence locally, a recent study implemented a local search to locally emphasize on both convergence and diversity through the use of hypervolume measure as the fitness function [Zhou et al., 2021]. Another study [Seada et al., 2019] attempted to tackle both convergence and diversity through the use of local search. It involved invocations of local search in three different manners for achieving different goals of EMO, including: (i) local search using BWS seeking convergence of extreme solutions, (ii) local search using ASF seeking solutions on empty RVs, and (iii) local search using ASF seeking a better convergence for poorly converged solutions.

Notably, implementing local search in EMO requires additional solution evaluations, over and above the default solution evaluations of the underlying EMO algorithm. To reduce the computational burden of these additional solution evaluations, a prominent method is to use a surrogate model and deploy the local search based on approximate solution evaluations. Such approaches have been discussed earlier in Section 2.1.1

2.3 Past studies specific to RV-EMO algorithms

The idea of using RVs to assist the search in EMO algorithms was first proposed in [Zhang and Li, 2007]. In that, the basic idea was to decompose an MOP into several scalar optimization problems, and then search for an optimal solution for each subproblem. While searching the best solution for each subproblem, different scalarizing functions could be used, such as, weighted sum (WS), ASF or penalty-based boundary intersection (PBI). The final step was to find the non-dominated solution set from the obtained optimal solutions (one per subproblem), that could offer a reasonable PF-approximation. Based on the above, RV-EMO algorithms have often been referred to as *decomposition* based algorithms in the literature. Moreover, RVs are often referred to as weight vectors, reference directions or preference vectors. Several extensions to these algorithms have been proposed, including: improved scalarizing functions, modified natural variation operators, enhanced replacement procedures, inclusion of dominance principles and adaptation of RVs.

Improved scalarizing functions: the original decomposition study [Zhang and Li, 2007] investigated three scalarizing functions, namely, WS, ASF and PBI. These traditional scalarizing functions have known shortcomings, especially while dealing with a higher number of objectives and a limited population size, as it results in a high improvement region for each RV [Sato, 2014, Wang et al., 2016a]. To address this, several other scalarizing functions have been proposed, including, adaptive penalty scheme (APS) [Yang et al., 2017], subproblem-based penalty scheme (SPS) [Yang et al., 2017], angle penalized distance (APD) [Cheng et al., 2016] and localized weighted sum (LWS) [Wang et al., 2018].

Further, it is known that no single scalarizing function can provide the best performance in all MOPs, which makes the identification of an appropriate function a crucial task [Trivedi et al., 2017]. Towards address this, some adaptive strategies that focus on identifying an appropriate scalarizing function for each RV (or subproblem) have been proposed, including, MOEA/D-SS [Ishibuchi et al., 2010], MOEA/D-AS [Ishibuchi et al., 2009] and MOEA/D-PaS [Wang et al., 2016b]. A relatively recent approach has explored the use of ML for choosing appropriate scalarizing functions on-the-fly [Wu et al., 2019]. In that, the current non-dominated solutions are considered as the training data, and Gaussian process regression is used to learn the characteristics of the estimated PF. Based on this learnt model, the scalarizing functions are selected and the corresponding subproblems are formulated.

Modified natural variation operators: several studies have been performed with the goal of modifying the commonly used natural variation operators, i.e., SBX crossover and polynomial mutation. For instance, the use of differential evolution (DE) as the crossover operator, has demonstrated excellent performance in handling MOPs with complicated PS shapes [Li and Zhang, 2009]. A later study then investigated the influence of using different DE schemes as crossover operator [Huang and Li, 2010]. However, it is well known that no single set of natural variation operators can outperform all other combinations of natural variation operators, across all MOPs. Towards this, the rewards-based adaptive operator selection (AOS) method has often been used to determine the rate of application of these operators on-the-fly [Li et al., 2014a]. AOS has been integrated into several algorithms, including, MOEA/D-FRRMAB [Li et al., 2014a], MOEA/D-UCB [Gonçalves et al., 2015] and mMOEA/D [Shim et al., 2012].

Enhanced replacement procedures: in all RV-EMO algorithms, it is imperative to examine the interrelationship between the solutions and the RVs, depending on their respective locations in the F-space. While, an intuitive idea is to find the solution offering the best scalarizing function

value for each given RV and select it, some recent studies have argued that such an approach may be biased towards convergence [Li et al., 2014b, 2015b]. In that, a stable matching model has been used that: (a) ranks each solution for each RV (or subproblem), to promote convergence, and (b) also, ranks each RV (or subproblem) for each solution, to promote diversity. Hence, the proposed model is capable of linking each RV to one single solution, such that a balance between convergence and diversity can be maintained [Li et al., 2014b].

Inclusion of dominance principles: some studies have extended the idea of decomposition based algorithms to the dominance based algorithms, resulting into a new category. In this thesis, such algorithms are considered a part of RV-EMO algorithms. In these algorithms, the basic idea is to rely on one or more dominance principles (instead of scalarizing functions) to emphasize convergence, while utilizing the RV-based architecture to emphasize diversity. Such algorithms include: NSGA-III [Deb and Jain, 2014, Jain and Deb, 2014], θ-DEA [Yuan et al., 2016] and MOEA/DD [Li et al., 2015a].

Adaptation of RVs: reportedly, the performance of RV-EMO algorithms strongly depends on the PF shape [Ishibuchi et al., 2017]. In that, the performance of known RV-EMO algorithms, including NSGA-III and θ-DEA, is shown to deteriorate on MOPs with irregular PF shapes. To address this, several approaches for adapting the RVs on-the-fly have been proposed [Ma et al., 2020], including, MOGLS [Ishibuchi and Murata, 1998], RVEA* [Cheng et al., 2016], and GP-A-NSGA-III [Masood et al., 2017]. Some ML-assisted approaches that focus on RV adaptation are discussed below.

Some approaches have focused on using the Gaussian process regression to learn the population distribution in the intermediate generations of an algorithmic run [Wu et al., 2017, 2019]. Once the model is learnt, several points can be sampled randomly, out of which the inferior samples (with large prediction variances) can be deleted and a subset from the remaining ones can be selected based on diversity in F-space. The notable limitations of such approaches include: (a) each objective function must be continuous, which hampers their applicability in real-world problems, and (b) they can be easily misguided by the poorly converged solutions from the intermediate generations [Ma et al., 2020]. Moreover, a recent approach employs incremental learning of RVs to simultaneously: (a) generate denser RVs close to the valid RVs (RVs with at least one solutions associated with them), and (b) eliminate the invalid RVs (RVs that have no solutions associated with them, across several consecutive generations) [Ge et al., 2019].

2.4 Research Gaps

In wake of the ML-based and non-ML-based enhancements in EMO algorithms reviewed in this chapter, some key research gaps have been highlighted below.

- Lack of exploitation of the inter-generational evolution trend towards achieving a better convergence: EMO algorithms are known to follow the Markov property, i.e., the evolution of the population is dependent only on the current state, not on the past history. Even though the RL-based efficient mutation approaches learn through good mutation operations, across the generations, the inter-generational evolution trend is only partially captured since the mutation is only one of the EMO's natural variation operators.

- Lack of approaches focusing on diversity improvement without requiring additional solution evaluations: notably, some approaches have explicitly focussed on diversity improvement [Seada et al., 2019, Zhou et al., 2021]. However, these approaches involve the implementation of a local search, that requires additional solutions over and above the default offspring evaluations of the base EMO algorithm.

- Lack of clarity on determining an appropriate timing to initiate learning and to terminate the EMO run: several studies support the claim that it might not be efficient to learn since the first generation of the EMO run, and rather should initiate learning at an intermediate generation [Gaur and Deb, 2017, Wang et al., 2021]. In that, [Gaur and Deb, 2017] suggested to initiate learning after 33% of $maxgen$ generations have passed, whereas [Wang et al., 2021] introduced a new parameter $r \in [0, 1]$ and suggested to initiate learning after $r \times maxgen$ generations have passed. While such an a priori fixation of r might be a non-trivial task, it points to a graver question of how to decide $maxgen$ a priori, especially while solving real-world problems with unknown characteristics.

2.5 Aim and Objectives of the Thesis

In the wake of the literature review and research gaps highlighted above, the aim of this thesis is *to develop a generic and practicable ML-based framework to assist in the performance enhancement of RV-EMO algorithms*. This aim is through the following objectives:

1. development of a ML based framework, that invokes:

   (a) an innovized progress operator (IP2) to utilize the inter-generational solutions along an RV-EMO run, for convergence enhancement.

   (b) an innovized progress operator (IP3) to utilize the intra-generational solutions along an RV-EMO run, for diversity enhancement.

   (c) a unified innovized progress operator (UIP) to utilize inter-generational and intra-generational solutions along an RV-EMO run, for convergence and diversity enhancement, conjunctly.

2. embedding generality in the proposed framework by avoiding adhoc decisions on critical aspects, including, when to initiate learning, and how frequently to learn.

3. embedding practicability in the proposed framework by avoiding any extra solution evaluations, compared to the base RV-EMO (without ML assistance).

Given the aim and objectives cited above, it is important to acknowledge that the performance enhancements offered by the IP2, IP3, or the UIP operator could be interpreted, in terms of:

1. Improvement in the *quality* of PF-approximation: this is realistic in situations where the base RV-EMO 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 proposed operators help 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-EMO run sufficiently long, is able to approximate the PF well. In such a scenario, the proposed operators could only speed-up the PF-approximation.

Chapter **3**

IP2 Operator for Convergence Enhancement

It has been highlighted in Chapter 1 that the RV-EMO architecture provides the scope for *mapping* of *inter-generational* solutions *along* the RVs in F-space, enabling a *mapping* of their underlying X-vectors, and eventually *learning* of efficient search trajectories in X-space. In this background, this chapter presents the IP2 operator, designed for convergence-enhancement in RV-EMO algorithms. It includes an ML-based approach that: (a) *maps* the solutions from the earlier generations of an RV-EMO run to the selected best solutions till current generation in F-space, *along* the RVs; (b) *learns* the directional improvements (in X-space) through the mapping using an ML model; and (c) utilizes the learnt ML model in the same generation to *advance* a proportion $\mathcal{P}^{\text{IP2}}$ of the offspring (in X-space), originally created using natural variation operators. This advancement, towards a better convergence, is referred to as *progression*. Notably, this creation of offspring solutions using natural variation operators and their subsequent advancement, has been referred to as the production of pro-convergence offspring Q^{IP2}, earlier in Chapter 1. In that, the justification for producing only $\mathcal{P}^{\text{IP2}} = 50\%$ pro-convergence offspring (through advancement) in a particular generation has also been provided, which guarantees a dominant contribution of Q^{V} across all generations of an RV-EMO-IP2 run, as symbolically depicted in Figure 3.1.

The remainder of this chapter is organized as follows: Section 3.1 describes the proposed IP2 operator, followed by an outline of its integration with NSGA-III, an RV-EMO algorithm, in Section 3.2. Section 3.3 discussed the computational complexity of the IP2 operator, followed by a discussion on the experimental setup towards demonstrating the efficacy of the IP2 operator in Section 3.4. Finally, the results and related discussions are presented in Section 3.5.

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

Figure 3.1. Symbolic depiction of the degree of IP2 operator's contribution to convergence and diversity, over an entire run of RV-EMO-IP2. The offspring created through natural variation operators Q^{V} do not impose any explicit preference for either convergence or diversity.

3.1 Proposed IP2 Operator for Convergence Enhancement

It has been highlighted above that the IP2 operator attempts to capture the directional improvements in the search space through inter-generational solutions, to help the offspring advance effectively towards a better convergence. This is realized through three modules, including: *Training-dataset construction*, *ML Training*, and *Offspring's Advancement*. The design and implementation of these modules are detailed in the following subsections.

3.1.1 Training-dataset Construction Module

At any generation t of the RV-EMO algorithm, the *training-dataset* is constituted by the *mapping* between members of an *input-archive* A_t and the members of a *target-archive* T_t. The process of constituting and updating A_t and T_t, and their members' *mapping* is presented below. Towards it's prerequisite terminology, let P_t (sized N) be the parent population; Q_t (sized N) be the offspring population; $\mathcal{R}$ (sized N) be the RV set; and t_{past} be a user-defined parameter that represents the number of past generations to be involved in the composition of A_t.

3.1.1.1 Input-archive Composition and Update

At any generation t, the *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 search space, and (b) such a *learning* could be utilized to help some of the current offspring advance or *progress* more effectively. In this spirit, $A_t = \{P_{t-t_{\mathrm{past}}}\} \cup \{Q_{t-t_{\mathrm{past}}}, Q_{t-t_{\mathrm{past}}+1}, \ldots Q_{t-1}\}$. Notably:

1. first both the parents and offspring in the $(t - t_{\mathrm{past}})^{\mathrm{th}}$ generation are included to account

for maximum diversity without incorporating any duplicate solutions (since the parents and offspring in any generation are distinct).

2. in addition, only the offspring from the $(t - t_{\text{past}} + 1)^{\text{th}}$ until the $(t-1)^{\text{th}}$ generation are included, while the corresponding parents are excluded. This is done to avoid duplicity of (parent) solutions, since in any particular generation, not all parents may replaced by the offspring. This duplicity would be more prevalent in the later generations of an RV-EMO-IP2 run, compared to earlier generations.

Given its composition, A_t naturally gets updated, with every increment in the t counter.

3.1.1.2 Target-archive Composition and Update

As indicated above, the *target-archive* T_t is intended to serve as a set of representative solutions, onto which the solutions from previous generations (A_t) could be mapped to provide a basis for ML training and its subsequent use. In this spirit, T_t is defined as a set of N target solutions obtained along the N RVs, *till the* t^{th} *generation*. This poses three pertinent questions:

1. how to initialize the *target-archive*, of size N, such that one target solution gets associated with each RV.

2. how to determine the potential targets from the parents in a subsequent generation t, namely, P_t.

3. how to update the existing *target-archive*, namely, T_{t-1}, by incorporating the potential targets from P_t. This, in effect, amounts to determining the best N target solutions *till the* t^{th} generation.

The pre-requisite for initialization of the *target-archive* T_t (at $t = 1$) and its subsequent update (at $t \geq 2$), is the normalization of the parent population P_t in the F-space. This normalization can be achieved with the help of ideal ($Z^{\text{ideal}}_{[1\times M]}$) and nadir ($Z^{\text{nadir}}_{[1\times M]}$) points[3], using the formulation given below, where M is the number of objectives.

$$\bar{f}_m(X) = \frac{f_m(X) - Z^{\text{ideal}}_m}{Z^{\text{nadir}}_m - Z^{\text{ideal}}_m}, \ \forall\, m \in \{1, 2, \ldots, M\} \tag{3.1}$$

[3]Majority of the RV-EMO algorithms (including NSGA-III [Deb and Jain, 2014] and MOEA/D [Zhang and Li, 2007]) rely on the normalization of the F-space, and have to approximate the ideal and nadir points as a pre-requisite. Those approximated ideal and nadir points can be used directly by the IP2 operator.

Then, T_t can be initialized by associating one member of P_t *with each RV*, as its target. For coherence, the criterion for such an association, is made to coincide with the criterion for association adopted by the base RV-EMO algorithm (that the IP2 operator is to be integrated with). For instance, given an RV, an RV-EMO algorithm such as NSGA-III [Deb and Jain, 2014] would associate the non-dominated solution offering minimum perpendicular distance (PD). In contrast, another RV-EMO algorithm, such as MOEA/D [Zhang and Li, 2007], would associate the solution offering minimum ASF or PBI value. These metrics define the criteria for identifying the best solution (in the normalized F-space) for a given RV, as discussed earlier in Section 2.3 (Chapter 2). For ease of reference, the formulations of these metrics are given below.

$$\begin{aligned}
\text{PD:}\quad V &= \|(\bar{F}_{(i)} - \mathcal{R}_{(j)}^T \bar{F}_{(i)} \mathcal{R}_{(j)} / \|\mathcal{R}_{(j)}\|^2)\| \\
\text{ASF:}\quad V &= \max_{m\in[1,M]}\{\bar{F}_{(i),m}/\mathcal{R}_{(j),m}\} \\
\text{PBI:}\quad V &= d_1 + \theta d_2, \text{ where} \\
&\quad d_1 = \|\bar{F}_{(i)}^T \mathcal{R}_{(j)}\| / \|\mathcal{R}_{(j)}\|;\ d_2 = \|\bar{F}_{(i)} - d_1(\mathcal{R}_{(j)}/\|\mathcal{R}_{(j)}\|)\|
\end{aligned} \tag{3.2}$$

In any subsequent generation t, the following methodology is used to determine the potential targets for the RVs. It may be noted that these RVs are common to both the IP2 operator and the underlying RV-EMO algorithm. Given this, *for each solution in* P_t, the relevant metric (PD[4] or PBI or ASF or any other, as used by the underlying RV-EMO algorithm) is computed with respect to each RV. Subsequently, each solution in P_t is associated as the potential target for the RV offering minimum/best metric value. Clearly, some RVs may get associated with multiple potential targets, while some may remain unassociated.

At any generation t ($t \geq 2$), the existing target archive T_{t-1} and the potential targets from P_t, are available. In this situation, the task of determining the best N target solutions *till the* t^{th} *generation*, reduces to updating of T_{t-1} by accounting for the potential targets from P_t. In such a case, each RV may have two contenders for the target: (a) one, in the form of the existing target member from T_{t-1}, and (b) another, in the form of the potential-target from P_t. Again, the contender which fares better in the metric adopted by the underlying RV-EMO algorithm is declared as the updated target for that RV. The process of updating T_t using any generic metric (PD or PBI or ASF) is given in Algorithm 3.1.

Notably, in lines 8–12 of Algorithm 3.1, for a given candidate target solution, first the best RV

[4]While comparing any two solutions, the first comparison is through dominance, followed by PD values. This is only applicable for PD since it does not account for convergence at all.

Algorithm 3.1: `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^{ideal}, nadir point Z^{nadir}

Output: Updated target archive T_t

$\{F^T,\ F^P\} \leftarrow$ Objective function values in $\{T_{t-1},\ P_t\}$

$\{\bar{F}^T,\ \bar{F}^P\} \leftarrow$ Normalized $\{F^T,\ F^P\}$ using Z^{ideal} and Z^{nadir}

$V_{[N\times N]} \leftarrow \emptyset$ `% stores evaluated metric values`

$T_t \leftarrow T_{t-1}$

for i = *1 to* N **do**

  for j = *1 to* N **do**

    $V_{i,j} \leftarrow$ Metric value of $\bar{F}^P_{(i)}$ w.r.t. $\mathcal{R}_{(j)}$

  $V^P \leftarrow \min_{j=1}^{N} V_{i,j}$ `% best value for` i^{th} `solution`

  $I \leftarrow \arg\min_{j=1}^{N} V_{i,j}$ `% Index of RV for` i^{th} `solution`

  $V^T \leftarrow$ Metric value of $\bar{F}^T_{(I)}$ w.r.t. $\mathcal{R}_{(I)}$

  if $V^P < V^T$ **then**

    $T_{t,(I)} \leftarrow P_{t,(i)}$

is identified on the basis of the underlying metric and then, the best solution among the candidate target solution and the existing target solution associated with the identified RV, is selected. Here, the former step ensures that the target solution lies in the neighbourhood of the RV, while the latter step ensures convergence *locally* within that neighbourhood. A similar approach has been proposed earlier in MOEA/D-STM [Li et al., 2014c], where both the steps mentioned above (with a small difference) are adopted to identify the association of one solution per RV. The difference is that in MOEA/D-STM, the latter step identifies the most converged solution from the entire pool of solutions, not just the solutions lying in the neighbourhood of a particular RV (as done in Algorithm 3.1).

Figure 3.2 captures a realistic scenario for any generation t ($t \geq 2$), where, 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) none, unique, or multiple potential targets from members in $P_t = \{s_1, s_2, \ldots, s_6\}$. Assuming that the IP2 operator is to be integrated into NSGA-III, that uses a dominance check followed by PD, the update of the target-archive entails the following:

1. for each of $\mathcal{R}_1$, $\mathcal{R}_3$, and $\mathcal{R}_6$, there is one associated target from T_{t-1}, and one potential target from P_t. As evident from the figure, the latter emerge as the updated targets, as they dominate the original targets from T_{t-1}.

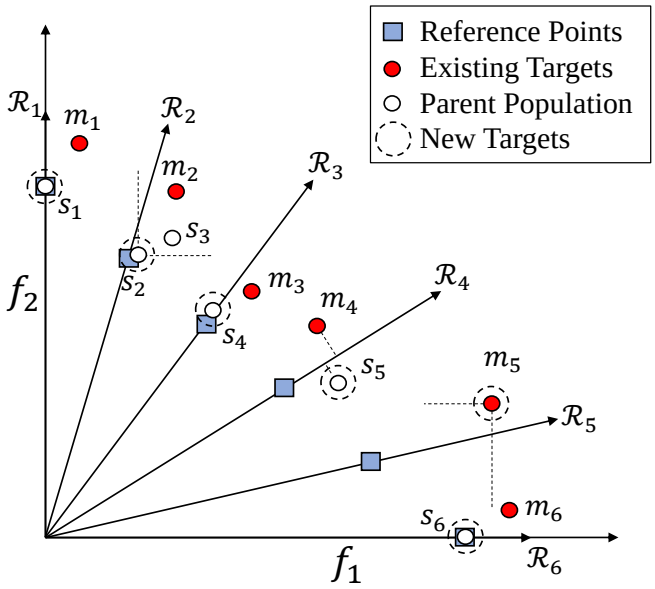


Figure 3.2. A schematic for identifying target solutions along each RV, using PD (with a dominance check, as applicable for integration with NSGA-III) in a two-objective space. Notice how a dominated solution (m_5) is considered as a target point for a poorly represented RV ($\mathcal{R}_5$) for encouraging possible creation of points around that RV.

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 as 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 better (lower) value of PD.

4. for $\mathcal{R}_5$, the associated target solution from T_{t-1} is m_5, however, there are no potential target solutions 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.

5. the *target archive* T_t, factoring in both P_t and T_{t-1}, becomes $\{s_1, s_2, s_4, s_5, m_5, s_6\}$.

3.1.1.3 Archive Mapping

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 the *training-dataset* D_t. Each solution in A_t is associated to some RV (as discussed above), and with each RV a solution from the T_t is associated (as

Algorithm 3.2: `Archive_Mapping` (A_t, T_t, $\mathcal{R}$, Z^{ideal}, Z^{nadir})

Input: Input archive A_t, target archive T_t, set of RVs $\mathcal{R}$, ideal point Z^{ideal}, nadir point Z^{nadir}

Output: Training Dataset D_t

$F^A \leftarrow$ Objective function values in A_t

$\bar{F}^A \leftarrow$ Normalized F^A using Z^{ideal} and Z^{nadir}

$D_t \leftarrow \emptyset$

$V_{[1\times N]} \leftarrow \emptyset$ `% stores evaluated metric values`

for $i = 1$ *to* $sizeof(A_t)$ **do**

  for $j = 1$ *to* N **do**

    $V_j \leftarrow$ Metric value of $\bar{F}^A_{(i)}$ w.r.t. $\mathcal{R}_{(j)}$

  $I \leftarrow \arg\min_{j=1}^{N} V_j$

  $input \leftarrow X$ vector of $A_{t,(i)}$; $target \leftarrow X$ vector of $T_{t,(I)}$

  $D_t \leftarrow D_t \bigcup [input, target]$

discussed above). Hence, the involved RV provides a basis for association of each solution in A_t with a particular solution in T_t. For instance, 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 search space, only the variable vectors (X) of these solutions are stored in D_t. In the context of the above example, the variable vectors of a_i and m_k together form one $input$-$target$ sample for D_t. This process, summarized in Algorithm 3.2, effects the capture of pertinent search directions in X-space, guided by the improvements *along* the RVs in F-space.

Notably in Figure 3.2, m_5 symbolizes a larger issue of maintaining the *convergence-diversity tradeoff*. Since m_5 is dominated by another target member (s_6), it characterizes poorer convergence. However, being the best-representative for $\mathcal{R}_5$ till the t^{th} generation, m_5 marks a suitable choice for maintaining diversity. In this background, the following options are open:

1. Both $\mathcal{R}_5$ and m_5 be dropped, implying that no solutions in A_t associated with $\mathcal{R}_5$ contribute to D_t: this option would fail to capture any information about the solutions associated to $\mathcal{R}_5$ in the *training-dataset*, making it an unsuitable choice.

2. $\mathcal{R}_5$ be kept while m_5 is dropped, implying that solutions in A_t associated with $\mathcal{R}_5$ can be mapped to a target associated with a nearby RV, say s_6 (more converged than m_5): this option would amount to pursuing better convergence at the cost of loss in diversity.

3. Both $\mathcal{R}_5$ and m_5 be kept, implying that the solutions in A_t associated with $\mathcal{R}_5$ are mapped to m_5: this option would amount to attaching importance to diversity preservation during offspring's advancement even at the cost of poorer convergence.

In this thesis, a judicious choice in favour of the third option has been made, guided by the following rationale: (a) diversity preservation is crucial, especially in the early RV-EMO generations, and (b) the marginal compromise in convergence can always be overcome during subsequent generations as long as all regions of the search space are equitably explored.

3.1.2 ML Training Module

The goal here is to train an ML model, here random forest (RF), so it learns such directional improvements in the search space that define the transition of $input$ solutions to their respective $target$ solutions. Ideally, any multi-output regression method can be used as an alternative to RF. Some potential alternatives include artificial neural network (ANN), gradient boost, XG boost, least angle regression and support vector regression. Since the primary motive here is to provide a proof of concept that ML methods could be used in an RV-EMO algorithm for creation of pro-convergence offspring solutions, the scope in this thesis has been restricted to providing the proof-of-concept with one of the ML methods rather than identifying the best ML method for this purpose (which may also depend on the characteristics of the given MOP).

The ML training module is executed as a two-step process: (a) normalization of the training-dataset using the proposed *dynamic normalization* method, as a pre-training step, and (b) the ML training itself, as presented in Algorithm 3.3. Notably, each time the IP2 operator is invoked, a new ML model is trained, and the last trained ML model is discarded.

3.1.2.1 Dynamic Normalization of the Training Dataset

The *dynamic normalization* of the training-dataset, here, refers to normalization of the variable vectors in each *input-target* sample, using adaptive bounds, for each variable $x_k \in X$. The motivation behind this idea is as follows:

(a) the scales and ranges of different variables, as per the problem, may be different,

(b) at any iteration t, the range explored by a particular (k^{th}) variable $[x_k^{l,t}, x_k^{u,t}]$, a fraction of its total permissible range $[x_k^l, x_k^u]$, may differ from that of another,

Algorithm 3.3: `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 $[x^{\min}, x^{\max}]$

$\{x^{l,t},\ x^{u,t}\} \leftarrow$ Minimum and Maximum of each variable in D_t

$x^{\min},\ x^{\max} \leftarrow \emptyset,\ \emptyset$ `% bounds for dynamic normalization`

for k = *1 to* n_{var} **do**

  $x_k^{\min} = 0.5(x_k^{l,t} + x_k^l)$

  $x_k^{\max} = 0.5(x_k^{u,t} + x_k^u)$

Normalize D_t using $x^{\min}$ and $x^{\max}$ as bounds

$ML \leftarrow$ Trained ML model using D_t

(c) the normalization of variables also promises to *even-out* each's contribution to the loss/error function *Mean Squared Error* (MSE), as used in this thesis.

Given the above, the *dynamic normalization* of any variable k is given below:

$$\bar{x}_k = \frac{x_k - x_k^{\min}}{x_k^{\max} - x_k^{\min}}, \tag{3.3}$$

where

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

This normalization, which gives equal importance to a variable's absolute and current range, is employed twice in a generation:

(a) to normalize the *training-dataset* before the ML training, and

(b) to normalize each offspring before advancing it (in the subsequent *offspring's advancement* module), and then to de-normalize each advanced offspring back to the original search space (in $x_k \in [x_k^l, x_k^u]\ \forall\ k \in \{1, 2, \ldots, n_{\text{var}}\}$).

3.1.2.2 Training an RF Model

Once the normalization is executed, as discussed above, an RF model is trained on the normalized training-dataset. In that, it requires three critical settings–the number of trees (N_{tr}); the number of variables/features considered while splitting a node (N_{feat}); and the splitting criterion. Considering that the training-dataset D_t is constituted by N_{sam} training samples, the parameters are

fixed as: $N_{\mathrm{tr}} = N_{\mathrm{sam}}$ and $N_{\mathrm{feat}} = n_{\mathrm{var}}$. The splitting criterion used is MSE, and the rest of the RF settings are kept as default[5].

3.1.3 Offspring's Advancement Module

In this module, the trained ML model is applied directly to advance the offspring solutions, without their prior *evaluation*. This module is executed as a four-step process, where each step advances/alters the offspring's variable vector. In this study, the act of "advancement" towards better convergence is referred to as "progression", and beyond this point, the "advanced offspring solutions" are referred to as the "progressed offspring solutions" (X^{pg}). The constitutive steps of offspring's advancement are detailed below.

3.1.3.1 Selection and Progression

Once the offspring solutions Q_t are created using the natural variation operators of the base RV-EMO algorithm on P_t, a proportion $\mathcal{P}^{\mathrm{IP2}}$ of them are randomly selected ($\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$ offspring), and the *trained* ML model is used to advance these selected offspring. In that, the following aspects are notable.

- Since the ML model is trained on the normalized dataset, the offspring solutions created through natural variation operators are first normalized, then advanced using the ML model, and then denormalized, as discussed earlier in Section 3.1.2.1.
- The random selection of $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$ offspring is made to avoid their evaluation *a priori*.

3.1.3.2 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 that x_k is pushed to its respective extreme at the time of progression. However, during the initial or even intermediate generations of the underlying RV-EMO algorithm, the training dataset may predominantly contain x_k values away from its extremes. Hence, the trained ML model may be biased against the extreme values for x_k. Consequently, if a natural offspring

[5]The RF Regressor used in this study has been taken from the Scikit-learn implementation (for python). Link: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

(created through natural variation operators) possessing a near-extreme value for x_k is subjected to progression using the ML model, it is likely that x_k gets pushed in the direction opposite 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 the variables having values very close to their extremes be barred from progression. From an implementation perspective, where the entire variable vector of the selected offspring is subjected to the trained ML model, without any direct control on individual variables in the vector (some $x_k \in X$), it is imperative that the progression be undone for those variables which, prior to progression, had values in very close proximity to their extremes. To this effect, it is proposed that in any progressed offspring, the original values be restored for all such variables which, prior to progression, had values within 1% of either of their extremes.

3.1.3.3 Jutting the advanced/progressed offspring

Understandably, the ML model *learns* the directional improvements that define the transition of the solutions from previous generations to the best solution till the current generation, along each RV. The current offspring's advancement 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:

$$X^{\text{jpg}} = X + \eta \times (X^{\text{pg}} - X), \tag{3.4}$$

where X and X^{pg} mark the original and the progressed offspring, respectively, and X^{jpg} represents the consequent jutted offspring. Notably, $\eta = 1$ leads to the originally progressed offspring ($X^{\text{jpg}} = X^{\text{pg}}$), while $\eta > 1$ leads to a different offspring ($X^{\text{jpg}} \neq X^{\text{pg}}$). 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 largely alleviated through jutting.

3.1.3.4 Boundary Repair

In the EMO domain, there are several common methods to prevent an operation from setting a variable's value 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 inwards, proportionally as much as it was outside the bound ("reflection"). While the first method may deteriorate the population's diversity in the search space since several offspring might end up with the same variable values (their respective bound), the second method may deteriorate the search efficiency by introducing random variable values into several offspring. In the wake of above, the IP2 operator incorporates a sophisticated variant of the third method [Padhye et al., 2013]. In that, any variable $x_k \in X^{\text{jpg}}$ that goes outside its permissible bounds ($[x_k^l, x_k^u]$) is mapped to an inner value based on an Inverse Parabolic Spread Distribution.

The overall process of the offspring's advancement, as described through the four steps above, is summarized in Algorithm 3.4.

Algorithm 3.4: `Progression`(Q_t,η,[$x^{\min}, x^{\max}$],[x^l, x^u], ML, $\mathcal{P}^{\text{IP2}}$)

Input: Original offspring Q_t, jutting parameter η, bounds from Algorithm 3.3 [$x^{\min}, x^{\max}$], variable bounds in problem definition [x^l, x^u], proportion of offspring to be advanced $\mathcal{P}^{\text{IP2}}$

Output: Progressed offspring Q_t

$I \leftarrow$ Randomly selected $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ offspring from Q_t
for $X \in I$ **do**
  $\bar{X} \leftarrow$ Normalize X using $x^{\min}$ and $x^{\max}$
  $\bar{X}^{\text{pg}} \leftarrow ML(\bar{X})$
  $X^{\text{pg}} \leftarrow$ Denormalize $\bar{X}^{\text{pg}}$ using $x^{\min}$ and $x^{\max}$
  for *each variable* $k \in [1, n_{\text{var}}]$ **do**
    Restore x_k^{pg} to x_k if x_k lies in 1% vicinity of its bounds
  $X^{\text{jpg}} \leftarrow$ Jutted offspring from X, X^{pg} and η `% equation 3.4`
  Boundary Repair on X^{jpg}
  Replace the original offspring in Q_t by X^{jpg}

3.2 Integration of IP2 operator into NSGA-III

This section outlines the integration of the IP2 operator with NSGA-III, leading to NSGA-III-IP2. This integration, summarized in Algorithm 3.5 is generic in nature, and can be extended to

any other RV-EMO algorithm.

Algorithm 3.5: Generation t of NSGA-III-IP2

Input: Ideal point Z^{ideal}, Nadir point Z^{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_{past} used in A_t, frequency of progression $t^{\text{IP2}}_{\text{freq}}$, generation of last progression $t^{\text{IP2}}_{\text{pg}}$, jutting parameter η, number of survived offspring in $(t-1)^{\text{th}}$ generation $N^{\text{survived}}_{t-1}$

Output: P_{t+1}, T_t, $t^{\text{IP2}}_{\text{pg}}$, $t^{\text{IP2}}_{\text{freq}}$, A_{t+1}, N^{survived}_t

1 $T_t \leftarrow$ `Update_Target_Archive`$(P_t, T_{t-1}, \mathcal{R}, Z^{\text{ideal}}, Z^{\text{nadir}})$

2 **if** *Non_Dominated*(P_t) & $t - t^{\text{IP2}}_{\text{pg}} = t^{\text{IP2}}_{\text{freq}}$ **then**

3 $\quad flag \leftarrow$ *True*

4 **else**

5 $\quad flag \leftarrow$ *False*

6 **if** $flag =$*True* **then**

7 $\quad D_t \leftarrow$ `Archive_Mapping`$(A_t, T_t, \mathcal{R}, Z^{\text{ideal}}, Z^{\text{nadir}})$

8 $\quad ML, [x^{\text{min}}, x^{\text{max}}] \leftarrow$ `Training`$(D_t, [x^l, x^u])$

9 $Q_t \leftarrow$ `Variation`(P_t)

10 **if** $flag =$ *True* **then**

11 $\quad Q_t \leftarrow$ `Progression`$(Q_t, \eta, [x^{\text{min}}, x^{\text{max}}], [x^l, x^u],$ ML, $\mathcal{P}^{\text{IP2}})$

12 $\quad t^{\text{IP2}}_{\text{pg}} \leftarrow t$

13 Evaluate Q_t

14 $A_{t+1} \leftarrow \left(A_t \cup Q_t \cup P_{t+1-t_{\text{past}}}\right) \setminus \left[P_{t-t_{\text{past}}} \cup Q_{t-t_{\text{past}}}\right]$

15 $P_{t+1} \leftarrow$ `Survival_selection`$(P_t \cup Q_t)$

16 $N^{\text{survived}}_t \leftarrow sizeof(Q_t \cap P_{t+1})$

17 **if** $t = t^{\text{IP2}}_{\text{pg}}$ **then**

18 $\quad$ **if** $N^{\text{survived}}_t > N^{\text{survived}}_{t-1}$ **then**

19 $\quad\quad t^{\text{IP2}}_{\text{freq}} \leftarrow t^{\text{IP2}}_{\text{freq}} - 1$

20 $\quad$ **if** $N^{\text{survived}}_t < N^{\text{survived}}_{t-1}$ **then**

21 $\quad\quad t^{\text{IP2}}_{\text{freq}} \leftarrow t^{\text{IP2}}_{\text{freq}} + 1$

Notably, Algorithm 3.5 represents any intermediate generation t of NSGA-III-IP2, and involves a new parameter, namely, $t^{\text{IP2}}_{\text{freq}}$, that specifies the number of generations between two successive progressions. First, the *target archive* T_t is updated using Algorithm 3.1 (line 1, Algorithm 3.5). Then, if the entire population has become non-dominated, and if $t^{\text{IP2}}_{\text{freq}}$ generations have passed after the last invoked progression in generation $t^{\text{IP2}}_{\text{pg}}$, the IP2 operator is invoked (lines

2–5, Algorithm 3.5). The non-domination check is crucial towards ensuring that before the IP2 operator is invoked for the first time, a reasonable diversity in the F-space has been achieved, to effect better learning. If yes, the $flag$ is marked *True*, following which the *training dataset* D_t is generated using Algorithm 3.2, and the ML training is done using Algorithm 3.3 (lines 6–8, Algorithm 3.5). Subsequently, offspring are produced using the variation operators (crossover and mutation in this case), followed by (i) progression of the offspring using Algorithm 3.4, (ii) their evaluation, (iii) update of the *archive* A_t, and (iv) NSGA-III's survival selection (lines 9–15, Algorithm 3.5). Towards the end, the count of offspring that survived to the next generation (N_t^{survived}) is estimated (line 16, Algorithm 3.5). If this count has improved compared to the previous generation, implying a good performance of the IP2 operator, then $t_{\text{freq}}^{\text{IP2}}$ is reduced by 1, implying a more frequent progression. However, if this count has degraded, implying a poorer performance of the IP2 operator, $t_{\text{freq}}^{\text{IP2}}$ is increased by 1 implying a less frequent progression. Notably, this adaptation of $t_{\text{freq}}^{\text{IP2}}$ is executed only in generations where IP2 is invoked.

3.3 Computational Complexity of IP2 operator

As detailed in the section above, the proposed IP2 operator is constituted by three modules. In the following subsections, time and space complexity of each constituent module has been discussed, followed by its overall summary.

3.3.1 Training-Dataset Construction Module

This module pertains to updating 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 input archive: The process of updating A_t to A_{t+1} involves replacement of $2N$ solutions. These solutions are known, and their selection involves no extra 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 3.1. 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 process of archive mapping is summarized in Algorithm 3.2. It includes $sizeof(A_t) \times N$ computations of the metric (such as ASF, PD, etc.). From the

defined composition of A_t, we have $sizeof(A_t) = N(t_{\text{past}} + 1)$. Hence, the resulting time complexity of the archive mapping step is $\mathcal{O}(MN^2 t_{\text{past}})$.

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

3.3.2 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 $\mathcal{O}(N_{\text{tr}} n_{\text{var}} N_{\text{sam}}^2 \log(N_{\text{sam}}))$, 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 [Louppe, 2015]. Similarly, the worst case space complexity of the RF is $\mathcal{O}(N_{\text{tr}} n_{\text{var}} N_{\text{sam}})$. As per the parameter settings discussed in Section 3.1.2, $N_{\text{tr}} = N(t_{\text{past}} + 1)$, $n_{\text{var}} = n_{\text{var}}$ and $N_{\text{sam}} = N(t_{\text{past}}+1)$. Upon substituting their values and simplifying, the obtained time and the space complexities of the ML training module are $\mathcal{O}(N^3 t_{\text{past}}^3 n_{\text{var}} \log(Nt_{\text{past}}))$ and $\mathcal{O}(N^2 t_{\text{past}}^2 n_{\text{var}})$, respectively.

3.3.3 Offspring's Advancement Module

Evidently, this module (Algorithm 3.4), 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 $\mathcal{O}(n_{\text{var}})$, that are repeated for $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ solutions. Hence, their time complexity is $\mathcal{O}(Nn_{\text{var}})$. However, in the first step of selection and progression, $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ offspring are advanced using the trained RF model. From [Louppe, 2015], the worst-case time complexity for prediction using an RF is $\mathcal{O}(N_{\text{tr}} n_{\text{var}} N_{\text{sam}})$, which is the same as the space complexity of the RF as discussed in the previous subsection. Upon simplifying, the time complexity of a prediction through RF becomes $\mathcal{O}(N^2 t_{\text{past}}^2 n_{\text{var}})$. Since $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ predictions are made (considering progression of only $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ offspring), the resulting time complexity is $\mathcal{O}(N^3 t_{\text{past}}^2 n_{\text{var}})$. 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 complexity and space complexity of each constituent module of the IP2

Table 3.1. Time- and space-complexities of different modules of the IP2 operator

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

operator is summarized in Table 3.1. Evidently, training of an RF has the highest time complexity, while the trained RF model has the highest space complexity.

3.4 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-EMO algorithm(s) and the IP2 operator.

3.4.1 Test-suite

To demonstrate the search efficacy infused by the IP2 operator into an RV-EMO algorithm, several two- and three-objective problems with varying degrees of difficulty have been used. These include: ZDT [Zitzler et al., 2000], DTLZ [Deb et al., 2005] and MaF [Cheng et al., 2017] problems[6] with the following specifications.

- ZDT ($M = 2$): their respective $g(X)$ have been modified to have the PO 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 $\tilde{\text{Z}}$DT in the remainder of this thesis.

- DTLZ and MaF ($M = 3$): the distance variables k have been kept as 20, to make the problems more convergence-hard, as compared to the generally used $k = 5$ or 10.

[6]Notably, the redundant problems such as DTLZ5, DTLZ6 and MaF6 have been excluded. Since only a small number of RVs pass through the PF of these problems, they are ineffective for 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.

3.4.2 Performance Indicators and Statistical Analysis

Hypervolume has been used as the primary performance indicator since it serves as a combined indicator for convergence and diversity. Computing the hypervolume requires a reference point to be specified, which is set as $R_{1\times M} = [1 + \frac{1}{p}, \ldots, 1 + \frac{1}{p}]$ [Ishibuchi et al., 2018], where p is the number of gaps set for the Das-Dennis method [Das and Dennis, 1998] while generating the RVs for RV-EMO. 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 F-space using the theoretical PF extremes, for the latter.
- while the median of the hypervolume values from two algorithms can be compared directly (the higher, the better), these values are subjected to a statistical analysis using the Wilcoxon ranksum test [Wilcoxon, 1945]. 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 F-space. For further insights into the convergence levels in the X-space, the population mean of the $g(X)$ function values has also been reported.

3.4.3 Parameter Settings

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

3.4.3.1 RV-EMO Settings

With an aim to obtain a reasonably sized set of RVs using the Das-Dennis method [Das and Dennis, 1998], 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. Further, the natural variation operators include SBX crossover ($p_c = 0.9$ and $\eta_c = 20$) and polynomial mutation ($p_m = 1/n_{\text{var}}$ and $\eta_m = 20$) for an n_{var} variable problem.

For each test instance, the performance of each RV-EMO algorithm has been assessed over its runs with 31 random seeds. To avoid an arbitrary fixation of the termination generations t_{term}, a

stabilization tracking algorithm [Saxena and Kapoor, 2019] has been included in NSGA-III-IP2. This stabilization tracking algorithm requires a parameter set, kept as $\psi_{\text{term}} \equiv \{3, 50\}$, which suggests the t_{term} for NSGA-III-IP2 on-the-fly. The mean t_{term} determined for NSGA-III-IP2 over 31 runs has been used as the t_{term} for NSGA-III.

3.4.3.2 IP2 Operator Settings

The IP2 operator involves four parameters: $\mathcal{P}^{\text{IP2}}$, t_{past}, $t^{\text{IP2}}_{\text{freq}}$ and η. Here, $\mathcal{P}^{\text{IP2}}$ refers to the proportion of the total offspring (N) advanced using the IP2 operator; t_{past} controls the size of the input archive A_t and the training-dataset D_t; $t^{\text{IP2}}_{\text{freq}}$ controls the invocations of the IP2 operator; and η controls the extent of jutting of the progressed offspring.

$\mathcal{P}^{\text{IP2}} = 50\%$ has been used, as reasoned earlier in Section 1.2 (Chapter 1). Further, $t^{\text{IP2}}_{\text{freq}}$ has been adapted on-the-fly based on the survival of the offspring, as can be observed in Algorithm 3.5. Its initial value is set as 1. To avoid an adhoc fixation of η, it is set as a random value in $[1.0, 1.5]$, implying a maximum of 50% extra advancement along the learnt search directions. While $t^{\text{IP2}}_{\text{freq}}$ and η could be rationally adapted or set, a direct impact of t_{past} on the performance of the IP2 operator is uncertain. Towards this, the behaviour of NSGA-III-IP2 with respect to different values of t_{past}, has been discussed and investigated here briefly.

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 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 $\mathcal{O}(t^3_{\text{past}} \log t_{\text{past}})$. 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 search space) learnt from a longer history of solutions may not be pertinent for subsequent generations.

In the wake of 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. Towards this, a sample parametric study on t_{past} is presented here, on three problems from different test suites, namely, $\tilde{\text{Z}}$DT1, DTLZ1 and MaF1, with $t_{\text{past}} = \{1, 3, 5, 7, 9\}$. The median hypervol-

ume obtained by NSGA-III-IP2 at t_{term} generations determined on-the-fly is shown in Table 3.2. In that, the best-obtained hypervolume and its statistically equivalent results, are marked in bold.

Table 3.2. Median hypervolume obtained by NSGA-III-IP2 with different t_{past} values.

Problem	$t_{\text{past}} = 1$	$t_{\text{past}} = 3$	$t_{\text{past}} = 5$	$t_{\text{past}} = 7$	$t_{\text{past}} = 9$
$\tilde{\text{Z}}$DT1	**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

Interestingly, Table 3.2 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 thesis, 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 data and moderate ML training time, *if* a reasonable population size (N) is used by the underlying EMO algorithm.

3.5 Results and Discussions

This section compares the performance of NSGA-III-IP2 vis-$\grave{a}$-vis NSGA-III, on a wide range of test problems.

3.5.1 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 3.3 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of t_{term} generations. In that, t_{term} 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

Table 3.3. Hypervolume and $g(X)$ based comparison of NSGA-III and NSGA-III-IP2 on benchmark ZDT, DTLZ and MaF problems, at t_{term} generations determined on-the-fly for NSGA-III-IP2 using a stabilization tracking algorithm. In each row, the respective generations (t_{term}) 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_{term}	Median hypervolume			Median $g(X)$			
			NSGA-III	NSGA-III-IP2	p-value	$g(X)\|_{X\in PS}$	NSGA-III	NSGA-III-IP2	p-value
$M=2$	ZDT1	1197	**0.681860**	**0.681860**	8.46E-02	1	1.000732	**1.000427**	2.47E-07
	ZDT2	1280	0.348793	**0.348794**	3.66E-02	1	1.000580	**1.000315**	6.19E-08
	ZDT3	1005	**1.068427**	**1.068537**	8.98E-02	1	1.006285	**1.005028**	4.21E-04
	ZDT4	1768	**0.681859**	**0.681860**	4.10E-01	1	1.000040	**1.000001**	3.59E-07
	ZDT6	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 ⟶		15	**20**	of 21 probs.		8	**18**	of 18 probs.

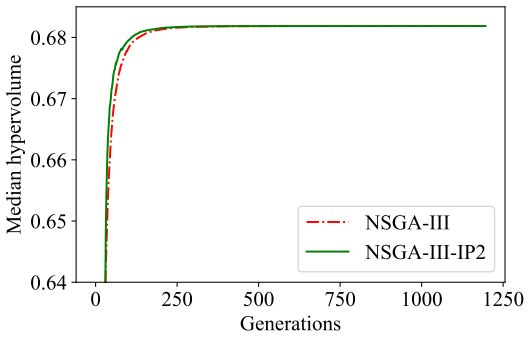
Note: (–) implies that the concerned problem does not have a $g(X)$ function.

(instances, where it is not existent, are marked by '–').

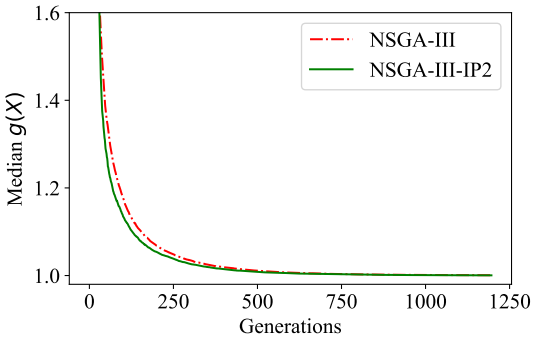
To share more insights into these results, a sample two- and three-objective problem where NSGA-III-IP2 reported statistically better than or equivalent hypervolume measures, than NSGA-III, is dedicatedly discussed below. In addition, the anomalous instance of MaF1 where NSGA-III-IP2 reported statistically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

3.5.2 Insights into a sample two- and three-objective problem

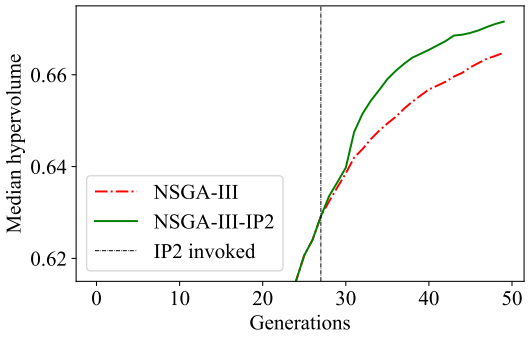
For a sample discussion on a two-objective problem, the $\tilde{\text{Z}}$DT1 problem has been randomly chosen. Figures 3.3a and 3.3b show the generation-wise median hypervolume and median $g(X)$ plots, respectively, among the 31 randomly seeded runs of NSGA-III and NSGA-III-IP2. The termination generation t_{term} had been set as 1197 for NSGA-III, as was determined on-the-fly for NSGA-III-IP2. Interestingly, both hypervolume and $g(X)$ measures suggest that the base



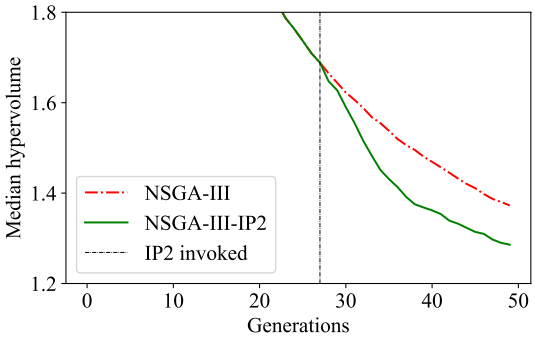
(a) Median hypervolume until t_{term}



(b) Median $g(X)$ until t_{term}



(c) Median hypervolume in early generations



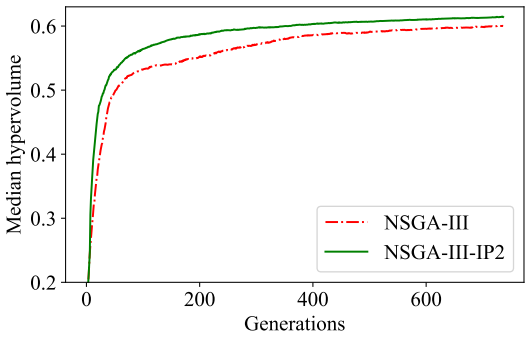
(d) Median $g(X)$ in early generations

Figure 3.3. Results and analysis of the IP2 operator on two-objective $\tilde{\text{Z}}$DT1 problem.

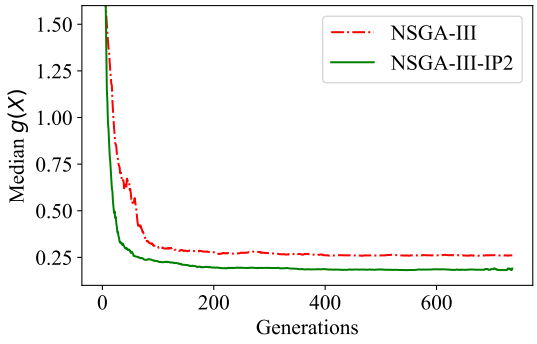
NSGA-III on its own could approximate the PF well (note, $g(X)|_{X \in PS} = 1$). Hence, as per the premise for interpretation of the results, laid earlier (Chapter 2, Section 2.5), the scope of the possible enhancements by the IP2 operator, reduces to *speeding-up* of the PF-approximation. This indeed is the case, as endorsed by Figures 3.3c and 3.3d. 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 on a three-

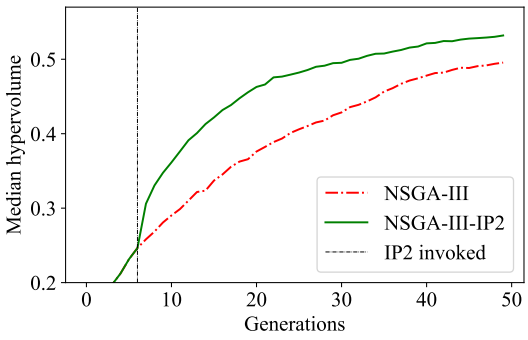
objective problem. Figures 3.4a and 3.4b show the generation-wise median hypervolume and median $g(X)$ plots, respectively, among the 31 randomly seeded runs of NSGA-III and NSGA-III-IP2. The termination generation t_{term} had been set as 737 for NSGA-III, as was determined on-the-fly for NSGA-III-IP2. Evidently, this presents an instance where the base NSGA-III could



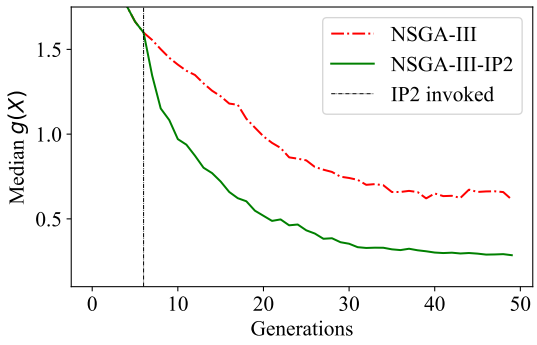
(a) Median hypervolume until t_{term}



(b) Median $g(X)$ until t_{term}



(c) Median hypervolume in early generations



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

Figure 3.4. Results and analysis of the IP2 operator on three-objective MaF12 problem.

not offer a good PF-approximation. This is because, even around/at 737 generations, the hypervolume for NSGA-III could not stabilize, and the corresponding $g(X)$ measures remained far way from the desired $g(X)|_{X \in PS} = 0$. Hence, as per the premise for interpretation of the results, laid earlier (Section 2.5, Chapter 2), such a scenario points to the possibility of improvement in the *quality* of PF-approximation by the IP2 operator. This indeed is the case, as endorsed by:

- better hypervolume and $g(X)$ measures/trend for NSGA-III-IP2 in Figures 3.4a and 3.4b, respectively.
- Figures 3.4c and 3.4d, where NSGA-III-IP2 can be seen to offer superior hypervolume and $g(X)$ measures, right after the underlying IP2 operator is invoked.

While the above discussions explain the general trend of results in Table 3.3, the focus here is to share visual insights on why the IP2 operator is able to enhance the performance of NSGA-III. Towards it, the $\tilde{\text{Z}}$DT1 problem has been chosen, and the capability of RF model trained at 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 the Figure 3.5. In that, nearly half of

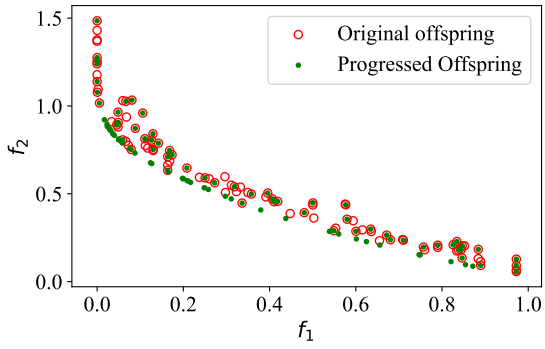


Figure 3.5. Actual Offspring progression at $t = 35$ (a randomly chosen intermediate generation in which the IP2 operator was invoked) to visually depict the capability of the underlying RF model.

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 aligns with the proposed offspring advancement module, wherein only $\mathcal{P}^{\text{IP2}} = 50\%$ 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 crossover 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 crossover-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.

3.5.3 Insights into the MaF1 Problem

As highlighted earlier, MaF1 happened to be the only problem in Table 3.3, 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 fares better in convergence criterion ($g(X)$) but poorer in conjoint convergence-diversity criteria (hypervolume)
- *the loss in diversity corresponding to the use of IP2 is more significant than the corresponding gain in convergence*.

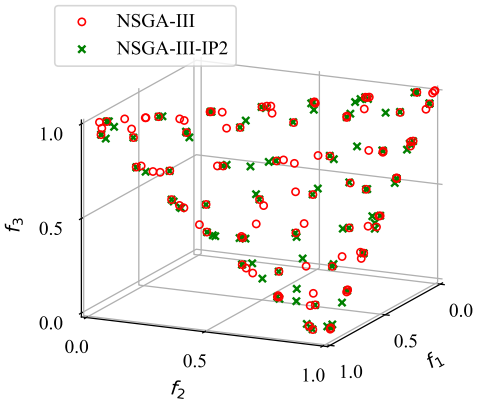


Figure 3.6. NSGA-III and NSGA-III-IP2 populations at t_{term}, for the MaF1 problem ($M = 3$).

The latter stands validated by the Figure 3.6, 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-EMO algorithms, since only a fraction of the underlying RVs pass through the true PF [Ishibuchi et al., 2017].
- the above challenge could be further augmented by the possibility that: (i) some of the $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ offspring that get randomly chosen for progression may have been the *sole* representatives of some of the RVs that pass through the true PF. For the sake of discussion, let the subset of such RVs be referred to as $\hat{\mathcal{R}}$. and (ii) post-progression, some of such (sole representatives) offspring advance to RVs other than those included in $\hat{\mathcal{R}}$. This implies that some of the members of $\hat{\mathcal{R}}$ may not be associated with any solution, adversely impacting

PF coverage/diversity. This additional challenge does not arise in the case of the base NSGA-III, where the sole representative may be selected.

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 dealt with in subsequent chapters.

3.5.4 IP2 Settings vis-à-vis Convergence-Diversity Balance and Risk-Rewards Tradeoff

In view of the results presented in the experimental investigation above, it is fair to infer that NSGA-III-IP2 effectively addresses the key considerations of convergence-diversity balance and risk-reward tradeoff, in majority of the considered problems. Such effective management, despite using a pro-convergence operator, could be attributed to the following settings that allow a dominant share of Q^{V} across the generations.

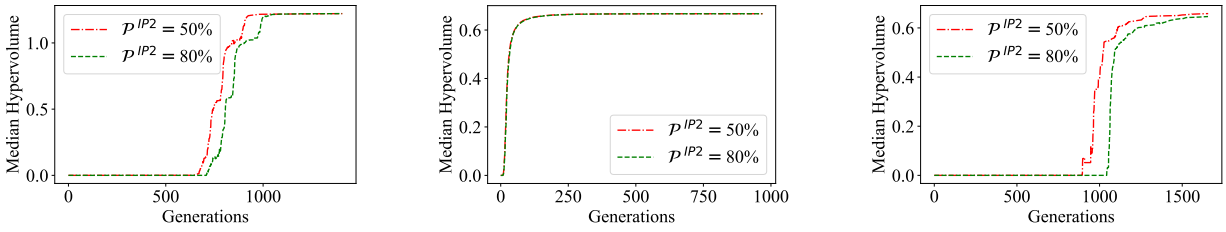
- In any generation of NSGA-III-IP2 run where the IP2 operator is invoked, only $\mathcal{P}^{\mathrm{IP2}} = 50\%$ pro-convergence offspring are created.
- The invocations of the IP2 operator (governed by $t^{\mathrm{IP2}}_{\mathrm{freq}} \geq 1$) are adaptive, based on the survival rate of the pro-convergence offspring.

In the context of setting $\mathcal{P}^{\mathrm{IP2}}$, following scenarios are possible.

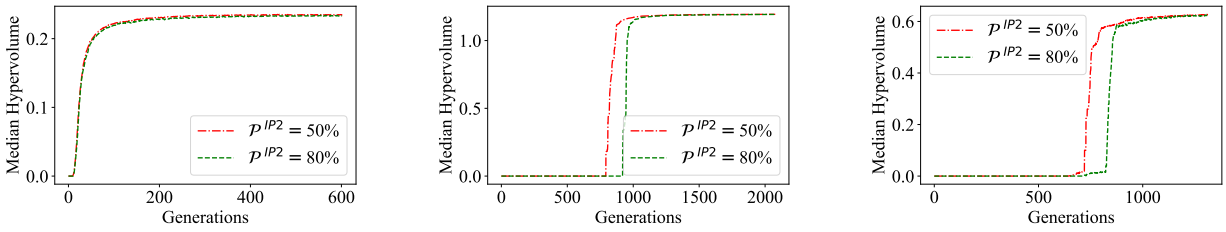
- $\mathcal{P}^{\mathrm{IP2}} < 50\%$: any such setting would inherently address the key considerations mentioned above, since the overall share of Q^{V} would be dominant in each generation as well as across all generations. However, it may lead to more frequent invocations of the IP2 operator compared to $\mathcal{P}^{\mathrm{IP2}} = 50\%$, to overall produce the same number of pro-convergence offspring across all generations. This would lead to more ML model trainings, which is a computationally inefficient choice.
- $\mathcal{P}^{\mathrm{IP2}} > 50\%$: any such setting would not ensure a dominant share of Q^{V} across all generations. As reasoned earlier in Section 1.2 (Chapter 1), this may hamper the management of the above considerations, consequently deteriorating the effectiveness of the IP2 operator.

While any setting of $\mathcal{P}^{\mathrm{IP2}} < 50\%$ does not pose a major implication on the effectiveness of the IP2 operator, any setting of $\mathcal{P}^{\mathrm{IP2}} > 50\%$ does. Hence, it is imperative to investigate the latter scenario. Towards this, the performances of IP2 operator with $\mathcal{P}^{\mathrm{IP2}} = 50\%$ and with

$\mathcal{P}^{\text{IP2}} = 80\%$ (a random value $> 50\%$), have been compared on some sample test problems. The generation-wise median hypervolume plots, from among the 31 randomly seeded runs, are shown in Figure 3.7. In that, the same t_{term} generations have been used as given in Table 3.3. From Figure 3.7, following maybe noted.



(a) DTLZ1 ($M = 3$) (b) DTLZ2 ($M = 3$) (c) DTLZ3 ($M = 3$)



(d) MaF1 ($M = 3$) (e) MaF3 ($M = 3$) (f) MaF4 ($M = 3$)

Figure 3.7. Results of NSGA-III-IP2 with different settings of $\mathcal{P}^{\text{IP2}}$ on some test problems.

- In the case of DTLZ2 and MaF1, both settings of $\mathcal{P}^{\text{IP2}} = 50\%$ and $\mathcal{P}^{\text{IP2}} = 80\%$ lead to attaining similar hypervolume values, implying a similar performance. This is plausible in problems where a reasonable coverage across the true PF has already been obtained before the IP2 operator is invoked for the first time. In such cases, an over-emphasis on convergence by producing a dominant share of pro-convergence offspring, may not impact the overall performance significantly.

- In other cases (DTLZ1, DTLZ3, MaF3 and MaF4), $\mathcal{P}^{\text{IP2}} = 80\%$ leads to a worse hypervolume in the intermediate generations, compared to $\mathcal{P}^{\text{IP2}} = 50\%$. This could be attributed to either or both of: (a) over-skew towards convergence, and (b) excessive reliance on an ML-based operator. However, towards the end generations, the performances with $\mathcal{P}^{\text{IP2}} = 50\%$ and $\mathcal{P}^{\text{IP2}} = 80\%$ are similar. This could be attributed to the adaptive invocations of IP2 operator (through $t^{\text{IP2}}_{\text{freq}}$), that eventually led to lesser frequent invocations of IP2 operator in case of $\mathcal{P}^{\text{IP2}} = 80\%$, compared to $\mathcal{P}^{\text{IP2}} = 50\%$.

Chapter **4**

IP3 Operator for Diversity Enhancement

It has been emphasized earlier that EMO algorithms pursue the dual goals of convergence-to and diversity-across the true PF. In that, diversity needs to be interpreted in terms of the extent of *spread* (coverage across the PF) and *uniformity of distribution* within a given spread. It has also been laid out in Chapter 1, that this thesis aims to propose ML-based operators that could be invoked at intermittent generations of RV-EMO algorithms, so as to advance or create a fraction of the offspring solutions, towards improvement in convergence and diversity. While this has been accomplished in terms of the convergence criterion in the previous chapter, the focus in this chapter is on enhancing diversity.

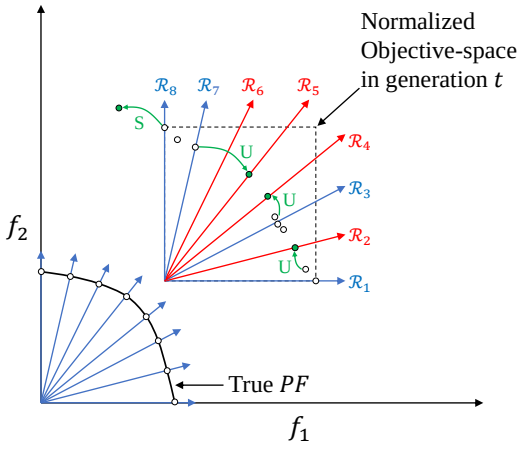


Figure 4.1. A symbolic depiction of how the IP3 operator, aims to advance *some* Parents in a given generation (shown in white circles), to create Offspring (shown in green circles) such that they contribute to expanded *spread* (denoted by S), and improved *uniformity* (denoted by U) through association with some of the otherwise unassociated RVs ($\mathcal{R}_2$, $\mathcal{R}_4$, $\mathcal{R}_5$, and $\mathcal{R}_6$).

It has been highlighted in Chapter 1, that this chapter proposes an IP3 operator which re-

lies on: (a) *learning* the efficient search directions (in X-space) based on a *mapping* of *intra-generational* solutions in F-space, *across* the reference vectors; and (b) utilizing the learnt directions for *creation* of a proportion $\mathcal{P}^{\text{IP3}}$ of the offspring (in X-space) in that generation. The justification for $\mathcal{P}^{\text{IP3}} = 50\%$ has been provided earlier in Section 1.2 (Chapter 1). While the details of the IP3 operator are presented in subsequent sections, some salient features are highlighted upfront, through the Figure 4.1. In that, the parents in an intermediate generation, shown in white circles, are associated only with a few RVs. IP3 operator aims to advance *some* of the parent solutions to create offspring so that they: (i) go beyond the normalized F-space in that generation, contributing to expanded *spread*, and (ii) get associated with *some* of unassociated RVs, contributing to improved *uniformity*. Such an advancement of parents, towards better diversity, is referred to as *progression*.

This example also reveals a significant feature, that the IP3 based offspring ($\lfloor \mathcal{P}^{\text{IP3}} N \rfloor$) are directly created through the advancement of the parents. This is in contrast to the approach adopted for the IP2 operator, where the first N offspring are created through the natural variation operators; and $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ of these are subjected to the IP2 based progression. This difference is rooted in the overarching objective in this thesis, *to avoid extra solution evaluations compared to the base RV-EMO algorithm*. In that:

- the IP2 approach could restrict the number of offspring evaluations to N despite the creation of $\lfloor (1 + \mathcal{P}^{\text{IP2}}) N \rfloor$ offspring (N using natural variation operators, and $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ using the IP2 operator) because the $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ naturally created offspring that were subjected to the IP2 operator could be *randomly chosen*, doing away with the need for their prior evaluation. Such liberty for random choice was, in turn, available because enhanced convergence along any subset of RVs, is acceptable.

- an IP2 like approach for IP3 would entail: (a) creation of N offspring solutions by the natural variation operators, (b) identification of the RVs that remain unassociated with the N parents and N offspring, combined, and (c) creation of IP3 based offspring ($\lfloor \mathcal{P}^{\text{IP3}} N \rfloor$), to cater to the unassociated RVs, besides expansion of spread. Considering that the identification of unassociated RVs (step-(b), above) necessitates the prior evaluation of the naturally created N offspring, this approach would call for a total evaluation of $\lfloor (1 + \mathcal{P}^{\text{IP3}}) N \rfloor$ offspring solutions, undesirable from a practical perspective. Hence, this approach has been avoided in this thesis.

Notably, the salient features highlighted above pertain to those intermittent generations where the IP3 operator is invoked. It must be re-iterated that, in all such generations of RV-EMO-IP3, where the IP3 operator is not invoked, the offspring solutions are solely generated by the natural variation operators, as in the case of base RV-EMO. The above setting guarantees a dominant contribution of Q^{V} across all generations of RV-EMO-IP3, as detailed earlier in Chapter 1, and symbolically depicted in Figure 4.2.

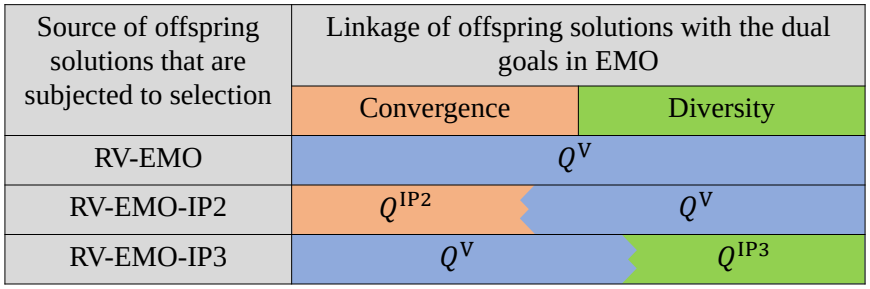


Figure 4.2. Symbolic depiction of the convergence-diversity balance across all generations of RV-EMO-IP3 vis-à-vis RV-EMO-IP2 and base RV-EMO. The offspring solutions created using natural variation operators Q^{V} do not impose any explicit preference for either convergence or diversity.

The remaining chapter is organized as follows: the proposed IP3 operator is detailed in Section 4.1, while its integration with NSGA-III, an RV-EMO algorithm, is presented in Section 4.2. Its computational complexity is highlighted in Section 4.3. Finally, Section 4.4 highlight the experimental settings, leading up to the presentation of results in Section 4.5.

4.1 Proposed IP3 Operator for Diversity Enhancement

Similar to the IP2 operator, the IP3 operator is constituted by three modules, including *Training-dataset construction*, *ML training*, and *Offspring creation*. The design and implementation of these constitutive modules are detailed in the following subsections.

4.1.1 Training-dataset Construction Module

This subsection is further split 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 training-dataset constitution is proposed; and finally, its algorithmic implementation is presented.

4.1.1.1 Deciphering the mapping requirements vis-à-vis the goal of diversity enhancement

It needs to be acknowledged, upfront, that the training-dataset is to be designed in a manner, that the subsequently trained ML model is able to cater to both – expansion of solutions' *spread* and *uniformity* of solutions with a given spread.

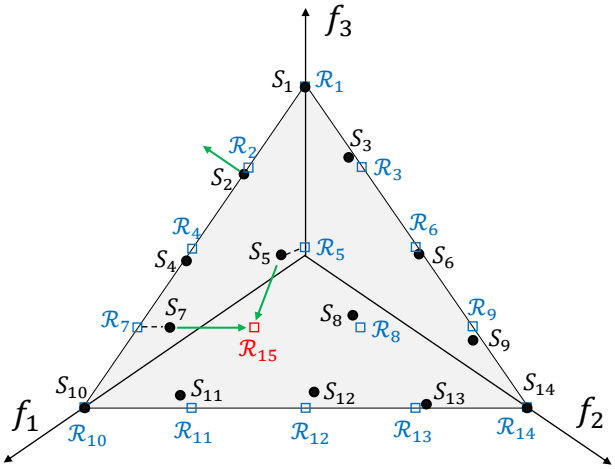


Figure 4.3. Depicting the need for objective-wise mapping, towards training-dataset construction.

Reference to Figure 4.3 reveals that no solution is associated with $\mathcal{R}_{15}$, adversely impacting the uniform distribution. Intuitively, this could be resolved if a trained ML model could help advance the nearest solution, namely, S_5, onto $\mathcal{R}_{15}$. Plausibly, the requirements on the nature of solutions' mapping, leading to the training-dataset could be better deciphered, if the desired advancement could be characterized in terms of the variations in the F values. In this light, the advancement of S_5 onto $\mathcal{R}_{15}$ could be characterized by an improvement in f_3 accompanied by deterioration in f_1 and f_2. In a different scenario, where S_5 may not exist or may not be nearest to $\mathcal{R}_{15}$, it would be fair to advance the (then) nearest solution to $\mathcal{R}_{15}$, say S_7. This advancement could be characterized as an improvement in f_1 and deterioration in f_2, accompanied by a constant f_3. The difference in the above two characterizations suggests that the *mapping requirements for diversity enhancement cannot be generalized based on the nature of variation in different objectives*. This challenge links to the fact that while traversing across the non-dominated intra-generational solutions, no transition from one solution to another could be characterized by improvement or deterioration in all the objectives, and a total of $2^M - 2$ combinations of partial improvement/deterioration in the objectives are possible.

Given the above, it becomes important that the underlying ML models are trained for the highest granularity – *capable of advancing any given solution towards an improvement in any desired objective*. If such a capability were to exist, then both aspects of diversity can be addressed, as highlighted below:

- uniform distribution: notably, the above propositions on constructing M datasets, and training M ML models towards improvement in each objective, provide a *common template* to advance any solution onto any unassociated RV, in its neighbourhood. For instance, in the context of the Figure 4.3, the unassociated RV $\mathcal{R}_{15}$ has multiple solutions in the neighborhood, including, S_5, S_7, S_8, S_{11}, and S_{12}. Hence, (i) the solution nearest to $\mathcal{R}_{15}$ can be identified, say, S_5, (ii) the objective which will undergo maximum improvement, if S_5 were to be advanced to $\mathcal{R}_{15}$ can be determined, say f_m, and (iii) the m^{th} ML model can be used to advance S_5.

- spread expansion: notably, a given spread of solutions could be expanded if the boundary solutions are pushed beyond the current envelope of solutions. In this context, consider the solution S_2 in Figure 4.3. The fact that it is a boundary solution can be established by the fact that one of the components of the underlying RV will be zero. In that, if w_{21}, w_{22}, and w_{23} denote that components of $\mathcal{R}_2$, then $w_{22} = 0$. Now, expanding the spread through S_2 calls for its advancement in the X-space, such that in the F-space it marks an improvement in f_2. This could be possible if an ML model trained for improvement in each objective (including f_2) were to exist.

While the above sets the foundation for the proposed training-dataset methodology, it is imperative to recognize the uniformity of the RVs used by the RV-EMO algorithm. The commonly used methods for generating RVs, such as Das and Dennis method [Das and Dennis, 1998], are computationally inexpensive and provide a uniform set of RVs. While the above holds true in case of multi-objective problems, in many-objective problems, the RVs generated are only on the boundary of the F-space and not in the interior. A commonly used alternative for many-objective problems is the two-layered simple-lattice design (TSLD) [Deb and Jain, 2014]. The inherent disadvantages of TSLD approach include: (a) more user-defined parameters, and (b) lower uniformity of RVs. Recently, a newer approach, referred to as incremental lattice design (ILD), has been proposed which is capable of generating a uniform set of RVs in many-objective problems covering both the boundary and interior regions [Takagi et al., 2020]. While a small

non-uniformity (as in TSLD) maybe in acceptable from an algorithmic perspective, it would be ideal to have an exact uniform distribution of the RVs.

4.1.1.2 Proposed Training-dataset

The above section has highlighted the potential utility of M ML models capable of facilitating improvement in any objective. This points to the need for M training-datasets, each of which can be used to train an ML model for improvement in a particular objective. Before detailing how such datasets could be constituted, this section sheds light on the numerical and conceptual prerequisites, including:

1. Projection of F-space onto the unit simplex: this recommendation for projecting the solutions (parent population) on to the unit simplex, has been made:

   - to ensure that the difference in the convergence levels of different solutions does not impact the endeavor for diversity improvement, in any manner.
   - towards an unbiased assessment, as to which ML model is to be used when advancing the nearest solution to an unassociated RV. Consider, that one of the unassociated RV is $\mathcal{R}_i$, whose components are w_{ik}, where $k = 1, \ldots, M$. Also consider that the nearest solution that could be advanced to $\mathcal{R}_i$ is S_j, the elements of whose objective-vector are given by f_{jk}, where $k = 1, \ldots, M$. Clearly, comparison between the components of an RV and an objective-vector, and corresponding computation of $\Delta f_k = f_{jk} - w_{ik}$ can be *viable* measure for the required improvement in the i^{th} objective *only if* the solutions have been projected on the same unit simplex, on which the RVs are sampled. The projected objective values, denoted by $\hat{F}(X_i) \equiv \{\hat{f}_1(X_i), \ldots, \hat{f}_M(X_i)\}$, can be computed from normalized objective values $\bar{F}(X_i) \equiv \{\bar{f}_1(X_i), \ldots, \bar{f}_M(X_i)\}$ (normalization as per Equation 3.1), as below.

     $$\hat{f}_m(X_i) = \frac{\bar{f}_m(X_i)}{\Sigma_{j=1}^{M} \bar{f}_j(X_i)} \; \forall \; m \in \{1, 2, \ldots, M\} \tag{4.1}$$

     Hence, if $\Delta f_k = \hat{f}_{jk} - w_{ik}$ is maximum for $k = m$, it implies that the advancement of S_j to $\mathcal{R}_i$ entails a maximum improvement in the m^{th} objective, and hence, the m^{th} ML model ought to be used.

2. Notion of neighborhood vis-à-vis adjacency of RVs: this section aims to formalize the notion of *neighborhood* of a solution, by first establishing the adjacency of any two RVs.

Since better diversity ideally calls for one solution representative per RV, it is pragmatic to *base* the mapping between the solutions that are associated with adjacent RVs. In an ideal scenario, where the RVs are uniformly distributed on the unit simplex in an exact sense, the spacing between any two RVs will be constant. However, in a practical scenario, where the RVs are uniformly distributed only in an approximate sense, the average spacing between the RVs, referred to as *neighborhood-radius* (r), can be computed as per the Equation 4.2.

$$r = \frac{\Sigma_{i=1}^{N}\{\min_{j=1}^{N}\text{dist}(\mathcal{R}_i, \mathcal{R}_j)\}}{N} \tag{4.2}$$

In this background, to facilitate the mapping of solutions between adjacent RVs, the notion of *neighborhood* of a solution is introduced. In that, given a solution s_i, its neighborhood is defined as $Nbd(S_i) \equiv \{S_j |\ 0.5r < dist(\mathcal{R}(S_i), S_j) < 1.5r\}$, where $\mathcal{R}(S_i)$ represents the RV which S_i is associated with, and $dist$ represents the euclidean distance. By this definition, as depicted in Figure 4.4, $\{S_j, S_k\} \in Nbd(S_i)$.

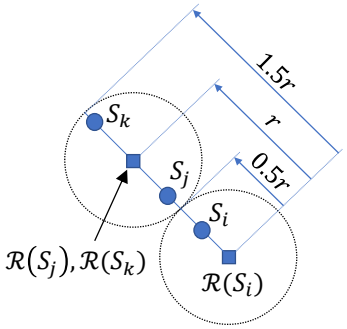


Figure 4.4. Symbolic depiction of the neighborhood of a solution vis-à-vis adjacency of RVs.

Proposed Training-dataset construction: With an aim to facilitate M ML models, *capable of advancing any given solution towards an improvement in any desired objective*:

- it is natural that M training-datasets are constructed, one-per-objective, say, $\mathcal{D}_1$–$\mathcal{D}_M$.
- for $\mathcal{D}_k$, that is to serve as the input dataset for training of k^{th} ML model (ML_k): it is proposed that for every solution $S_i,\ i = 1 \ldots N$, its underlying X-vector is mapped on to the X-vector of another solution $S_j \in Nbd(S_i)$, such that S_j offers the *maximum improvement* in $\hat{f}_k$, compared to S_i. In effect, $\mathcal{D}_k$ captures the pertinent X-space transitions, which facilitate maximum improvement in $\hat{f}_k$, across all the solutions. Such pertinent X-space transitions would be learnt by the ML_k. This process, repeated over all the objectives,

would lead to M ML models, capable of advancing any given solution, towards improvement in any desired objective.

Given the above, $\mathcal{D}_1$–$\mathcal{D}_M$ can be seen to correspond to the columns of a matrix $\mathcal{S}$, ideally sized $N \times M$, such that, its element $\mathcal{S}_{im} = \{X(S_i), X(S_j) - X(S_i)\}; i = 1 \ldots N; m = 1 \ldots M$. In that, S_i is an input solution in the F-space; $S_j \in Nbd(S_i)$ is the target solution that offers *maximum improvement* in $\hat{f}_m$; and $X(S_i)$ and $X(S_j)$ represent the X-vectors of S_i and S_j, respectively. Notably, the $Nbd(S_i)$ may be such that solutions offering improvement in each objective may not exist. Hence, the matrix $\mathcal{S}$ may not be fully populated. For visual depiction of the proposed training-dataset construction, a three-objective scenario has been presented in the Figure 4.5. In that, hypothetical solutions projected on 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.

1. $m = 1$: only one solution S_3 offers a better value in $\hat{f}_1$ 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 $\hat{f}_2$ than S_1. Among these, S_2 is selected as the *target* solution since it offers larger improvement in $\hat{f}_2$.

3. $m = 3$: there is no solution offering a better value in $\hat{f}_3$ than S_1, implying no *target* solution. Given this, S_1 does not contribute to $\mathcal{D}_3$.

4.1.1.3 Algorithmic implementation of Training-dataset Construction

In the above background, the overall procedure of constructing $\mathcal{D}_1$–$\mathcal{D}_M$ is summarized in Algorithm 4.1. In that, the first step is the computation of projected objective values $\hat{F}$, using Equation 4.1. Subsequently, each solution $S_i \in P_t$ is considered as a potential *input* solution (line 3, Algorithm 4.1), and subjected to the following steps.

1. *Identification of target solution cluster $\mathcal{C}$ (lines 4–7, Algorithm 4.1)*: 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 $S_j \in Nbd(S_i)$.

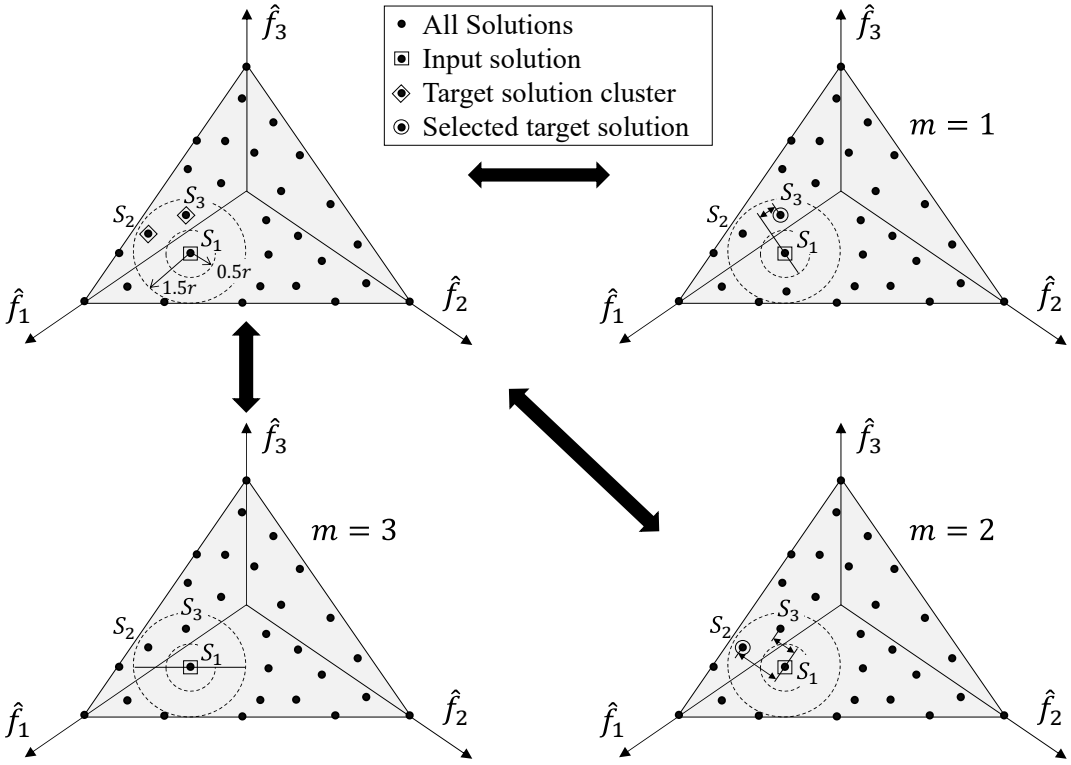


Figure 4.5. A schematic representation of the training-dataset construction based on *objective-wise* mapping, within a predefined neighborhood, in the *projected* F-space. Here, S_1 is the *input* solution, and S_2 and S_3 constitute the *target* solutions for different objectives.

2. *Identification of the target solutions from $\mathcal{C}$, towards each of $\mathcal{D}_1$–$\mathcal{D}_M$ (lines 8–11, Algorithm 4.1):* for each objective $m \in [1, M]$, the solution $S_j \in \mathcal{C}$ offering the minimum value of $\hat{f}_m(S_j)$, is identified. If the latter is better than $\hat{f}_m(S_i)$, then the underlying X vectors are included in the corresponding training-dataset $\mathcal{D}_m$ (lines 11–12, Algorithm 4.1).

4.1.2 ML training Module

The goal here is to train M ML models, over the M training-datasets, such that for each objective $m \in [1, M]$, an aggregated search direction in X-space promising improvement in $\hat{f}_m$, could be *learnt* on the basis of corresponding N promising search directions embedded in $\mathcal{D}_m$. Towards it, k-Nearest Neighbours (kNN) regression[7] has been used as the ML method. When a test instance is provided for prediction, the kNN model identifies its k nearest inputs in the original training-

[7]The kNN Regressor used in this study has been taken from the Scikit-learn implementation (for python). Link: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

Algorithm 4.1: `Dataset_Construction`$(P_t, \mathcal{R}, r)$

Input: Parent population P_t, RVs $\mathcal{R}$, neighbourhood radius r

Output: M training-datasets $\mathcal{D}_1$–$\mathcal{D}_M$

Compute $\hat{F}$ values using Equation 4.1
$\{\mathcal{D}_1, \mathcal{D}_1, \ldots, \mathcal{D}_M\} \leftarrow \{\emptyset, \emptyset, \ldots, \emptyset\}$
for $S_i \in P_t$ **do**
    $\mathcal{C} \leftarrow \emptyset$
    for $S_j \in P_t \; \forall \; j \neq i$ **do**
        if $S_j \in Nbd(S_i)$ **then**
            $\mathcal{C} \leftarrow \mathcal{C} \cup S_j$
    for $m = 1$ *to* M **do**
        $S_j \leftarrow \text{argmin}_{S_j \in \mathcal{C}} \hat{f}_m(S_j)$
        if $\hat{f}_m(S_j) < \hat{f}_m(S_i)$ **then**
            $\mathcal{D}_m \leftarrow \mathcal{D}_m \cup [X(S_i), (X(S_j) - X(S_i))]$

dataset, and the average of their respective targets is returned as the prediction [Cover, 1968]. In the context of $\mathcal{D}_m$, given the X-vector of any solution as a test input: (a) the kNN model (trained on $\mathcal{D}_m$) identifies its k nearest *input* solutions in $\mathcal{D}_m$; (b) averages their respective difference vectors; and (c) returns the average difference vector, which could be treated as a potential search direction in X-space. Besides the above, the following aspects are noteworthy.

- During the training, the *input* solutions in $\mathcal{D}_m$ are strategically stored using a k-d tree method [Bentley, 1975], so that while making the prediction for a test input, the corresponding k nearest neighbours can be identified efficiently.

- It is known that a very low value of k leads to overfitting of the kNN, whereas a very high value of k leads to underfitting. In that, an appropriate choice for k depends on the dataset and the application it represents. In the context of IP3 operator, $k = n_{\text{var}}$ has been used, based on the rationale that: (i) the training-dataset, say $\mathcal{D}_m$, involves $n_{\text{var}} \times 1$ dimensional vectors, say, X_I and X_O for the input and output, respectively, and (ii) given an $n_{\text{var}} \times 1$ dimensional test-vector (say, X_T), it is appropriate to account for as many nearest-neighbors (k) of X_T among X_I, as their are variables, so that even if each neighboring X_I accounts for variation in only one of distinct elements in an $n_{\text{var}} \times 1$ dimensional X_T, the neighboring X_Is collectively account for sufficient variation on the input side, so that the

average of the corresponding X_Os does not amount to overfitting. Moreover, a sensitivity analysis for k has been presented later in Section 4.5.3.

Similar to the IP2 operator, the ML training module is executed as a two-step process: (a) normalization of the training-dataset using the proposed *dynamic normalization* method, as a pre-training step; and (b) the ML training itself, as presented in Algorithm 4.2. Since the kNN model relies on identification of the k nearest neighbours, it is important that the dataset is normalized in X-space before training. Here, the above process is repeated M times for training M ML models. The details of the proposed *dynamic normalization* method can be found in Section 3.1.2 (Chapter 3).

Algorithm 4.2: `ML_Training` $(\mathcal{D}, [x^l, x^u])$

Input: Training datasets $\mathcal{D}$, lower & upper bounds of variables specified in the problem, x^l and x^u

Output: M trained ML models ML_1–ML_M

$\{x^{l,t}, x^{u,t}\} \leftarrow$ Minimum and Maximum of each variable in $\mathcal{D}$

$x^{\min}, x^{\max} \leftarrow \emptyset, \emptyset$

for $k = 1$ *to* n_{var} **do**

  $x_k^{\min} = 0.5 \times (x_k^{l,t} + x_k^l)$

  $x_k^{\max} = 0.5 \times (x_k^{u,t} + x_k^u)$

Normalize $\mathcal{D}$ using $x^{\min}$ and $x^{\max}$ as bounds

for $m = 1$ *to* M **do**

  Train ML_m using $\mathcal{D}_m$

4.1.3 Offspring Creation Module

In any generation where IP3 is invoked, this module entails the advancement of a proportion $\mathcal{P}^{\text{IP3}}$ of the parent solutions P_t, $\lfloor \mathcal{P}^{\text{IP3}} N \rfloor$ in number, leading to offspring which mark an improvement in both uniformity and spread. In that, $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ offspring, Q_t^B, are created by subjecting suitable P_t members to *boundary progression* for better spread; and another $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ offspring, Q_t^G, are created by subjecting suitable P_t members to *gap progression* for better uniformity. The respective procedures are detailed below.

4.1.3.1 Boundary Progression

The procedure for creating $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ offspring, Q_t^B, by subjecting suitable P_t members to *boundary progression* for better spread, is summarized in Algorithm 4.3. As per the rationale set up in Section 4.1.1.1, it entails identification of the boundary solutions in the parent population P_t, and their advancement using an appropriate ML model to create new offspring solutions.

Algorithm 4.3: `Boundary_Progression` (P_t, $\mathcal{R}$, ML, [$x^{\min}, x^{\max}$], [x^l, x^u], $\mathcal{P}^{\text{IP3}}$)

Input: Current population P_t, RVs $\mathcal{R}$, ML models ML_1–ML_m, bounds from Algorithm 4.2 [$x^{\min}, x^{\max}$], variable bounds in problem definition [x^l, x^u], offspring proportion to be created using IP3 $\mathcal{P}^{\text{IP3}}$

Output: New solutions created Q_t^B

1 $\mathcal{R}^B \leftarrow$ All associated RVs in $\mathcal{R}$ with at least one '0' component

2 $Q_t^B \leftarrow \emptyset$ `% sized` $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor \times n_{\text{var}}$

3 **for** $i = 1$ *to* $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ **do**

4 $\quad \vec{B} \leftarrow$ A randomly selected RV from $\mathcal{R}^B$

5 $\quad S_{\text{start}} \leftarrow$ Nearest solution associated with $\vec{B}$

6 $\quad m \leftarrow$ A randomly selected objective such that $\vec{B}_m = 0$

7 $\quad \bar{X}(S_{\text{start}}) \leftarrow$ Normalized $X(S_{\text{start}})$ using $x^{\min}$ and $x^{\max}$

8 $\quad \bar{d_X} \leftarrow ML_m(\bar{X}(S_{\text{start}}))$

9 $\quad d_X \leftarrow$ Denormalized $\bar{d_X}$ using $x^{\min}$ and $x^{\max}$

10 $\quad X(S_{\text{new}}) \leftarrow X(S_{\text{start}}) + \lambda_B \times \hat{d}_X$

11 $\quad$ Boundary repair on $X(S_{\text{new}})$

12 $\quad Q_t^B \leftarrow Q_t^B \cup S_{\text{new}}$

The boundary solutions in P_t can be characterized as those associated with boundary RVs, implying RVs having at least one '0' component ($w_i = 0$ for some $i \in [1, M]$). To facilitate access to such boundary solutions, in algorithmic implementation, let the boundary RVs which have at least one member of P_t, associated with themselves, be denoted by $\mathcal{R}^B$ (line 1, Algorithm 4.3). Subsequently, the creation of each offspring in Q_t^B, involves the following steps.

1. *Identification of the solution to be advanced (lines 4–5, Algorithm 4.3):* an RV $\vec{B}$ is randomly selected from $\mathcal{R}^B$. If there is only one solution associated with $\vec{B}$, it is identified as the desired solution, denoted as S_{start}. Otherwise, if there are more than one solutions associated with $\vec{B}$, the solution nearest to $\vec{B}$ by perpendicular distance is identified as S_{start}.

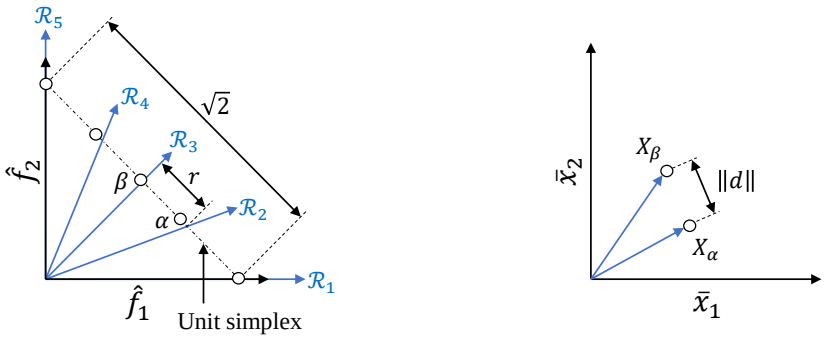
2. *Selection of an appropriate ML model (line 6, Algorithm 4.3):* the solution S_{start} identified above for advancement, can be characterized by the fact that it offers one of the best values in at least one particular objective. For instance, if the j^{th} component of S_{start}'s underlying $\vec{B}$ is 0, it implies that S_{start} offers one of the best values in the j^{th} objective, and that further improvement in the j^{th} objective would imply boundary expansion (beyond the current projected F-space.). In case more than one components of $\vec{B}$ are 0, then one of them, say indexed as m, is randomly picked, and the corresponding ML model, namely, ML_m becomes the desired ML model.

3. *Obtaining the search direction (lines 7–9, Algorithm 4.3):* as highlighted earlier in Section 4.1.2, given the X-vector of a solution, the trained model ML_m is capable of providing a potential search direction in X-space. Hence, a search direction corresponding to S_{start}, denoted by d_X, could be obtained by applying ML_m on $X(S_{\text{start}})$. However, since ML_m was trained on the normalized dataset: (i) $X(S_{\text{start}})$ needs to be normalized using the bounds computed in Algorithm 4.2, before applying ML_m; and (ii) the normalized search direction obtained by applying ML_m, denoted by $\bar{d_X}$, needs to be denormalized using the same bounds, leading to d_X.

4. *Advancement and repair (lines 10–11, Algorithm 4.3):* the advancement 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. Towards the determination of λ_B, the desired scaling in F-space for boundary progression needs to be imposed on the relationship of scales in F-space and X-space that is inherent in the ML training-dataset, as discussed below:

   (a) the scaling requirements in the F-space, say in multiple of r, where r represents the average distance between two adjacent RVs. In a hypothetical situation, if the spread in F-space is desired to double-up compared to the current spread, it could be achieved with reference to the unit simplex, by accounting for the product of: (i) the average distance between any two adjacent RVs in the unit simplex, given by r, and (ii) the number of adjacent-RV transitions required to span the boundary of the unit simplex, given by $\lfloor\sqrt{2}/r\rceil$. Reference may be made to Figure 4.6a, that symbolically depicts the projected F-space with 5 equi-spaced RVs ($\mathcal{R}_1$–$\mathcal{R}_5$) and associated solutions. In order to span the boundary, i.e., to move from one extreme

RV ($\mathcal{R}_1$) to another extreme RV ($\mathcal{R}_5$), the number of adjacent-RV transitions required are given by $\sqrt{2}/r = 4$. Notably, there may be scenarios where the RVs are not exactly equi-spaced, leading to a non-integral value of $\sqrt{2}/r$. Hence, $\lfloor\sqrt{2}/r\rceil$ is used. In effect, to double-up the current spread, a scaling factor of $\lfloor\sqrt{2}/r\rceil \times r$ is required in the projected F-space.



(a) Projected F-space

(b) Normalized X-space (only 2 of n_{var})

Figure 4.6. A symbolic depiction of distance between two solutions associated with adjacent RVs.

(b) the inherent relationship of scales in F-space and X-space in the ML training-dataset. In that, consider two solutions α and β as associated with adjacent RVs (as depicted in Figure 4.6a). While in the F-space, they can be reasonably assumed to be separated by a distance r; their distance in the X-space can be given by $\|d\|$, where $d \equiv X_\beta - X_\alpha$ (as depicted in Figure 4.6b). Such $\|d\|$ can be computed for the solutions associated with all the pairwise adjacent RVs, and an average value of $\|d_A\|$ can be arrived at. Now, considering that the required scaling factor to double-up the spread in the F-space is $\lfloor\sqrt{2}/r\rceil \times r$, the corresponding step length in the X-space can be given by $\lambda_B = \lfloor\sqrt{2}/r\rceil \times \|d_A\|$, where $\|d_A\|$ is the average of $\|d\|$ over the solutions associated with all the pairwise adjacent RVs.

While the founding principles of the scaling factor in the F-space, and the step length in the X-space are presented above, the actual scaling factor and step length need to be adapted, so that they can cater to the varying scope for spread expansion, along the RV-EMO generations. In that:

- during the early generations of an RV-EMO algorithm, the current spread may be only

a minor fraction of the targeted PF spread, and a large scaling factor may be desired to cater to the large scope for spread expansion,

- during the later generations, where the actual spread may be comparable to the targeted PF spread, and a small scaling factor may be desired to cater to the limited scope for spread expansion.

To account for such varying requirements, it is important to: (i) impose an upper and lower bound on the scaling factor, and (ii) infuse randomness while assigning an intermediate value to it (since the scope for spread expansion cannot be determined a priori). Considering such requirements, this thesis proposes that the upper bound for the scaling factor be based on doubling the current spread, while the lower bound and intermediate values can be obtained through a randomized scalar. Considering the above modifications to the scaling factor (in F-space), the corresponding step length (in X-space) can be given by $\lambda_B = rand(0,1) \times \lfloor \sqrt{2}/r \rceil \times \|d_A\|$. Notably, the obtained $X(S_{\text{new}})$ may have some variables outside their respective permissible bounds. These variables are repaired using the method proposed in [Padhye et al., 2013], which is the same as used in the IP2 operator.

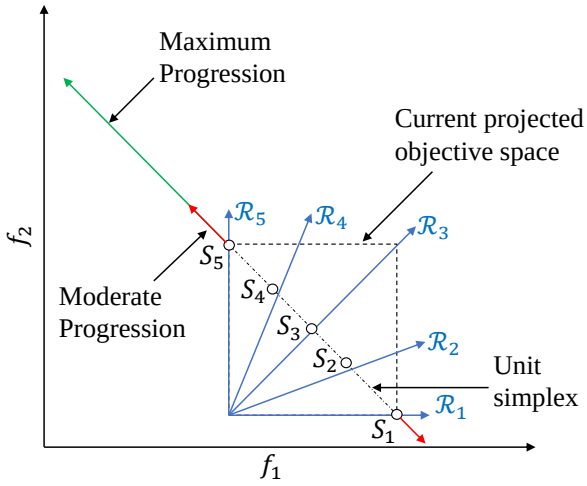


Figure 4.7. A symbolic depiction of the boundary progression, during an RV-EMO-IP3 run.

Finally, the key features of the boundary progression procedure are symbolically depicted in Figure 4.7. In that, the projected F-space at an intermediate generation of an RV-EMO-IP3 run is shown. This space is discretized through 5 equi-spaced RVs ($\mathcal{R}_1$–$\mathcal{R}_5$), whose associated parent solutions are marked. Since $\mathcal{R}_1$ and $\mathcal{R}_5$ are the only boundary RVs,

only S_1 and S_5 can be used for boundary progression. Notably, the maximum progression corresponding to $rand(0,1) = 1$, implying a scaling factor of $\lfloor\sqrt{2}/r\rceil \times r$, is depicted by the green arrow. Further, the moderate progression corresponding to an intermediate value of $rand(0,1)$, implying a scaling factor of $< \lfloor\sqrt{2}/r\rceil \times r$, is depicted by the red arrows.

While the above steps are repeated $\lfloor\mathcal{P}^{\text{IP3}}N/2\rfloor$ times for creation of $\lfloor\mathcal{P}^{\text{IP3}}N/2\rfloor$ offspring, the random selection of the boundary solutions and their respective progression with a random step length ensures that, across the generations, the spread expansion is fairly emphasized on all boundaries.

4.1.3.2 Gap Progression

The procedure for creating $\lfloor\mathcal{P}^{\text{IP3}}N/2\rfloor$ offspring, Q_t^G, by subjecting suitable P_t members to *gap progression* for better uniformity, is summarized in Algorithm 4.4. As per the rationale set up in Section 4.1.1.1, it entails identification of the unassociated RVs, and the advancement of their respective nearest solutions in P_t to create new offspring solutions, using an appropriate ML model. The unassociated RVs can simply be characterized as those with no solution in P_t associated with them. To facilitate their access, in algorithmic implementation, they are denoted as $\mathcal{R}^G$ (line 1, Algorithm 4.4). Subsequently, the creation of each offspring solution in Q_t^G, involves the following steps.

1. *Identification of the solution to be advanced (lines 4–5, Algorithm 4.4):* an RV $\vec{G}$ is randomly selected from $\mathcal{R}^G$. The solution nearest to $\vec{G}$ (by perpendicular distance) is identified as the solution to be advanced, and denoted as S_{start}. Notably, S_{start} will be associated with some other RV than $\vec{G}$, since $\vec{G}$ is an unassociated RV.

2. *Selection of an appropriate ML model (lines 6–7, Algorithm 4.4):* as per the rationale set earlier in Section 4.1.1.1, S_{start} can be advanced to $\vec{G}$ by improving the particular objective requiring maximum improvement, say Δf_i. Alternatively, S_{start} may require maximum deterioration in a particular objective to advance to $\vec{G}$, say Δf_j. Plausibly: (i) $\Delta f_i \geq \Delta f_j$, where the model ML_i can be applied to improve S_{start} in the i^{th} objective; and (ii) $\Delta f_i < \Delta f_j$, where the model ML_j can be applied, with a minor adaptation, to deteriorate S_{start} in the j^{th} objective. As per the case applicable, $m = i$ or j, and the corresponding model is referred to as ML_m.

Algorithm 4.4: `Gap_Progression` (P_t, $\mathcal{R}$, ML, $[x^{\min}, x^{\max}]$, $[x^l, x^u]$), $\mathcal{P}^{\text{IP3}}$

Input: Current population P_t, RVs $\mathcal{R}$, ML models ML_1–ML_M, bounds from Algorithm 4.2 $[x^{\min}, x^{\max}]$, variable bounds in problem definition $[x^l, x^u]$, offspring proportion to be created using IP3 $\mathcal{P}^{\text{IP3}}$

Output: New solutions created Q_t^G

1 $\mathcal{R}^G \leftarrow$ All empty RVs in $\mathcal{R}$

2 $Q_t^G \leftarrow \emptyset$ `% sized` $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor \times n_{\text{var}}$

3 **for** $i = 1$ *to* $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ **do**

4 $\quad \vec{G} \leftarrow$ A randomly selected RV from $\mathcal{R}^G$

5 $\quad S_{\text{start}} \leftarrow$ Nearest solution to $\vec{G}$ `% associated with some other vector`

6 $\quad \vec{\delta} \leftarrow \vec{G} - \hat{F}(S_{\text{start}})$

7 $\quad m \leftarrow \text{argmax}_{m \in [1,M]}(\text{abs}(\vec{\delta}))$

8 $\quad \bar{X}(S_{\text{start}}) \leftarrow$ Normalized $X(S_{\text{start}})$ using $x^{\min}$ and $x^{\max}$

9 $\quad$ **if** $\vec{\delta}_m < 0$ **then**

10 $\quad\quad \bar{d_X} \leftarrow ML_m(\bar{X}(S_{\text{start}}))$

11 $\quad$ **else**

12 $\quad\quad \bar{d_X} \leftarrow -1 \times ML_m(\bar{X}(S_{\text{start}}))$

13 $\quad d_X \leftarrow$ Denormalized $\bar{d_X}$ using $x^{\min}$ and $x^{\max}$

14 $\quad X(S_{\text{new}}) \leftarrow X(S_{\text{start}}) + \lambda_G \times \hat{d}_X$

15 $\quad$ Boundary repair on $X(S_{\text{new}})$

16 $\quad Q_t^G \leftarrow Q_t^G \cup S_{\text{new}}$

3. *Obtaining the search direction (lines 8–13, Algorithm 4.4):* as highlighted earlier in Section 4.1.2, ML_m applied on to $X(S_{\text{start}})$ provides a potential search direction in X-space, say, d_X. Since ML_m was trained on the normalized dataset: (i) $X(S_{\text{start}})$ needs to be normalized using the bounds computed in Algorithm 4.2, before applying ML_m; and (ii) the normalized search direction obtained by applying ML_m, denoted by $\bar{d_X}$, needs to be denormalized using the same bounds, leading to d_X. Notably, by default, ML_m aims to provide improvement in the m^{th} objective. Hence, if deterioration is required in the m^{th} objective, ML_m is applied with a minor adaptation. In that, the obtained search direction ($\bar{d_X}$) is reversed by multiplying each of its components with (-1), as depicted in line 12 (Algorithm 4.4).

4. *Advancement and repair (lines 14–15, Algorithm 4.4):* the advancement 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 d_X (computed above), and λ_G is the step length. In the context of *boundary progression*,

the founding principles enabling computation of the scaling factor and the step length were laid down. These principles apply, as such, in the context of *gap progression* too, just that the specific formulations differ owing to the contextual difference, as evident below.

(a) the scaling requirements in the F-space, say in multiple of r, where r represents the average distance between two adjacent RVs. Intuitively, given a solution S_{start} to be advanced to an unassociated RV $\vec{G}$, the appropriate scaling could be achieved with reference to the unit simplex, by accounting for the product of: (i) the average distance between any two adjacent RVs in the unit simplex, given by r; and (ii) the number of adjacent-RV transitions required to arrive at $\vec{G}$ from $\hat{F}(S_{\text{start}})$, given by $\lfloor \|\vec{\delta}\|/r \rceil$, where $\vec{\delta} \equiv \vec{G} - \hat{F}(S_{\text{start}})$. Reference may be made to Figure 4.8, that symbolically

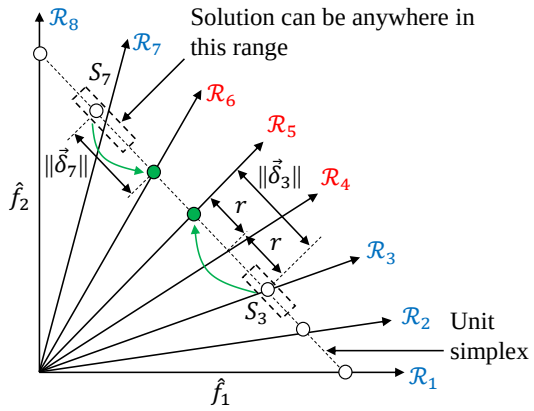


Figure 4.8. A symbolic depiction of the gap progression, during an RV-EMO-IP3 run.

depicts the projected F-space with 8 equi-spaced RVs ($\mathcal{R}_1$–$\mathcal{R}_8$), where $\mathcal{R}_4$–$\mathcal{R}_6$ are unassociated. To depict the advancement of solutions onto adjacent and non-adjacent RVs, two instances are highlighted by green arrows, including:

- advancement of solution S_3 to non-adjacent $\mathcal{R}_5$: $\|\vec{\delta}_3\|$ marked in the figure implies $\|\vec{\delta}_3\|/r = 2$. Notably, S_3 may lie within a distance of $0.5r$ on either side of $\mathcal{R}_3$ (depicted by the engulfing rectangle). Hence, in general, a non-integral value of $\|\vec{\delta}_3\|/r$ may result, necessitating the use of $\lfloor \|\vec{\delta}\|/r \rceil$.
- advancement of solution S_7 to adjacent $\mathcal{R}_6$: $\|\vec{\delta}_7\|$ marked in the figure implies $\lfloor \|\vec{\delta}_7\|/r \rceil = 1$. Notably, S_7 may lie within a distance of $0.5r$ on either side of $\mathcal{R}_7$ (depicted by the engulfing rectangle). Yet, regardless of S_7's location, $\lfloor \|\vec{\delta}_7\|/r \rceil = 1$ would hold.

In effect, it is fair to infer that a scaling factor of $\lfloor \|\vec{\delta}\|/r \rceil \times r$ is required in the projected F-space.

(b) the inherent relationship of scales in F-space and X-space in the ML training-dataset, as discussed earlier in reference to the boundary progression. In that, computation of average spacing in X-space, given by $\|d_A\|$, has also been discussed. Hence, considering the scaling required in F-space for *gap progression*, the corresponding step length in the X-space can be given by $\lfloor \|\vec{\delta}\|/r \rceil \times \|d_A\|$.

While the founding principles of the scaling factor (in F-space) and the step length (in X-space) are presented above, the actual scaling factor and corresponding step length needs to be adapted, so it can cater to the uncertainty around S_{start}'s location vis-à-vis its underlying RV. Since this uncertainty is limited to a range of $\pm 0.5r$ about the underlying RV, this thesis postulates that the desired step length be given by $\lambda_G = rand(\lfloor \|\vec{\delta}\|/r \rceil - 0.5, \lfloor \|\vec{\delta}\|/r \rceil + 0.5) \times \|d_A\|$. Notably, the obtained $X(S_{\text{new}})$ may have some variables outside their respective permissible bounds, that are repaired using the same method [Padhye et al., 2013], as used in the boundary progression.

While the above steps are repeated $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ times for creating $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ offspring solutions, the random selection of the unassociated RVs ensures that each identified gap in the current population is fairly emphasized.

4.2 Integration of IP3 operator into NSGA-III

This section outlines the integration of the IP3 operator with NSGA-III, leading to NSGA-III-IP3. This integration, summarized in Algorithm 4.5 is generic in nature, and can be extended to any other RV-EMO algorithm.

Notably, Algorithm 4.5 represents any intermediate generation t of NSGA-III-IP3, and involves a new parameter, namely, $t^{\text{IP3}}_{\text{freq}}$, that specifies the number of generations between two successive progressions. First, it is checked if the population has mildly stabilized for the first time (line 1, Algorithm 4.5). For its detection, a stabilization tracking algorithm [Saxena and Kapoor, 2019] has been used[8]. If detected, the $startIP3$ flag is marked as *True* (lines 2–3, Algorithm 4.5). Until this flag is marked, the IP3 operator is not invoked at all. It is crucial to let

[8]This stabilization tracking algorithm can be used for: (a) triggering the IP3 operator with a mild setting, and (b) terminate the NSGA-III-IP3 run with a strict setting.

Algorithm 4.5: 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 $t^{\text{IP3}}_{\text{freq}}$, generation of last progression $t^{\text{IP3}}_{\text{pg}}$, neighbourhood radius r, number of survived offspring in $(t-1)^{\text{th}}$ generation $N^{\text{survived}}_{t-1}$

Output: P_{t+1}, $t^{\text{IP3}}_{\text{pg}}$, $t^{\text{IP3}}_{\text{freq}}$, N^{survived}_{t}

1 $check \leftarrow$ `Check_Mild_Stabilization()`

2 **if** $check$ **then**

3 $\quad startIP3 = True$

4 **if** $startIP3$ *and* $t - t^{\text{IP3}}_{\text{pg}} = t^{\text{IP3}}_{\text{freq}}$ **then**

5 $\quad \mathcal{D} \leftarrow$ `Dataset_Construction`$(P_t, \mathcal{R}, r)$

6 $\quad ML, [x^{\min}, x^{\max}] \leftarrow$ `Training`$(\mathcal{D}, [x^l, x^u])$

7 $\quad Q^B_t \leftarrow$ `Boundary_Progression` $(P_t, \mathcal{R}, ML, [x^{\min}, x^{\max}], [x^l, x^u], \mathcal{P}^{\text{IP3}})$ % $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$

8 $\quad Q^G_t \leftarrow$ `Gap_Progression` $(P_t, \mathcal{R}, ML, [x^{\min}, x^{\max}], [x^l, x^u], \mathcal{P}^{\text{IP3}})$ % $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$

9 $\quad Q^{\text{V}}_t \leftarrow$ `Variation`(P_t) % $\lceil (1 - \mathcal{P}^{\text{IP3}}) N \rceil$

10 $\quad Q_t \leftarrow Q^B_t \cup Q^G_t \cup Q^{\text{V}}_t$ % N

11 $\quad t^{\text{IP3}}_{\text{pg}} \leftarrow t$

12 **else**

13 $\quad Q_t \leftarrow$ `Variation`(P_t) % N

14 Evaluate Q_t

15 $P_{t+1} \leftarrow$ `Survival_selection`$(P_t \cup Q_t)$

16 $N^{\text{survived}}_t \leftarrow sizeof(Q_t \cap P_{t+1})$

17 **if** $t = t^{\text{IP3}}_{\text{pg}}$ **then**

18 $\quad$ **if** $N^{\text{survived}}_t > N^{\text{survived}}_{t-1}$ **then**

19 $\quad\quad t^{\text{IP3}}_{\text{freq}} \leftarrow t^{\text{IP3}}_{\text{freq}} - 1$

20 $\quad$ **if** $N^{\text{survived}}_t < N^{\text{survived}}_{t-1}$ **then**

21 $\quad\quad t^{\text{IP3}}_{\text{freq}} \leftarrow t^{\text{IP3}}_{\text{freq}} + 1$

the population stabilize a little before applying the IP3 operator since focusing excessively on diversity enhancement before the population has mildly converged may lead to a delayed convergence, at the cost of additional computational expense. In other words, NSGA-III-IP3 is identical to base NSGA-III till the generation in which mild stabilization is detected. Notably, the underlying stabilization tracking algorithm requires an additional parameter set (ψ_{mild}) to detect the mild stabilization, as discussed later in Section 4.4.3.2.

In the subsequent generations, if $t^{\text{IP3}}_{\text{freq}}$ generations have passed after the last invoked progression in generation $t^{\text{IP3}}_{\text{pg}}$, the IP3 operator is invoked (line 4, Algorithm 4.5). If invoked: the M

training datasets $\mathcal{D}_1$–$\mathcal{D}_M$ are constructed using Algorithm 4.1; M ML models are trained on $\mathcal{D}_1$–$\mathcal{D}_M$ using Algorithm 4.2; $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ offspring solutions are created by each of Algorithms 4.3 and 4.4, and remaining $\lceil (1 - \mathcal{P}^{\text{IP3}})N \rceil$ offspring solutions are created using the variation operators (crossover and mutation in this case); and the offspring solutions created by all three methods are combined into N sized Q_t (lines 4–11, Algorithm 4.5). Otherwise, if the IP3 operator is not invoked, N offspring are created directly using the variation operators, as in the case of base NSGA-III (lines 12–13, Algorithm 4.5). Subsequently, all N offspring solutions are evaluated and NSGA-III's survival selection procedure is executed (lines 14–15, Algorithm 4.5). Towards the end, the count of offspring that survived to the next generation (N_t^{survived}) is estimated (line 16, Algorithm 4.5). If this count has improved compared to the previous generation, implying a good performance of the IP3 operator, then $t_{\text{freq}}^{\text{IP3}}$ is reduced by 1, implying a more frequent progression. However, if this count has degraded, implying a poorer performance of the IP3 operator, $t_{\text{freq}}^{\text{IP3}}$ is increased by 1 implying a less frequent progression. This adaptation is only executed in generations where IP3 is invoked (line 17, Algorithm 4.5). Notably, in absence of gaps in the population, i.e., if no RVs are unassociated, then the $\lfloor \mathcal{P}^{\text{IP3}} N/2 \rfloor$ offspring solutions created using gap progression, are created using boundary progression, ensuring that overall $\lfloor \mathcal{P}^{\text{IP3}} N \rfloor$ offspring solutions are created using the IP3 operator.

4.3 Computational Complexity of IP3 operator

As detailed in the section above, the proposed IP3 operator is constituted by three modules. In the following subsections, time and space complexity of each constituent module has been discussed, followed by its overall summary.

4.3.1 Training-datasets Construction Module

The process of constructing M training-datsets $\mathcal{D}_1$–$\mathcal{D}_M$ has been summarized in Algorithm 4.1. In that, for each solution of 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-EMO algorithms (as also used in this thesis) satisfy $N \geq M^2$. Hence, the resulting time complexity of this module

can be approximated as $\mathcal{O}(N^2)$.

Further, only the training-datasets $\mathcal{D}_1$–$\mathcal{D}_M$ are additionally created, over 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 $\mathcal{O}(MN)$.

4.3.2 ML Training Module

The training-datasets constructed above are 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 the kNN is $\mathcal{O}(N_{\text{dim}}N_{\text{sam}})$. In this case, $N_{\text{dim}} = n_{\text{var}}$, $\max(N_{\text{sam}}) = N$, and M trainings are executed. Upon substituting their values, the resulting time and space complexities of the ML training module are $\mathcal{O}(MNn_{\text{var}}\log(N))$ and $\mathcal{O}(MNn_{\text{var}})$, respectively.

4.3.3 Offspring Creation Module

Evidently, this module constitutes two submodules, each of which follow 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 to be advanced; (b) selection of the ML model; (c) identification of the search direction; and (d) advancement and repair. The worst case computational complexity of step-(a) is $\mathcal{O}(MN)$, owing to the identification of the nearest 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 neighbours. Since $k = n_{\text{var}}$ has been used, the resulting time complexity of making a prediction is $\mathcal{O}(Nn_{\text{var}})$. Finally, step-(d) involves the advancement of a given solution with $\mathcal{O}(M)$ complexity. Amongst the four steps, the worst time complexity can be given as $\max\{\mathcal{O}(MN), \mathcal{O}(Nn_{\text{var}})\}$. Generally, $n_{\text{var}} > M$ is majority of the MOPs. Hence, the worst case time complexity can be approximated as $\mathcal{O}(Nn_{\text{var}})$. Since, these steps are repeated for $\lfloor \mathcal{P}^{\text{IP3}}N/2 \rfloor$ solutions, the resulting time complexity becomes $\mathcal{O}(N^2n_{\text{var}})$. Moreover, since the offspring solutions created through advancement at included in the N offspring solutions that are created by any RV-EMO algorithm, there is no related space complexity of this module.

The worst case time complexity and space complexity of each constituent module of the IP3 operator is summarized in Table 4.1. Evidently, training of M kNN models have the highest time complexity, and the M trained kNN models have the highest space complexity.

Table 4.1. 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	$\mathcal{O}(MNn_{\text{var}}\log(N))$	$\mathcal{O}(MNn_{\text{var}})$
Offspring creation	$\mathcal{O}(N^2n_{\text{var}})$	–

4.4 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-EMO algorithm(s) and the IP3 operator.

4.4.1 Test-suite

To demonstrate the search efficacy infused by the IP3 operator into an RV-EMO algorithm, several two- and three-objective problems with varying degrees of difficulty have been used. These include: CIBN [Lin et al., 2017], DASCMOP [Fan et al., 2020] and MW [Ma and Wang, 2019] problems with the following specifications.

- CIBN: CIBN1–3 are two-objective and CIBN4–5 are three-objective, with $n_{\text{var}} = 10$.
- DASCMOP: DASCMOP1–6 are two-objective problems and DASCMOP7–9 are three-objective problems, with $n_{\text{var}} = 30$. Since different difficulty settings are available for these problems [Fan et al., 2020], setting 5 has been used here that corresponds to diversity-hardness in these problems.
- MW: majority of these problems are two-objective, except for MW4, MW8 and MW14 that are three-objective. Here, $n_{\text{var}} = 15$ for all 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 EMO algorithms, such NSGA-III, that rely

on the constraint dominance principle for handling constraints, exhibit severe under-performance on some of these problems [Ma et al., 2021].

4.4.2 Performance Indicators and Statistical Analysis

The choice of performance indicators is exactly the same as adopted earlier in Chapter 3. The key details are re-iterated below.

- Hypervolume is used the primary indicator with 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-EMO. Further, for the problems where the scales of different objectives are different, the solutions are normalized in the F-space using the theoretical PF extremes.

- Population mean of the $g(X)$ function is used as the secondary indicator, to provide insights into the convergence levels in the X-space. Although the IP3 operator focuses explicitly on diversity-enhancement, it would be intriguing to assess if 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, Wilcoxon ranksum test [Wilcoxon, 1945] is performed on the indicator values reported over multiple/independently seeded runs. In that, the threshold value of 0.05 (95% confidence interval) is used.

- when comparing *more than two* algorithms, at a time, Kruskal-Wallis test [Kruskal and Wallis, 1952] with 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 reference. Furthermore, the threshold p-value is adjusted using the standard Bonferroni correction [Abdi, 2007], to retain the same overall confidence.

4.4.3 Parameter Settings

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

4.4.3.1 RV-EMO Settings

These settings have been kept exactly the same as used for the IP2 operator in Chapter 3. The key details are re-iterated 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 $p_c = 1/n_{\text{var}}$ and $\eta_m = 20$.
- Each of NSGA-III and NSGA-III-IP3 has been run for 31 times, with random seeds.
- For NSGA-III-IP3, t_{term} has been determined on-the-fly through the stabilization tracking algorithm, using $\psi_{\text{term}} \equiv \{3, 50\}$. For NSGA-III, the mean t_{term} determined for NSGA-III-IP3 over 31 runs has been used as the t_{term}.

4.4.3.2 IP3 Operator Settings

The proposed IP3 operator involves five parameters: $\mathcal{P}^{\text{IP3}}$, r, $t^{\text{IP3}}_{\text{freq}}$, and ψ_{mild}. In that: $\mathcal{P}^{\text{IP3}}$ refers to the proportion of the total offspring (N) created using the IP3 operator; the neighbourhood-radius r governs the identification of the target solutions cluster during the mapping; $t^{\text{IP3}}_{\text{freq}}$ 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-EMO-IP3 run.

$\mathcal{P}^{\text{IP3}} = 50\%$ has been used, as reasoned earlier in Section 1.2 (Chapter 1). r is simply derived from the given RV set $\mathcal{R}$ (Equation 4.2), and $t^{\text{IP3}}_{\text{freq}}$ has been adapted on-the-fly based on the survival of the offspring, as can be observed in Algorithm 4.5. The initial value of $t^{\text{IP3}}_{\text{freq}}$ is set as 1. While the first three (out of four) parameters could be rationally derived or adapted, a direct impact of ψ_{mild} on the performance of the IP3 operator is not that straight.

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 [Saxena and Kapoor, 2019]. According to that, the mild stabilization corresponds to $\psi_{\text{mild}} = \{2, 20\}$.

4.5 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. Towards the end, an assessment of the IP3 operator's performance sensitivity towards the variation in k (used for kNN, the underlying ML method), has been presented.

4.5.1 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 background, Table 4.2 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of t_{term} generations. In that, t_{term} has been 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 performs 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) towards 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, towards diversity-enhancement.

4.5.2 Insights into sample 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

Table 4.2. Hypervolume and $g(X)$ based comparison of NSGA-III and NSGA-III-IP3 on benchmark CIBN, DASCMOP and MW problems, at t_{term} 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_{term} generations. The best performing algorithm and its statistical equivalent are marked in bold. Note: UPS denotes the Pareto-set of the unconstrained MOP.

	Problem	t_{term}	Median hypervolume			Median $g(X)$			
			NSGA-III	NSGA-III-IP3	p-value	$g(X)\|_{X\in UPS}$	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.

achieves a reasonable PF approximation; and (c) DASCMOP9 – a three objective problem, as presented below.

4.5.2.1 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 4.9 shows the final set of solutions obtained in the respective median runs of NSGA-III and NSGA-III-IP3 (out of 31 randomly seeded runs each). The termination generation t_{term} has been set as 1404 for NSGA-III, as 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, as per the premise for interpretation of the results, laid earlier (Chapter 2, Section 2.5), such a scenario points to the possibility of improvement in the *quality* of the PF approximation.

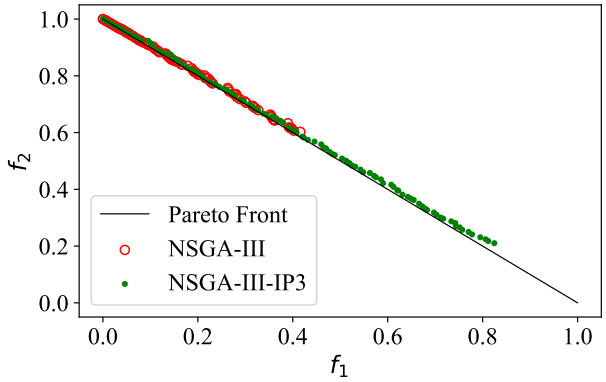
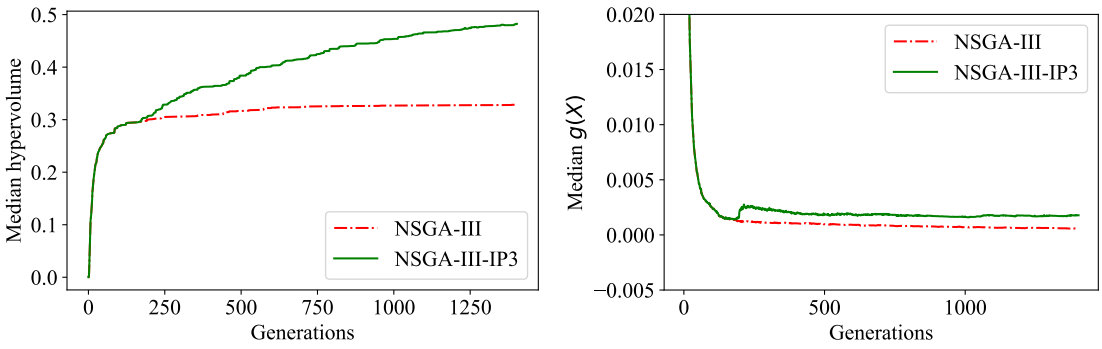


Figure 4.9. Final obtained solutions in the respective median runs of NSGA-III and NSGA-III-IP3 on CIBN1 problem.

As can be observed in Figure 4.9, NSGA-III-IP3 clearly achieved a better PF approximation than NSGA-III. Even though 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 towards diversity-enhancement.

For further insights into the performance, Figures 4.10a and 4.10b show the generation-wise median hypervolume and median $g(X)$ plots, respectively. In that, there is a significant improvement in the 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 $t_{\text{term}} = 1404$, the hypervolume measures for

NSGA-III-IP3 could not stabilize. This suggests that even though the population had stabilized as per the termination criterion defined, and the NSGA-III-IP3 run should have been terminated from a practical perspective, there was a scope of further improvement in spread across the PF.



(a) Hypervolume plot

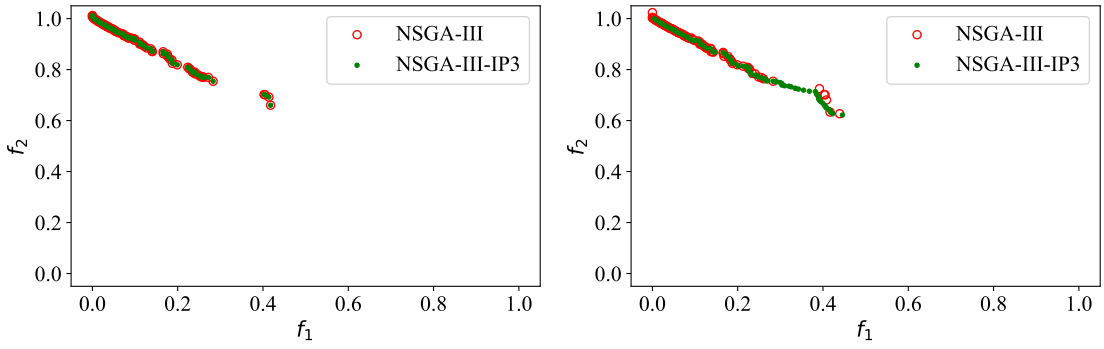
(b) $g(X)$ plot

Figure 4.10. Generation-wise performance of NSGA-III and NSGA-III-IP3 on CIBN1.

It may be noted that the IP3 operator attempts to cater to both aspects of diversity-enhancement, i.e., spread expansion and uniformity within the spread. While the first aspect has clearly been demonstrated through the discussion presented above in the context of CIBN1 problem, it is imperative to gather insights into the performance of IP3 operator in terms of achieving a better uniformity of solutions within a given spread. Towards it, the parent solutions in the generation $t_{\text{mild}} = 178$, right before the IP3 operator is invoked for the first time, are shown in Figure 4.11a; and the parent solutions at a subsequent arbitrary generation, say $t = 200$, are shown in Figure 4.11b. As can be observed at $t = t_{\text{mild}}$, the solutions of NSGA-III and NSGA-III-IP3 coincide, implying that NSGA-III-IP3 behaves in exactly the same manner as NSGA-III till 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 solutions has significantly improved for NSGA-III-IP3. This improvement in the uniformity of solutions can only be attributed to the efficacy of the IP3 operator.

4.5.2.2 MW12 Problem $(M = 2)$

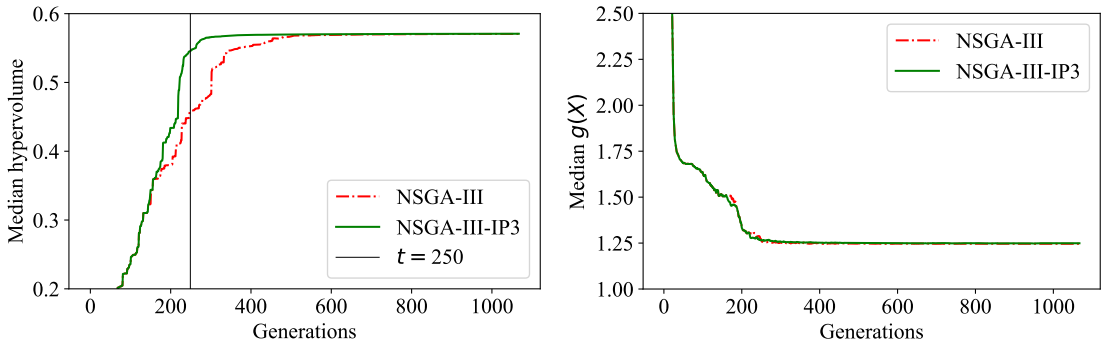
Here, MW12 problems is considered for discussion, where NSGA-III is able to achieve a reasonable PF-approximation. Reference may be made to Figures 4.12a and 4.12b, which present



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

Figure 4.11. 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 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_{term} had been set as 1068 for NSGA-III, as 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_{term} generations. Hence, as per the premise for interpretation of the results, laid earlier (Chapter 2, Section 2.5), the scope of possible enhancements by the IP3 operator, reduces to *speeding-up* of the PF-approximation. This is endorsed by the reference to an arbitrarily chosen generation $t = 250$ in Figure 4.12a, where NSGA-III-IP3 clearly achieved a better hypervolume than NSGA-III.

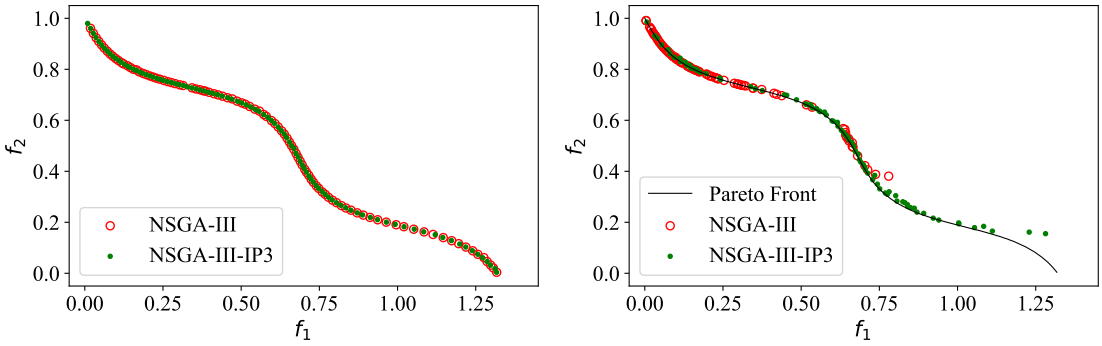


(a) Hypervolume plot (b) $g(X)$ plot

Figure 4.12. Generation-wise performance of NSGA-III and NSGA-III-IP3 on MW12.

Towards a deeper investigation, the solutions obtained in the respective median runs of NSGA-

III and NSGA-III-IP3, at $t = t_{\text{term}}$ and at $t = 250$ are shown in Figures 4.13a and 4.13b, respectively. In that, while the solutions at $t = t_{\text{term}}$ reflect the similar performance of NSGA-III and NSGA-III-IP3, the solutions at $t = 250$ clearly reflect the better performance of NSGA-III-IP3 in terms of diversity. Such an improvement sped up the PF approximation, which could only be attributed to the search efficacy infused by the IP3 operator into NSGA-III.



(a) Obtained solutions at $t = t_{\text{term}}$

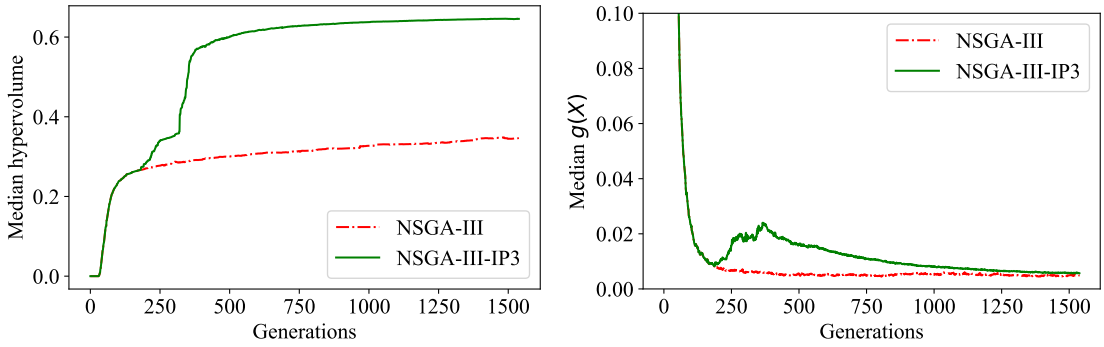
(b) Obtained solutions at $t = 250$

Figure 4.13. 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.

4.5.2.3 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, Figures 4.14a and 4.14b show 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, $t_{\text{term}} = 1539$ 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) similar 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 Figure 4.15. As evident, 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 proposed IP3 operator is suitable for diversity-enhancement in more than two objectives as well.



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

Figure 4.14. Generation-wise performance of NSGA-III and NSGA-III-IP3 on DASCMOP9.

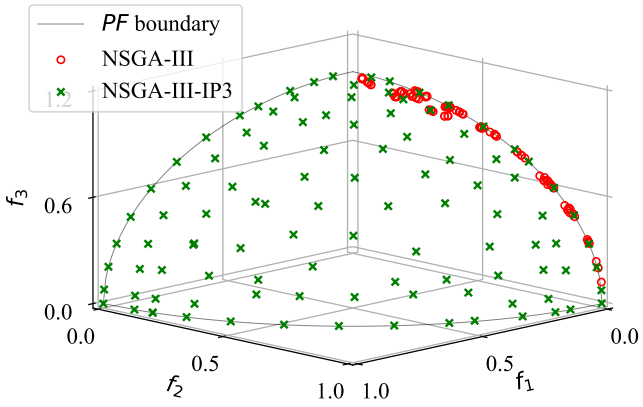


Figure 4.15. Final obtained solutions in the respective median runs of NSGA-III and NSGA-III-IP3 on DASCMOP9 problem.

4.5.3 IP3 Operator's Performance Sensitivity towards Variation in *k*

As discussed earlier in Section 4.1.2, setting the number of nearest neighbours k (in kNN) is important since keeping it very low or very high may lead to underfitting or overfitting, respectively. Although the choice of $k = n_{\text{var}}$ has been reasoned, it is imperative to analyze the IP3 operator's performance sensitivity towards the variation in k. In this background, the performance of NSGA-III-IP3 is presented here with three different settings of k, on all considered test problems. These include: (a) $k = 0.5n_{\text{var}}$, leading to a value lower than the proposed setting; (b) $k = n_{\text{var}}$, the proposed setting; and (c) $k = 1.5n_{\text{var}}$, leading to a value higher than the proposed setting. The median hypervolume obtained by NSGA-III-IP3 with all three settings of k, at the end of t_{term} generations determined on-the-fly for $k = n_{\text{var}}$, are shown in Table 4.3. In that, the best obtained hypervolume, and its statistically equivalent results are marked in bold. From Table 4.3,

Table 4.3. Hypervolume based comparison of NSGA-III-IP3, across different settings of k (used in the ML method). Here, t_{term} determined on-the-fly for NSGA-III-IP3 with $k = n_{\text{var}}$, has been used for other values of k. The best performing algorithm and the statistically equivalent algorithms are marked in bold.

	Problem	t_{term}	$k = 0.5n_{\text{var}}$	$k = n_{\text{var}}$	$k = 1.5n_{\text{var}}$
$M = 2$	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

following may be noted.

- With $k = n_{\text{var}}$ (proposed), the performance was either statistically better than or equivalent to other settings of k in all (28 out of 28) instances. As evident, the proposed setting of k performed well, compared to other settings.

- With $k = 0.5n_{\text{var}}$, the performance was statistically better than or equivalent to other settings of k in only 22 out of 28 instances. This deteriorated performance could be attributed to the lower value of k than desired, implying that the performance of the IP3 operator is

sensitive towards variation in k, for $k < n_{\text{var}}$.

- With $k = 1.5n_{\text{var}}$, 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 towards variation in k, for $k > n_{\text{var}}$.

The above endorses the use of $k = n_{\text{var}}$ for the kNN method in IP3 operator. Notably, only a limited variation in k has been investigated here, owing to the scope of this thesis that focuses on providing the proof-of-concept that ML methods could be used for such performance enhancements in RV-EMO algorithms, rather than tuning the parameters of these ML methods.

4.5.4 IP3 Settings vis-à-vis the Convergence-Diversity Balance and Risk-Rewards Tradeoff

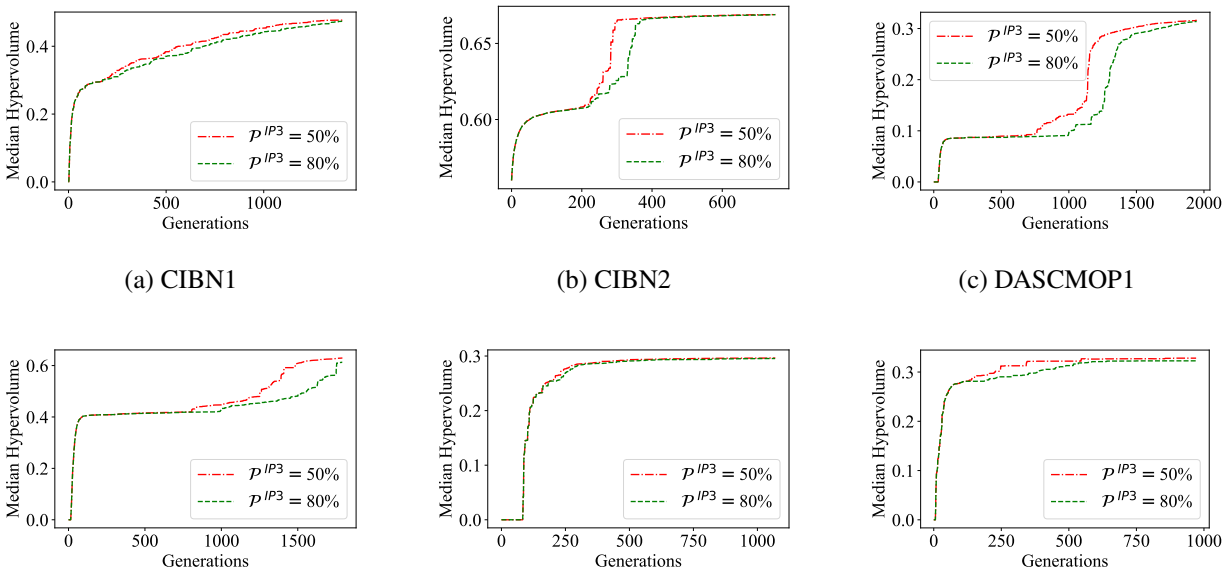
In view of the results presented in experimental investigation above, it is fair to infer that NSGA-III-IP3 effectively addresses the key considerations of convergence-diversity balance and risk-rewards tradeoff, in majority of the considered problems. Such an effective management, despite the use of a pro-diversity operator, could be attributed to the following settings that allow a dominant share of Q^{V} across the generations.

- In any generation of NSGA-III-IP3 run where the IP3 operator is invoked, only $\mathcal{P}^{\text{IP3}} = 50\%$ pro-diversity offspring are created.
- The invocations of the IP3 operator (governed by $t^{\text{IP3}}_{\text{freq}} \geq 1$) are adaptive, based on the survival rate of the pro-diversity offspring.

In the context of setting $\mathcal{P}^{\text{IP3}}$, following scenarios are possible.

- $\mathcal{P}^{\text{IP3}} < 50\%$: any such setting would inherently address the key considerations mentioned above, since the overall share of Q^{V} would be dominant in each generation as well as across all generations. However, it may lead to more frequent invocations of the IP3 operator compared to $\mathcal{P}^{\text{IP3}} = 50\%$, to overall produce the same number of pro-diversity offspring across all generations. This would lead to more ML model trainings, which is a computationally inefficient choice.
- $\mathcal{P}^{\text{IP3}} > 50\%$: any such setting would not ensure a dominant share of Q^{V} across all generations. As reasoned earlier in Section 1.2 (Chapter 1), this may hamper the management of the above considerations, consequently deteriorating the effectiveness of the IP3 operator.

While any setting of $\mathcal{P}^{\text{IP2}} < 50\%$ does not pose a major implication on the effectiveness of the IP2 operator, any setting of $\mathcal{P}^{\text{IP2}} > 50\%$ does. Hence, it is imperative to investigate the latter scenario. Towards this, the performances of IP2 operator with $\mathcal{P}^{\text{IP2}} = 50\%$ and with $\mathcal{P}^{\text{IP2}} = 80\%$ (a random value $> 50\%$), have been compared on some sample test problems. The generation-wise median hypervolume plots, from among the 31 randomly seeded runs, are shown in Figure 4.16. In that, the same t_{term} generations have been used as given in Table 4.2.



(a) CIBN1 (b) CIBN2 (c) DASCMOP1

(d) DASCMOP2 (e) MW9 (f) MW13

Figure 4.16. Results of NSGA-III-IP3 with different settings of $\mathcal{P}^{\text{IP3}}$ on some test problems.

From Figure 4.16, it may be noted that $\mathcal{P}^{\text{IP2}} = 80\%$ leads to a worse hypervolume in the intermediate generations, compared to $\mathcal{P}^{\text{IP2}} = 50\%$. This difference in performance may or may not be significant depending on the nature of problem, for instance: (a) the difference is not significant in CIBN1 and MW9, whereas (b) it is significant in other problems. Further, this difference could be attributed to either or both of: (a) over-skew towards diversity, and (b) excessive reliance on an ML-based operator. However, towards the end generations, the performances with $\mathcal{P}^{\text{IP3}} = 50\%$ and $\mathcal{P}^{\text{IP3}} = 80\%$ are similar. This could be attributed to the adaptive invocations of IP3 operator (through $t^{\text{IP3}}_{\text{freq}}$), that eventually led to lesser frequent invocations of IP3 operator in case of $\mathcal{P}^{\text{IP3}} = 80\%$, compared to $\mathcal{P}^{\text{IP3}} = 50\%$.

Chapter **5**

UIP Operator for Simultaneous Convergence and Diversity Enhancement

In the previous chapters, the IP2 and IP3 operators had been proposed, with sole focus on convergence- and diversity-enhancement, respectively. Interestingly, on convergence-hard problems: NSGA-III-IP2 reported (statistically) significantly better convergence over the base NSGA-III, without compromising on diversity. Similarly, on diversity-hard problems: NSGA-III-IP3 reported significantly better diversity over the base NSGA-III, without compromising on convergence. Such revelations testify that the delicate convergence-diversity balance which is essential for a good PF-approximation, could be retained, by ensuring that the convergence-diversity-neutral offspring produced by the natural variation operators, remained at least 50% of the total offspring, utilized across all the generations of NSGA-III-IP2 or NSGA-III-IP3.

Notably, a priori characterization of a given MOP, as convergence-hard or diversity-hard, is not a trivial task. Hence, it is critically important to have an operator that provides the scope for improvement in both convergence and diversity, without a priori assumption or knowledge of the MOPs' characteristics. To this effect, this chapter proposes the UIP operator, which invokes both IP2 and IP3 operators, for the creation of pro-convergence Q^{IP2} and pro-diversity Q^{IP3} offspring solutions, respectively. In that, the overall contribution of Q^{V} across all generations dominates the collective contribution of Q^{IP2} and Q^{IP3}, as symbolically depicted in Figure 1.3. The rationale behind ensuring this dominance of Q^{V} has been provided earlier is Section 1.2 (Chapter 1).

The remaining chapter is organized as follows: the proposed UIP operator is detailed in Section 5.1, along with its integration with some RV-EMO algorithms, including NSGA-III, θ-DEA and MOEA/DD. Its computational complexity is highlighted in Section 5.2, followed by its comparison with some common enhancements used in EMO algorithms in Section 5.3. The

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

Figure 5.1. Symbolic depiction of the convergence-diversity balance across all generations of RV-EMO-UIP vis-à-vis RV-EMO-IP3, RV-EMO-IP2 and RV-EMO. The offspring created using natural variation operators Q^{V} do not impose any explicit preference for either convergence or diversity.

experimental settings are discussed in Section 5.4, followed by results in Sections 5.5 and 5.6. Finally, a small analysis of the additional run time associated with the UIP operator is presented in Section 5.7.

5.1 Proposed UIP Operator for Convergence and Diversity Enhancement

It has been highlighted above that the UIP operator relies on the invocations of both IP2 and IP3 operators, in a manner that the delicate convergence-diversity balance is not disrupted. It may be noted that in Chapter 3, the constitutive modules of the IP2 operator were presented, and their integration with NSGA-III was summarized in Algorithm 3.5, in reference to any generation t of NSGA-III-IP2. Similarly, in Chapter 4, the constitutive modules of the IP3 operator were presented, and their integration with NSGA-III was summarized in Algorithm 4.5, in reference to any generation t of NSGA-III-IP3. With an aim to avoid clutter in the algorithmic description of NSGA-III-UIP, it is important to define the IP2 and IP3 operators as compact, yet self-sufficient *functions*, that could be suitably invoked, as part of NSGA-III-UIP. Considering this, the remaining section is structured as follows: (a) the IP2 and IP3 operators are defined as compact *functions*, and (b) the integration of IP2 and IP3, leading up to the UIP operator is presented.

5.1.1 The IP2 operator's representation as a *Function*

The IP2 operator as a *function* is presented in Algorithm 5.1. It includes: (a) construction of the training-dataset D_t using Algorithm 3.2; (b) training of the ML model on D_t using Algorithm 3.3; (c) creation of $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$ offspring solutions using the natural variation operators, denoted as Q_t^{V}; and (d) advancement of all the offspring solutions in Q_t^{V} using Algorithm 3.4, leading to advanced

offspring solutions Q_t^{IP2} (sized $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$).

Algorithm 5.1: `IP2`$(A_t, T_t, \mathcal{R}, [x^l, x^u], P_t, \mathcal{P}^{\mathrm{IP2}})$

Input: Input archive A_t, target archive T_t, RV set $\mathcal{R}$, original variable bounds $[x^l, x^u]$, parent population P_t, proportion of the offspring advanced using IP2 $\mathcal{P}^{\mathrm{IP2}}$

Output: Offspring solutions $Q_t^{(2)}$

1 $D_t \leftarrow$ `Archive_Mapping`$(A_t, T_t, \mathcal{R})$

2 $ML, [x^{\min}, x^{\max}] \leftarrow$ `Training`$(D_t, [x^l, x^u])$

3 $Q_t^{\mathrm{V}} \leftarrow$ `Variation`(P_t) `% sized` $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$

4 $Q_t^{\mathrm{IP2}} \leftarrow$ `Progression`$(Q_t^{\mathrm{V}}, ML, [x^{\min}, x^{\max}], [x^l, x^u], \mathcal{P}^{\mathrm{IP2}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$

5.1.2 The IP3 operator's representation as a *Function*

The IP3 operator as a *function* is presented in Algorithm 5.2. It includes: (a) construction of M training-datasets $\mathcal{D}_1$–$\mathcal{D}_M$ using Algorithm 4.1; (b) training of M ML models on $\mathcal{D}_1$–$\mathcal{D}_M$ (one per dataset) using Algorithm 4.2; (c) creation of $\lfloor \mathcal{P}^{\mathrm{IP3}} N/2 \rfloor$ offspring solutions, Q_t^B, using Algorithm 4.3 for a better spread of solutions; (d) creation of $\lfloor \mathcal{P}^{\mathrm{IP3}} N/2 \rfloor$ offspring solutions, Q_t^G, using Algorithm 4.4 for a better uniformity of solutions, and (e) merging of Q_t^B and Q_t^G, leading to $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$ offspring solutions, denoted as Q_t^{IP3}.

Algorithm 5.2: `IP3`$(P_t, \mathcal{R}, r, [x^l, x^u], \mathcal{P}^{\mathrm{IP3}})$

Input: Parent population P_t, RV set $\mathcal{R}$, neighbourhood radius r, original variable bounds $[x^l, x^u]$, proportion of the offspring created using IP3 $\mathcal{P}^{\mathrm{IP3}}$

Output: Offspring solutions $Q_t^{(3)}$

1 $\mathcal{D} \leftarrow$ `Dataset_Construction`$(P_t, \mathcal{R}, r)$

2 $ML, [x^{\min}, x^{\max}] \leftarrow$ `Training`$(\mathcal{D}, [x^l, x^u])$

3 $Q_t^B \leftarrow$ `Boundary_Progression`$(P_t, \mathcal{R}, ML, [x^{\min}, x^{\max}], [x^l, x^u], \mathcal{P}^{\mathrm{IP3}})$ `%` $\lfloor \mathcal{P}^{\mathrm{IP3}} N/2 \rfloor$

4 $Q_t^G \leftarrow$ `Gap_Progression`$(P_t, \mathcal{R}, ML, [x^{\min}, x^{\max}], [x^l, x^u], \mathcal{P}^{\mathrm{IP3}})$ `%` $\lfloor \mathcal{P}^{\mathrm{IP3}} N/2 \rfloor$

5 $Q_t^{\mathrm{IP3}} \leftarrow Q_t^B \cup Q_t^G$ `%` $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$

5.1.3 Formalization of the UIP operator, and its integration with NSGA-III

This section formalizes the UIP operator, as one which pursues improvement in both convergence and diversity, by independently invoking the IP2 and IP3 operators. In that:

- the criterion for the first time invocations of the IP2 and IP3 operators remain the same as before, implying that IP2 is invoked when the underlying RV-EMO population becomes entirely non-dominated, while IP3 is invoked when the underlying RV-EMO population indicates mild stability.

- the successive invocations of IP2 and IP3 are guided by independent parameters, namely, $t^{\text{IP2}}_{\text{freq}}$ and $t^{\text{IP3}}_{\text{freq}}$, respectively. It implies, that in any generation t of an RV-EMO algorithm, either or both of IP2 and IP3 operators may be active.

In this background, a representative generation t of NSGA-III-UIP has been summarized in Algorithm 5.3. In that, first the target-archive T_t as required by the IP2 function is updated (line 1, Algorithm 5.3). 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 2–7, Algorithm 5.3). In the subsequent generations:

- if $t^{\text{IP2}}_{\text{freq}}$ generations have passed after the last invocation of IP2 (at $t = t^{\text{IP2}}_{\text{pg}}$), then IP2 is invoked and $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ offspring solutions are created, denoted as Q^{IP2}_t (lines 8–10, Algorithm 5.3).

- similarly, if $t^{\text{IP3}}_{\text{freq}}$ generations have passed after the last invocation of IP3 (at $t = t^{\text{IP3}}_{\text{pg}}$), then IP3 is invoked and $\lfloor \mathcal{P}^{\text{IP3}} N \rfloor$ offspring solutions are created, denoted as Q^{IP3}_t (lines 11–13, Algorithm 5.3).

- if the total count of offspring created in the above two steps ($|Q^{\text{IP2}}| + |Q^{\text{IP3}}|$) is smaller than N, then rest of the offspring are created using the natural variation operators (lines 14–15, Algorithm 5.3).

- the offspring solutions Q^{IP2}_t, Q^{IP3}_t and Q^{V}_t are merged into Q_t, sized N. Given that $\mathcal{P}^{\text{IP2}} = \mathcal{P}^{\text{IP3}} = 50\%$ has been used in this thesis as reasoned earlier in Section 1.2 (Chapter 1), the possible shares of $\{Q^{\text{IP2}}_t, Q^{\text{IP3}}_t, Q^{\text{V}}_t\}$ include: (i) $\{0, 0, 100\}\%$; (ii) $\{50, 0, 50\}\%$; (iii) $\{0, 50, 50\}\%$; and (iv) $\{50, 50, 0\}\%$. Then the offspring solutions Q_t are evaluated (line 16, Algorithm 5.3).

- Q_t is used to update an input-archive A_{t+1}, as required by the IP2 function, followed by NSGA-III's survival selection (lines 17–18, Algorithm 5.3).

Algorithm 5.3: Generation t of NSGA-III-UIP

Input: RV set $\mathcal{R}$, variable bounds $[x^l, x^u]$, parent population P_t, offspring survived $N_{t-1}^{\mathrm{surv(V)}}$

IP2-specific: target archive T_{t-1}, input archive A_t, frequency $t_{\mathrm{freq}}^{\mathrm{IP2}}$, last invocation $t_{\mathrm{pg}}^{\mathrm{IP2}}$, proportion $\mathcal{P}^{\mathrm{IP2}}$

IP3-specific: neighbourhood radius r, frequency $t_{\mathrm{freq}}^{\mathrm{IP3}}$, last invocation $t_{\mathrm{pg}}^{\mathrm{IP3}}$, proportion $\mathcal{P}^{\mathrm{IP3}}$

Output: P_{t+1}, T_t, A_{t+1}, $t_{\mathrm{freq}}^{\mathrm{IP2}}$, $t_{\mathrm{freq}}^{\mathrm{IP3}}$, $t_{\mathrm{pg}}^{\mathrm{IP2}}$, $t_{\mathrm{pg}}^{\mathrm{IP3}}$

1 $T_t \leftarrow$ `Update_Target_Archive`$(P_t, T_{t-1}, \mathcal{R})$ `% Algorithm 3.1`

2 $check1 \leftarrow$ `Check_Non_Dominated`(P_t)

3 $check2 \leftarrow$ `Check_Mild_Stabilization`$()$

4 **if** $check1$ **then**

5 $\quad startIP2 = True$

6 **if** $check2$ **then**

7 $\quad startIP3 = True$

8 **if** $startIP2$ *and* $t - t_{\mathrm{pg}}^{\mathrm{IP2}} = t_{\mathrm{freq}}^{\mathrm{IP2}}$ **then**

9 $\quad Q_t^{\mathrm{IP2}} \leftarrow$ `IP2`$(A_t, T_t, \mathcal{R}, [x^l, x^u], P_t, \mathcal{P}^{\mathrm{IP2}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$

10 $\quad t_{\mathrm{pg}}^{\mathrm{IP2}} \leftarrow t$

11 **if** $startIP3$ *and* $t - t_{\mathrm{pg}}^{\mathrm{IP3}} = t_{\mathrm{freq}}^{\mathrm{IP3}}$ **then**

12 $\quad Q_t^{\mathrm{IP3}} \leftarrow$ `IP3`$(P_t, \mathcal{R}, r, [x^l, x^u], \mathcal{P}^{\mathrm{IP3}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$

13 $\quad t_{\mathrm{pg}}^{\mathrm{IP3}} \leftarrow t$

14 **if** $|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}| < N$ **then**

15 $\quad Q_t^{\mathrm{V}} \leftarrow$ `Variation`(P_t) `% sized` $N - (|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}|)$

16 `Evaluate`(Q_t), where $Q_t \equiv Q_t^{\mathrm{IP2}} \cup Q_t^{\mathrm{IP3}} \cup Q_t^{\mathrm{V}}$ `% size` N

17 $A_{t+1} \leftarrow (A_t \cup Q_t \cup P_{t+1-t_{\mathrm{past}}}) \backslash [P_{t-t_{\mathrm{past}}} \cup Q_{t-t_{\mathrm{past}}}]$

18 $P_{t+1} \leftarrow$ `Survival_Selection`$(P_t \cup Q_t)$

19 **if** $t = t_{\mathrm{pg}}^{\mathrm{IP2}}$ **then**

20 $\quad t_{\mathrm{freq}}^{\mathrm{IP2}} \leftarrow$ `Adapt`$(Q_t^{\mathrm{IP2}}, P_{t+1}, Q_{t-1}^{\mathrm{V}}, N_{t-1}^{\mathrm{surv(V)}}, t_{\mathrm{freq}}^{\mathrm{IP2}})$ `% Algorithm 5.4`

21 **if** $t = t_{\mathrm{pg}}^{\mathrm{IP3}}$ **then**

22 $\quad t_{\mathrm{freq}}^{\mathrm{IP3}} \leftarrow$ `Adapt`$(Q_t^{\mathrm{IP3}}, P_{t+1}, Q_{t-1}^{\mathrm{V}}, N_{t-1}^{\mathrm{surv(V)}}, t_{\mathrm{freq}}^{\mathrm{IP3}})$ `% Algorithm 5.4`

- finally, $t_{\mathrm{freq}}^{\mathrm{IP2}}$ and $t_{\mathrm{freq}}^{\mathrm{IP3}}$ are adapted, if the respective operators were invoked in the current generation t (lines 19–22, Algorithm 5.3). The procedure for this adaptation has been discussed below.

The procedure for adapting both $t_{\mathrm{freq}}^{\mathrm{IP2}}$ and $t_{\mathrm{freq}}^{\mathrm{IP3}}$ is exactly the same. Hence, a generic adaptation procedure has been presented for 'IPj' in Algorithm 5.4, where j = 2 and 3 correspond to IP2 and IP3 functions, respectively. In that procedure,

Algorithm 5.4: `Adapt`$(Q_t^{\mathrm{IPj}}, P_{t+1}, Q_{t-1}^{\mathrm{V}}, N_{t-1}^{\mathrm{surv(V)}}, t_{\mathrm{freq}}^{\mathrm{IPj}})$

Input: Offspring created by IPj in current generation Q_t^{IPj}, survived population P_{t+1}, offspring created by variation operators in previous generation Q_{t-1}^{V}, number of offspring in Q_{t-1}^{V} that survived in $(t-1)^{\mathrm{th}}$ generation $N_{t-1}^{\mathrm{surv(V)}}$, and frequency $t_{\mathrm{freq}}^{\mathrm{IPj}}$

Output: $t_{\mathrm{freq}}^{\mathrm{IPj}}$

1 $N_t^{\mathrm{surv(IPj)}} \leftarrow sizeof(Q_t^{\mathrm{IPj}} \cap P_{t+1})$
2 $S_{\mathrm{IPj}} \leftarrow N_t^{\mathrm{surv(IPj)}} / |Q_t^{\mathrm{IPj}}|$
3 $S_{\mathrm{V}} \leftarrow N_{t-1}^{\mathrm{surv(V)}} / |Q_{t-1}^{\mathrm{V}}|$
4 **if** $S_{\mathrm{IPj}} > S_{\mathrm{V}}$ **then**
5 $\quad t_{\mathrm{freq}}^{\mathrm{IPj}} \leftarrow t_{\mathrm{freq}}^{\mathrm{IPj}} - 1$
6 **if** $S_{\mathrm{IPj}} < S_{\mathrm{V}}$ **then**
7 $\quad t_{\mathrm{freq}}^{\mathrm{IPj}} \leftarrow t_{\mathrm{freq}}^{\mathrm{IPj}} + 1$
8 **if** $t_{\mathrm{freq}}^{\mathrm{IPj}} < 2$ **then**
9 $\quad t_{\mathrm{freq}}^{\mathrm{IPj}} = 2$

- first, the offspring solutions in Q_t^{IPj} that are selected through NSGA-III's survival selection are counted (line 1, Algorithm 5.4). This count is denoted as $N_t^{\mathrm{surv(IPj)}}$.

- then, the survival rate of the offspring solutions: (i) created through IPj in current generation, S_{IPj}; and (ii) created through natural variation operators in previous generation, S_{V}, are computed (lines 2–3, Algorithm 5.4).

- if $S_{\mathrm{IPj}} > S_{\mathrm{V}}$, implying a better performance of IPj operator than variation operators, $t_{\mathrm{freq}}^{\mathrm{IPj}}$ is reduced by 1, leading to a more frequent invocation of the IPj function.

- if $S_{\mathrm{IPj}} < S_{\mathrm{V}}$, implying a worse performance of IPj operator than variation operators, $t_{\mathrm{freq}}^{\mathrm{IPj}}$ is increased by 1, leading to a lesser frequent invocation of the IPj function.

- finally, it is ensured that $t_{\mathrm{freq}}^{\mathrm{IPj}}$ satisfies its minimum value of 2 (lines 8–9, Algorithm 5.4). The rationale behind this minimum value has been provided earlier in Section 1.2 (Chapter 1), in the context of risk-rewards tradeoff.

5.1.4 UIP operator's integration with other RV-EMO algorithms

As mentioned earlier in Chapter 1, the proposed UIP operator is generic, and can ideally be integrated with any RV-EMO algorithm. To facilitate its implementation, two RV-EMO algorithms

(in addition to NSGA-III) have been selected, including, θ-DEA and MOEA/DD. The algorithmic details of the resulting θ-DEA-UIP and MOEA/DD-UIP algorithms are provided below.

5.1.4.1 θ-DEA-UIP

The algorithmic description of any generation t of θ-DEA-UIP is summarized in Algorithm 5.5. In that, first the target-archive T_t as required by the IP2 function is updated (line 1, Algorithm 5.5). 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 2–7, Algorithm 5.5). In the subsequent generations:

- if $t^{\mathrm{IP2}}_{\mathrm{freq}}$ generations have passed after the last invocation of IP2 (at $t = t^{\mathrm{IP2}}_{\mathrm{pg}}$), then IP2 is invoked and $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$ offspring solutions are created, denoted as Q^{IP2}_t (lines 8–10, Algorithm 5.5).

- similarly, if $t^{\mathrm{IP3}}_{\mathrm{freq}}$ generations have passed after the last invocation of IP3 (at $t = t^{\mathrm{IP3}}_{\mathrm{pg}}$), then IP3 is invoked and $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$ offspring solutions are created, denoted as Q^{IP3}_t (lines 11–13, Algorithm 5.5).

- if the total count of offspring created in the above two steps ($|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}|$) is smaller than N, then rest of the offspring are created using the natural variation operators (lines 14–15, Algorithm 5.5).

- the offspring solutions Q^{IP2}_t, Q^{IP3}_t and Q^{V}_t are merged into Q_t, sized N. Then the offspring solutions Q_t are evaluated (line 16, Algorithm 5.5).

- Q_t is used to update an input-archive A_{t+1}, as required by the `IP2` function (line 17, Algorithm 5.5).

- the steps in lines 18–30 (Algorithm 5.5) relate to the steps of the survival selection procedure of θ-DEA [Yuan et al., 2016].

- finally, $t^{\mathrm{IP2}}_{\mathrm{freq}}$ and $t^{\mathrm{IP3}}_{\mathrm{freq}}$ are adapted, if the respective operators were invoked in the current generation t (lines 31–34, Algorithm 5.5).

Algorithm 5.5: Generation t of θ-DEA-UIP

Input: RV set $\mathcal{R}$, variable bounds $[x^l, x^u]$, parent population P_t, offspring survived $N_{t-1}^{\mathrm{surv(V)}}$

IP2-specific: target archive T_{t-1}, input archive A_t, frequency $t_{\mathrm{freq}}^{\mathrm{IP2}}$, last invocation $t_{\mathrm{pg}}^{\mathrm{IP2}}$, proportion $\mathcal{P}^{\mathrm{IP2}}$

IP3-specific: neighbourhood radius r, frequency $t_{\mathrm{freq}}^{\mathrm{IP3}}$, last invocation $t_{\mathrm{pg}}^{\mathrm{IP3}}$, proportion $\mathcal{P}^{\mathrm{IP3}}$

Output: P_{t+1}, T_t, A_{t+1}, $t_{\mathrm{freq}}^{\mathrm{IP2}}$, $t_{\mathrm{freq}}^{\mathrm{IP3}}$, $t_{\mathrm{pg}}^{\mathrm{IP2}}$, $t_{\mathrm{pg}}^{\mathrm{IP3}}$

$T_t \leftarrow$ `Update_Target_Archive`$(P_t, T_{t-1}, \mathcal{R})$ `% Algorithm 3.1`

$check1 \leftarrow$ `Check_Non_Dominated`(P_t)

$check2 \leftarrow$ `Check_Mild_Stabilization`$()$

if $check1$ **then**

  $startIP2 = True$

if $check2$ **then**

  $startIP3 = True$

if $startIP2$ *and* $t - t_{\mathrm{pg}}^{\mathrm{IP2}} = t_{\mathrm{freq}}^{\mathrm{IP2}}$ **then**

  $Q_t^{\mathrm{IP2}} \leftarrow$ `IP2`$(A_t, T_t, \mathcal{R}, [x^l, x^u], P_t, \mathcal{P}^{\mathrm{IP2}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$

  $t_{\mathrm{pg}}^{\mathrm{IP2}} \leftarrow t$

if $startIP3$ *and* $t - t_{\mathrm{pg}}^{\mathrm{IP3}} = t_{\mathrm{freq}}^{\mathrm{IP3}}$ **then**

  $Q_t^{\mathrm{IP3}} \leftarrow$ `IP3`$(P_t, \mathcal{R}, r, [x^l, x^u], \mathcal{P}^{\mathrm{IP3}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$

  $t_{\mathrm{pg}}^{\mathrm{IP3}} \leftarrow t$

if $|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}| < N$ **then**

  $Q_t^{\mathrm{V}} \leftarrow$ `Variation`(P_t) `% sized` $N - (|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}|)$

`Evaluate`(Q_t), where $Q_t \equiv Q_t^{\mathrm{IP2}} \cup Q_t^{\mathrm{IP3}} \cup Q_t^{\mathrm{V}}$ `% size` N

$A_{t+1} \leftarrow (A_t \cup Q_t \cup P_{t+1-t_{\mathrm{past}}}) \backslash [P_{t-t_{\mathrm{past}}} \cup Q_{t-t_{\mathrm{past}}}]$

$R_t \leftarrow P_t \cup Q_t$

$S_t \leftarrow$ `Get_Pareto_Nondomination_Levels`(R_t)

`Update_Ideal_Point`(S_t)

`Normalize` $(S_t, \mathbf{z}^*, \mathbf{z}^{\mathrm{nad}})$

$\mathcal{C} \leftarrow$ `Clustering`$(S_t, \mathcal{R})$

$\{F_1', F_2', \ldots\} \leftarrow \theta$`-Nondominated_Sort`$(S_t, \mathcal{C})$

$P_{t+1} \leftarrow \emptyset$

$i \leftarrow 1$

while $|P_{t+1}| + |F_i'| < N$ **do**

  $P_{t+1} \leftarrow P_{t+1} \cup F_i'$

  $i \leftarrow i + 1$

== Continued on the next page ==

```
                    == In continuation from the previous page ==
25 Random_Sort (F'_i)
26 P_{t+1} ← P_{t+1} ∪ F'_i[N − |P_{t+1}|]
27 if t = t^{IP2}_{pg} then
28     t^{IP2}_{freq} ← Adapt(Q^{IP2}_t, P_{t+1}, Q^{V}_{t−1}, N^{surv(V)}_{t−1}, t^{IP2}_{freq})
29 if t = t^{IP3}_{pg} then
30     t^{IP3}_{freq} ← Adapt(Q^{IP3}_t, P_{t+1}, Q^{V}_{t−1}, N^{surv(V)}_{t−1}, t^{IP3}_{freq})
```

5.1.4.2 MOEA/DD-UIP

The algorithmic description of any generation t of MOEA/DD-UIP is summarized in Algorithm 5.6. In that, first the target-archive T_t as required by the IP2 function is updated (line 1, Algorithm 5.6). 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 2–7, Algorithm 5.6). In the subsequent generations:

- if $t^{\mathrm{IP2}}_{\mathrm{freq}}$ generations have passed after the last invocation of IP2 (at $t = t^{\mathrm{IP2}}_{\mathrm{pg}}$), then IP2 is invoked and $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$ offspring solutions are created, denoted as Q^{IP2}_t (lines 8–10, Algorithm 5.6).

- similarly, if $t^{\mathrm{IP3}}_{\mathrm{freq}}$ generations have passed after the last invocation of IP3 (at $t = t^{\mathrm{IP3}}_{\mathrm{pg}}$), then IP3 is invoked and $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$ offspring solutions are created, denoted as Q^{IP3}_t (lines 11–13, Algorithm 5.6).

- if the total count of offspring created in the above two steps ($|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}|$) is smaller than N, then rest of the offspring are created using the natural variation operators (lines 14–16, Algorithm 5.6).

- the offspring solutions Q^{IP2}_t, Q^{IP3}_t and Q^{V}_t are merged into Q_t, sized N. Then the offspring solutions Q_t are evaluated (line 17, Algorithm 5.6).

- Q_t is used to update an input-archive A_{t+1}, as required by the `IP2` function (line 18, Algorithm 5.6).

- the steps in lines 19–21 (Algorithm 5.6) relate to the steps of the survival selection procedure of MOEA/DD [Li et al., 2015a].

- finally, $t^{\mathrm{IP2}}_{\mathrm{freq}}$ and $t^{\mathrm{IP3}}_{\mathrm{freq}}$ are adapted, if the respective operators were invoked in the current generation t (lines 22–25, Algorithm 5.6).

Algorithm 5.6: Generation t of MOEA/DD-UIP

Input: RV set $\mathcal{R}$, variable bounds $[x^l, x^u]$, parent population P_t, offspring survived $N^{\mathrm{surv(V)}}_{t-1}$

IP2-specific: target archive T_{t-1}, input archive A_t, frequency $t^{\mathrm{IP2}}_{\mathrm{freq}}$, last invocation $t^{\mathrm{IP2}}_{\mathrm{pg}}$, proportion $\mathcal{P}^{\mathrm{IP2}}$

IP3-specific: neighbourhood radius r, frequency $t^{\mathrm{IP3}}_{\mathrm{freq}}$, last invocation $t^{\mathrm{IP3}}_{\mathrm{pg}}$, proportion $\mathcal{P}^{\mathrm{IP3}}$

Output: $P_{t+1}, T_t, A_{t+1}, t^{\mathrm{IP2}}_{\mathrm{freq}}, t^{\mathrm{IP3}}_{\mathrm{freq}}, t^{\mathrm{IP2}}_{\mathrm{pg}}, t^{\mathrm{IP3}}_{\mathrm{pg}}$

1 $T_t \leftarrow$ `Update_Target_Archive`$(P_t, T_{t-1}, \mathcal{R})$ `% Algorithm 3.1`

2 $check1 \leftarrow$ `Check_Non_Dominated`(P_t)

3 $check2 \leftarrow$ `Check_Mild_Stabilization`$()$

4 **if** $check1$ **then**

5 $\quad startIP2 = True$

6 **if** $check2$ **then**

7 $\quad startIP3 = True$

8 **if** $startIP2$ *and* $t - t^{\mathrm{IP2}}_{\mathrm{pg}} = t^{\mathrm{IP2}}_{\mathrm{freq}}$ **then**

9 $\quad Q^{\mathrm{IP2}}_t \leftarrow$ `IP2`$(A_t, T_t, \mathcal{R}, [x^l, x^u], P_t, \mathcal{P}^{\mathrm{IP2}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP2}} N \rfloor$

10 $\quad t^{\mathrm{IP2}}_{\mathrm{pg}} \leftarrow t$

11 **if** $startIP3$ *and* $t - t^{\mathrm{IP3}}_{\mathrm{pg}} = t^{\mathrm{IP3}}_{\mathrm{freq}}$ **then**

12 $\quad Q^{\mathrm{IP3}}_t \leftarrow$ `IP3`$(P_t, \mathcal{R}, r, [x^l, x^u], \mathcal{P}^{\mathrm{IP3}})$ `% sized` $\lfloor \mathcal{P}^{\mathrm{IP3}} N \rfloor$

13 $\quad t^{\mathrm{IP3}}_{\mathrm{pg}} \leftarrow t$

14 **if** $|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}| < N$ **then**

15 $\quad \bar{P}_t \leftarrow$ `Mating_Selection`(P_t)

16 $\quad Q^{\mathrm{V}}_t \leftarrow$ `Variation`$(\bar{P}_t)$ `% sized` $N - (|Q^{\mathrm{IP2}}| + |Q^{\mathrm{IP3}}|)$

17 `Evaluate`(Q_t), where $Q_t \equiv Q^{\mathrm{IP2}}_t \cup Q^{\mathrm{IP3}}_t \cup Q^{\mathrm{V}}_t$ `% size` N

18 $A_{t+1} \leftarrow (A_t \cup Q_t \cup P_{t+1-t_{\mathrm{past}}}) \backslash [P_{t-t_{\mathrm{past}}} \cup Q_{t-t_{\mathrm{past}}}]$

19 $P_{t+1} \leftarrow P_t$

20 **for** $q \in Q_t$ **do**

21 $\quad P_{t+1} \leftarrow$ `Update_Population`(P_{t+1}, q)

22 **if** $t = t^{\mathrm{IP2}}_{\mathrm{pg}}$ **then**

23 $\quad t^{\mathrm{IP2}}_{\mathrm{freq}} \leftarrow$ `Adapt`$(Q^{\mathrm{IP2}}_t, P_{t+1}, Q^{\mathrm{V}}_{t-1}, N^{\mathrm{surv(V)}}_{t-1}, t^{\mathrm{IP2}}_{\mathrm{freq}})$

24 **if** $t = t^{\mathrm{IP3}}_{\mathrm{pg}}$ **then**

25 $\quad t^{\mathrm{IP3}}_{\mathrm{freq}} \leftarrow$ `Adapt`$(Q^{\mathrm{IP3}}_t, P_{t+1}, Q^{\mathrm{V}}_{t-1}, N^{\mathrm{surv(V)}}_{t-1}, t^{\mathrm{IP3}}_{\mathrm{freq}})$

5.2 Computational Complexity of UIP operator

As detailed earlier in Section 5.1, the proposed UIP operator is constituted by two modules. These modules are based on the IP2 and IP3 operators, whose computational complexity analysis has already been presented in Section 3.3 (Chapter 3) and Section 4.3 (Chapter 4), respectively. From that analysis, the worst case time and space complexities of IP2 based offspring advancement and IP3 based offspring creation, are summarized in Table 5.1.

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

Module	Time-complexity	Space-complexity
IP2 based offspring advancement	$\mathcal{O}(N^3 t_{\text{past}}^3 n_{\text{var}} \log(N t_{\text{past}}))$	$\mathcal{O}(N^2 t_{\text{past}}^2 n_{\text{var}})$
IP3 based offspring creation	$\mathcal{O}(MN n_{\text{var}} \log(N))$	$\mathcal{O}(MN n_{\text{var}})$

It may be noted that these worst case complexities, for both IP2 and IP3, correspond to their underlying ML methods, i.e., RF and kNN, respectively. The use of a different ML method may affect the corresponding complexities in Table 5.1. However, since the focus of this thesis is to provide a proof-of-concept that ML methods could be utilized for such enhancements relating to convergence and diversity in RV-EMO algorithms, the choice of ML method has not been incorporated within the scope of this thesis.

5.3 Comparison with Some Existing Enhancements

The concept of the UIP operator (and underlying IP2 and IP3 operators), as proposed in this thesis, may seem to be similar to certain existing enhancements used in the EMO 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 with the proposed operators are discussed. In this discussion, the proposed IP2, IP3 and UIP operators are collectively referred to as the IP operators.

5.3.1 A Local Search Method

The IP operators attempt to advance/create the offspring solutions through their ML-based progression, at any intermediate generation on an RV-EMO run. This operation may not be confused with a local search method that aims to improve the local convergence of solutions at any gener-

ation. The key differences are highlighted below.

- The presence of multiple conflicting objectives (usually the case with any MOP) 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. In contrast, multiple objectives do not pose any additional challenge for any of the IP operators.

- Implementing a local search necessitates additional solution evaluations beyond the default solution evaluations of any EMO algorithm, which is not the case with any of the IP operators.

- A local search usually means searching for the best solution in a local neighbourhood, while the progression of any solution's X-vector using any of the IP operators may be substantial, and not necessarily be locAlgorithm

5.3.2 A Surrogate-modeling Method

In EMO algorithms, a surrogate model is often constructed and used as a search basis, to evolve the solutions toward the PF. In real-world problems, where the solution evaluations are computationally very expensive, such an approach helps in reducing the number of actual solution evaluations needed to converge, hence saving computational effort and run time, as discussed earlier in Section 2.1.1 (Chapter 2). In past studies, several ML models have been used for building surrogate-models, including RF. In this sense, one might mistake the proposed ML-based IP operators as being nothing but a standard ML-based surrogate-modeling method.

However, the two differ significantly in terms of their scope and methodology, the key highlights of which are presented in Table 5.2. While the eventual goal of ML-based surrogate modeling is to help replace the actual objective and constraint computations with their approximations (via surrogates), the IP operators aspire to advance/create offspring solutions. In terms of methodology, the ML-based surrogate-models learn from an X-F mapping, while the IP operators utilize ML method(s) to learn from an X-X mapping. This explains the differences in terms of ML model application, evolution of offspring, and mode of fitness evaluation, as cited in the Table 5.2. While the proposed IP operators provide an alternate method to exploit current and past solutions to create new and, hopefully, better offspring solutions, it could also be profitably used in conjunction with a surrogate-modeling method in addressing the same problem.

Table 5.2. Fundamental differences between an EMO algorithm coupled with surrogate-modeling and with one of the proposed IP operators (IP2, IP3 or UIP).

Point of difference	EMO with surrogate-modeling	EMO with IP operators
Learning with ML model(s)	X-F mapping	X-X mapping
ML model application	Does not alter a solution's X-vector directly	Alters a solution's X-vector directly
Evolution of Offspring	Performed by variation operators only	Performed by the variation operators and/or the learnt ML model(s)
Fitness evaluation	Guided by both approximate and actual objective values	Guided by actual objective values only

It is notable that the use of a surrogate-model in an EMO algorithm can only offer a speed-up in convergence. In other words, a surrogate-model may help achieve convergence in lesser solution evaluations than required by the base EMO algorithm alone. However, in MOPs where the base EMO algorithm fails to reasonably approximate the PF, the use of a surrogate-model may not help achieve a better PF-approximation. However, the proposed IP operators (IP2, IP3 and UIP) may help in improving the quality of the PF-approximation, owing to the fact that these operators directly participate in the evolution of offspring solutions.

5.4 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-EMO algorithm(s) and the UIP operator.

5.4.1 Test-suite

To demonstrate the efficacy of the UIP operator, first, several multi-objective problems have been used, as used earlier in Chapters 3 and 4. These include: (i) convergence-hard (ZDT, DTLZ and MaF); and (ii) diversity-hard (CIBN, DASCMOP and MW) problems.

Once the efficacy of the UIP operator is established on multi-objective test problems, several many-objective test problems with $M = 5$, 8 and 10 have also been considered, that include DTLZ [Deb et al., 2005] and MaF [Cheng et al., 2017] problems. In that, the distance variables

have been set as $k = 20$, wherever applicable.

5.4.2 Performance Indicators and Statistical Analysis

The choice of performance indicators is the same as adopted earlier in Chapters 3 and 4. The key details are re-iterated below.

- Hypervolume is used the primary indicator with 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-EMO. Further, for the problems where the scales of different objectives are different, the solutions are normalized in the F-space using the theoretical PF extremes.
- Population mean of the $g(X)$ function is used as the secondary indicator, to provide insights into the convergence levels in the X-space.

Further, in the context of statistical analysis of the performance indicator values,

- when comparing *only two* algorithms, at a time, Wilcoxon ranksum test [Wilcoxon, 1945] is performed on the indicator values reported over multiple/independently seeded runs. In that, the threshold value of 0.05 (95% confidence interval) is used.
- when comparing *more than two* algorithms, at a time, Kruskal-Wallis test [Kruskal and Wallis, 1952] with threshold p-value of 0.05 is used, to infer if their overall differences are statistically insignificant or not. If not, the Wilcoxon test is used for their pairwise comparisons, treating NSGA-III-UIP as reference. Furthermore, the threshold p-value is adjusted using the standard Bonferroni correction [Abdi, 2007], to retain the same overall confidence.

5.4.3 Parameter Settings

In this subsection, the parameters and settings used for: (a) the EMO algorithms; and (b) the UIP operator, have been discussed.

5.4.3.1 EMO Settings

With an aim to obtain a reasonably sized set of RVs using the Das-Dennis method [Das and Dennis, 1998], the gap parameter is set as given in Table 5.3. In that, wherever 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 interior RVs [Deb and Jain, 2014]. For coherence, the population size N is kept the same as the number of RVs corresponding to a particular objective, as given in Table 5.3. Further, the natural variation operators include SBX crossover ($p_c = 0.9$ and $\eta_c = 20$) and polynomial mutation ($p_c = 1/n_{\text{var}}$ and $\eta_m = 20$) for an n_{var} variable problem.

Table 5.3. 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

Further, each EMO algorithm has been run for 31 times, with random seeds. In that, for NSGA-III-UIP, the termination generation t_{term} has been determined on-the-fly through the stabilization tracking algorithm, using $\psi_{\text{term}} \equiv \{3, 50\}$. For all other EMO algorithms, the mean t_{term} determined for NSGA-III-UIP over 31 runs has been used as the t_{term}.

5.4.3.2 UIP Operator Settings

Notably, the UIP operator does not involve any additional parameters, other than the ones that are a part of either IP2 or IP3 operator, for which the specifications have been provided in Chapters 3 and 4, respectively. The only change is that the initial values of $t_{\text{freq}}^{\text{IP2}}$ and $t_{\text{freq}}^{\text{IP3}}$ have been set as 2 instead of 1, which aligns with their respective minimum values. Further, there is an additional constraint on the setting of $\mathcal{P}^{\text{IP2}}$ and $\mathcal{P}^{\text{IP3}}$, given as $\mathcal{P}^{\text{IP2}} + \mathcal{P}^{\text{IP3}} \leq 100\%$. In case this constraint is violated, the total offspring created in a particular generation where both IP2 and IP3 operators are invoked, will exceed N, resulting in additional solution evaluations. Notably, the setting of $\mathcal{P}^{\text{IP2}} = \mathcal{P}^{\text{IP3}} = 50\%$, as used in this thesis, satisfies the above constraint.

5.5 Results and Discussions

This section first presents the assessment 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 UIP operator on multi-objective problems, following which, the assessment of UIP operator on many-objective problems has been presented.

5.5.1 UIP vis-à-vis IP2 operator

In this subsection, the performance comparison of UIP and IP2 operators has been realized through a direct comparison of NSGA-III-UIP and NSGA-III-IP2 on convergence-hard multi-objective problems. Notably, NSGA-III has also been included as reference, to assess if UIP's integration into NSGA-III leads to a deteriorated performance, in any test instance.

Table 5.4. Hypervolume and $g(X)$ based comparison of NSGA-III-UIP with NSGA-III and NSGA-III-IP2, on convergence-hard (ZDT, DTLZ and MaF) problems. The termination generation (t_{term}) 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 better than, equivalent to, or worse than the underlying algorithm, respectively.

	Problem	t_{term}	Hypervolume			$g(X)$		
			NSGA-III	NSGA-III-IP2	NSGA-III-UIP	NSGA-III	NSGA-III-IP2	NSGA-III-UIP
$M=2$	ZDT1	1198	0.681860=	0.681860=	0.681859	1.0007E+00–	1.0004E+00=	1.0004E+00
	ZDT2	1267	0.348794=	0.348794=	0.348794	1.0005E+00–	1.0003E+00=	1.0003E+00
	ZDT3	1007	1.068445=	1.068408=	1.068492	1.0096E+00=	1.0061E+00=	1.0070E+00
	ZDT4	1767	0.681859=	0.681860=	0.681860	1.0007E+00–	1.0004E+00=	1.0002E+00
	ZDT6	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 ⟶		07/14/00	04/16/01	of 21 probs.	06/12/00	03/12/03	of 18 probs.

Note: (—) implies that the concerned problem does not have a $g(X)$ function.

Table 5.4 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of t_{term} generations. In that, t_{term} has been determined on-the-fly for NSGA-III-UIP, and same has been used for NSGA-III and NSGA-III-IP2. From Table 5.4, following can be observed with respect to NSGA-III-UIP versus NSGA-III-IP2.

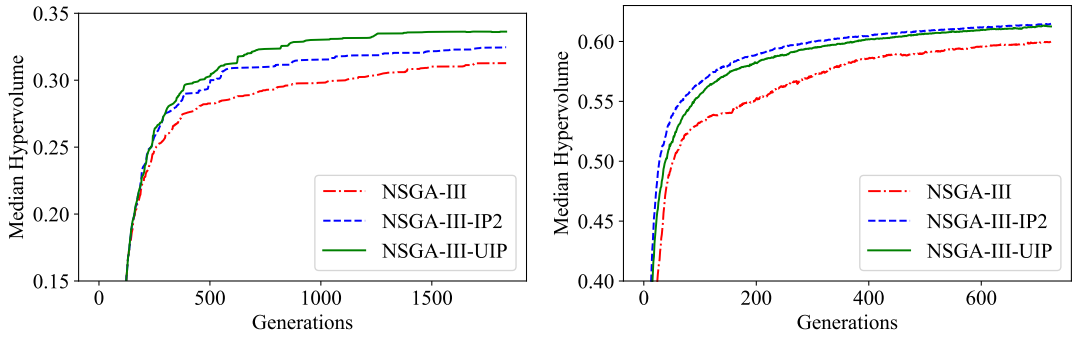
- In the context of hypervolume: NSGA-III-UIP performs either statistically better than or equivalent to NSGA-III-IP2 in 20 out of 21 instances. In that, NSGA-III-UIP performs statistically better than and worse than NSGA-III-IP2 in 4 and 1 instance(s), respectively.
- In the context of $g(X)$ values: NSGA-III-UIP performs either statistically better than or equivalent to NSGA-III-IP2 in 15 out of 18 instances, where the $g(X)$ function was existent/computable (instances where it is non-existent, are marked by '—'). In that, NSGA-III-UIP performs statistically better than and worse than NSGA-III-IP2 in equal number of instances.

Overall, it is fair to infer NSGA-III-UIP offers a better performance in terms of hypervolume and an equivalent performance in terms of $g(X)$ values, than NSGA-III-IP2.

Moreover, from Table 5.4, it can be observed that NSGA-III-UIP performs significantly better than or equivalent to NSGA-III in all instances, in terms of both hypervolume and $g(X)$ values. This clearly suggests that integrating the UIP operator into NSGA-III did not deteriorate its performance in any of the convergence-hard problems considered.

To share more insights into these results, two sample test instances have been chosen, that include: (i) $\tilde{\text{Z}}$DT6, where NSGA-III-UIP offers a better hypervolume; and (ii) MaF12, where NSGA-III-UIP offers a worse hypervolume, than NSGA-III-IP2. Figure 5.2a shows the generation wise median hypervolume plot among the 31 randomly seeded runs of NSGA-III, NSGA-III-IP2 and NSGA-III-UIP. In that, the better hypervolume of NSGA-III-UIP could be attributed to the role of IP3 function in UIP, that helped achieve a diverse set of target solutions for the IP2 function to learn from. This led to a better learning in IP2 function and consequently, led to a better convergence of NSGA-III-UIP, as can be observed through the $g(X)$ values in Table 5.4.

Further, Figure 5.2b shows the generation-wise median hypervolume plot among the 31 randomly seeded runs of NSGA-III, NSGA-III-IP2 and NSGA-III-UIP. Clearly, NSGA-III-UIP offers a slightly worse hypervolume than NSGA-III-IP2, which could also be attributed to the role of IP3 function in UIP. In that, if the population has already achieved a reasonable diversity across the PF in early generations, which is fair to expect in a convergence-hard problem, the scope of



(a) ŽDT6 ($t_{\text{term}} = 1836$) (b) MaF12 ($t_{\text{term}} = 725$)

Figure 5.2. Performance of NSGA-III-UIP vis-à-vis NSGA-III and NSGA-III-IP2 on two sample convergence-hard problems.

diversity-enhancement through the IP3 function would diminish, causing some loss of solution evaluations. This may lead to a worse convergence (as can be observed through $g(X)$ values in Table 5.4), and consequently, a worse hypervolume.

5.5.2 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 multi-objective problems. Notably, NSGA-III has also been included as reference, to assess if UIP's integration into NSGA-III leads to a deteriorated performance, in any test instance.

Table 5.5 reports the median hypervolume and median $g(X)$ values, from among the 31 randomly seeded runs at the end of t_{term} generations. In that, t_{term} has been determined on-the-fly for NSGA-III-UIP, and same has been used for NSGA-III and NSGA-III-IP3. From Table 5.5, following can be observed with respect to NSGA-III-UIP versus NSGA-III-IP3.

- In context of hypervolume: NSGA-III-UIP performs either statistically better than or equivalent to NSGA-III-IP3 in 27 out of 28 instances. In that, NSGA-III-UIP performs statistically better and worse than NSGA-III-IP3 in 6 and 1 instance(s), respectively.

- In context of $g(X)$ values: NSGA-III-UIP performs either statistically better than or equivalent to NSGA-III-IP3 in all 28 out of 28 instances.

Overall, it is fair to infer than NSGA-III-UIP offers a better performance than NSGA-III-IP3, in terms of both hypervolume and $g(X)$ values. Moreover, from Table 5.5, following can be

Table 5.5. 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_{term}) 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 better than, equivalent to, or worse than the underlying algorithm, respectively.

	Problem	t_{term}	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
$M=3$	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 $\longrightarrow$	14/13/01	06/21/01	of 28 probs.	08/13/07	08/20/00	of 28 probs.

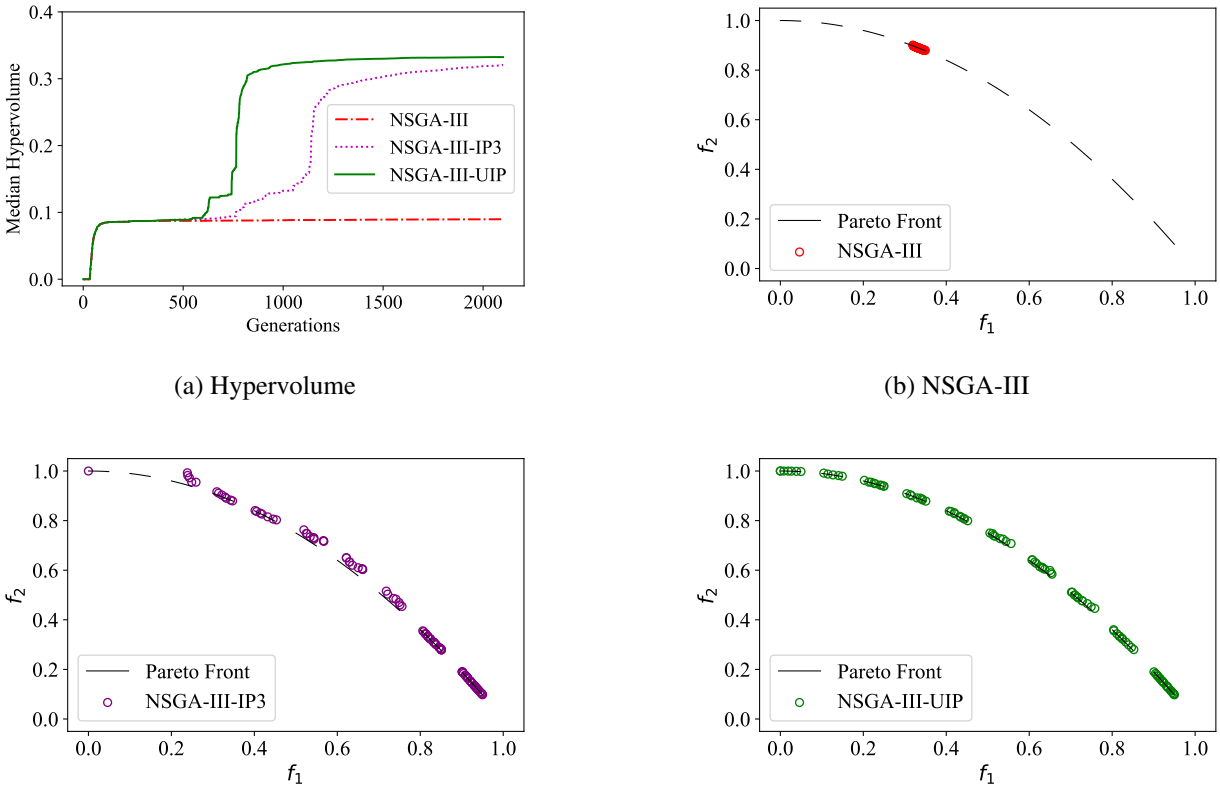
observed with respect to NSGA-III-UIP versus NSGA-III.

- In context of hypervolume: NSGA-III-UIP performs either statistically better than or equiv-

alent to NSGA-III in 27 out of 28 instances. In that, NSGA-III-UIP performs statistically better and worse than NSGA-III in 14 and 1 instance(s), respectively.

- In context of $g(X)$ values: NSGA-III-UIP performs either statistically better than or equivalent to NSGA-III in 21 out of 28 instances. In that, NSGA-III-UIP performs statistically better and worse than NSGA-III in 8 and 7 instances, respectively.

Overall, it is fair to infer that NSGA-III-UIP offers a better performance than NSGA-III in terms of hypervolume, and similar performance to NSGA-III in terms of $g(X)$ values.



(a) Hypervolume

(b) NSGA-III

(c) NSGA-III-IP3

(d) NSGA-III-UIP

Figure 5.3. Generation-wise hypervolume trend and solutions obtained in the respective median runs of NSGA-III, NSGA-III-IP3, and NSGA-III-UIP at $t_{\text{term}} = 2101$ generations, on DASCMOP1.

To share more insights into these results, DASCMOP1 problem has been chosen here for a sample discussion. Figure 5.3a shows the generation-wise median hypervolume plot among the 31 randomly seeded runs of NSGA-III, NSGA-III-IP3 and NSGA-III-UIP. Besides, the solutions obtained at the end of $t_{\text{term}} = 2101$ generations in the respective median runs of NSGA-III, NSGA-III-IP3 and NSGA-III-UIP, are shown in Figures 5.3b, 5.3c and 5.3d, respectively. In

that, NSGA-III-UIP clearly offers the best hypervolume among the three, which is endorsed by the NSGA-III-UIP's best PF-approximation. Figures 5.3c and 5.3d clearly demonstrate NSGA-III-UIP could offer a better diversity as well as convergence than NSGA-III-IP3. This better convergence in DASCMOP1 can also be observed through $g(X)$ values in Table 5.5.

5.5.3 UIP Operator on Many-objective Problems

This subsection aims to assess if the efficacy of the UIP operator, established above on multi-objective problems, can be extended to many-objective problems as well. Notably, since the UIP operator clearly outperformed the IP2 and IP3 operators, they have been excluded from further investigation on many-objective test problems. To this end, Table 5.6 reports the median hypervolume values from among the 31 randomly seeded runs of NSGA-III and NSGA-III-UIP,

Table 5.6. 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_{term}) 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 better than, equivalent to, or worse than the underlying algorithm, respectively.

Problem	$M = 5$			$M = 8$			$M = 10$		
	t_{term}	NSGA-III	NSGA-III-UIP	t_{term}	NSGA-III	NSGA-III-UIP	t_{term}	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			07/07/02			06/09/01	of 16 probs.

at the end of t_{term} generations. In that, t_{term} has been determined on-the-fly for NSGA-III-UIP, and the same has been used for NSGA-III. From Table 5.6, following may be observed.

- NSGA-III-UIP performs statistically better than NSGA-III in 15 out of 48 instances, spread over $M = 5$, 8 and 10.
- NSGA-III-UIP performs statistically better than or equivalent to NSGA-III in 43 out of 48 instances, spread over $M = 5$, 8 and 10.

From the above, it is fair to infer that NSGA-III-UIP offers an overall better performance than NSGA-III on considered many-objective problems, ranging from 5 to 10 objectives. This serves as a proof-of-concept that the proposed UIP operator is scalable in terms of objectives. Hence, the UIP operator is capable of improving the performance of NSGA-III, not only in multi-objective problems, but in many-objective problems as well.

5.6 Results with Other RV-EMO Algorithms

To demonstrate how the proposed UIP operator can improve the search efficacy of other RV-EMO algorithms, the performance of θ-DEA-UIP and MOEA/DD-UIP has been investigated vis-à-vis their respective base variants, on both multi- and many-objective test problems.

5.6.1 Multi-objective Problems

In this subsection, the performance of θ-DEA-UIP and MOEA/DD-UIP has been investigated sequentially on both convergence- and diversity-hard problems.

5.6.1.1 Convergence-hard Problems

Table 5.7 reports the median hypervolume, from among 31 randomly seeded runs at the end of t_{term} generations, for both θ-DEA-UIP and MOEA/DD-UIP. In that, t_{term} has been determined separately for θ-DEA-UIP and MOEA/DD, and the same has been used for their respective base variants. From Table 5.7, following can be observed.

- θ-DEA-UIP performs either statistically better than or equivalent to θ-DEA in 20 out of 21 instances. In that θ-DEA-UIP performs statistically better than and worse than θ-DEA in 5 and 1 instances, respectively.

Table 5.7. Hypervolume based comparison of θ-DEA-UIP and MOEA/DD-UIP with their respect base versions, on convergence-hard (ZDT, DTLZ and MaF) multi-objective problems. The termination generation (t_{term}) has been respectively determined on-the-fly for θ-DEA-UIP and MOEA/DD-UIP. The entries are formatted as $median$ indicator values from 31 runs followed by '–', '=' or '+'. –/=/+ inform that the corresponding UIP variant performs statistically better, equivalent, or worse, respectively.

	Problem	t_{term}	θ-DEA	θ-DEA-UIP	t_{term}	MOEA/DD	MOEA/DD-UIP
$M=2$	ZDT1	1161	0.681280=	0.681282	1155	0.681215=	0.681159
	ZDT2	1237	0.347854=	0.347823	1247	0.347746=	0.347793
	ZDT3	977	1.067375=	1.067561	1005	1.067135=	1.067364
	ZDT4	1744	0.681257=	0.681356	1675	0.681196=	0.681241
	ZDT6	1822	0.313645–	0.334241	1771	0.313384–	0.332646
$M=3$	DTLZ1	1118	1.213503+	0.873703	1136	1.216124=	1.215704
	DTLZ2	963	0.651611=	0.652131	912	0.653244=	0.653748
	DTLZ3	1237	0.611978=	0.085917	981	0.551322+	0.000000
	DTLZ4	1935	0.652918=	0.654282	1226	0.654143–	0.655911
	MaF1	618	0.228233=	0.229284	625	0.229522=	0.228215
	MaF2	515	0.390904=	0.391138	513	0.393007=	0.394059
	MaF3	2123	1.193223=	1.194777	2133	1.191562–	1.194469
	MaF4	1300	0.610459–	0.622088	1295	0.613593–	0.623476
	MaF5	1831	1.228640=	1.228685	1503	1.228484=	1.228528
	MaF7	1166	0.371659=	0.372033	1118	0.371731=	0.371328
	MaF8	1532	0.000331=	0.000172	1608	0.000109=	0.000087
	MaF9	1155	0.610583–	0.614858	1227	0.610000–	0.613124
	MaF10	1019	0.477226–	0.494561	1025	0.508591=	0.499967
	MaF11	1071	1.144277=	1.138350	998	1.142115=	1.142032
	MaF12	743	0.534650–	0.590140	761	0.535199–	0.592456
	MaF13	1021	0.366427=	0.369516	1174	0.501206=	0.524403
	Total (–/=/+) $\longrightarrow$		**05**/15/01			**06**/14/01	of 21 probs.

- MOEA/DD-UIP performs either statistically better than or equivalent to MOEA/DD in 20 out of 21 instances. In that MOEA/DD-UIP performs statistically better than and worse than MOEA/DD in 6 and 1 instances, respectively.

Overall, it is fair to infer that both θ-DEA-UIP and MOEA/DD-UIP offer a better performance in terms of hypervolume than their respective base variants, on convergence-hard multi-objective problems.

5.6.1.2 Diversity-hard Problems

Table 5.8 reports the median hypervolume, from among 31 randomly seeded runs at the end of t_{term} generations, for both θ-DEA-UIP and MOEA/DD-UIP. In that, t_{term} has been determined

Table 5.8. Hypervolume based comparison of θ-DEA-UIP and MOEA/DD-UIP with their respect base versions, on diversity-hard (CIBN, DASCMOP and MW) multi-objective problems. The termination generation (t_{term}) has been respectively determined on-the-fly for θ-DEA-UIP and MOEA/DD-UIP. The entries are formatted as $median$ indicator values from 31 runs followed by '–', '=' or '+'. –/=/+ inform that the corresponding UIP variant performs statistically better, equivalent, or worse, respectively.

	Problem	t_{term}	θ-DEA	θ-DEA-UIP	t_{term}	MOEA/DD	MOEA/DD-UIP
$M=2$	CIBN1	1367	0.335619–	0.508967	1325	0.334598–	0.488089
	CIBN2	811	0.667394–	0.670158	766	0.668834–	0.669776
	CIBN3	1013	0.213996–	0.226054	914	0.214547–	0.226791
	DASCMOP1	1889	0.087565–	0.333138	1944	0.132878–	0.333529
	DASCMOP2	1851	0.415182–	0.665041	1963	0.472963–	0.669614
	DASCMOP3	1836	0.398531–	0.490193	1866	0.408387–	0.483395
	DASCMOP4	2006	0.335362=	0.335602	2183	0.335677=	0.335719
	DASCMOP5	2056	0.671229=	0.671456	2293	0.671290=	0.671308
	DASCMOP6	2536	0.574568–	0.574870	2024	0.575065–	0.575155
	MW1	1022	0.414481=	0.414556	1074	0.413709–	0.414596
	MW2	833	0.482597–	0.489976	891	0.482312–	0.490426
	MW3	1050	0.469747=	0.469479	971	0.469695=	0.469748
	MW5	1632	0.100401–	0.200033	1453	0.195836=	0.200113
	MW6	1212	0.297263=	0.310665	1272	0.296733=	0.309847
	MW7	905	0.365294=	0.365050	890	0.365752=	0.366082
	MW9	986	0.293277–	0.295686	1128	0.293964=	0.294019
	MW10	1032	0.246414=	0.263056	1054	0.261012=	0.289881
	MW11	987	0.265164=	0.208305	1087	0.265863=	0.254258
	MW12	988	0.570107–	0.570289	1085	0.570166–	0.570444
	MW13	990	0.322834–	0.336527	945	0.322475=	0.336561
$M=3$	CIBN4	505	0.895468–	0.908948	480	0.896755=	0.904669
	CIBN5	299	0.619556=	0.619197	305	0.622079=	0.621514
	DASCMOP7	1767	1.014610=	1.013342	1888	1.016219=	1.017115
	DASCMOP8	1751	0.648435=	0.632479	1850	0.651319=	0.652080
	DASCMOP9	1524	0.451661–	0.643800	1511	0.638729–	0.647575
	MW4	756	1.024349=	1.026623	755	1.024167–	1.028343
	MW8	732	0.609678=	0.616218	668	0.615115=	0.624471
	MW14	894	0.156222=	0.156373	960	0.164248=	0.168807
	Total (–/=/+) $\longrightarrow$		**14**/14/00			**12**/16/00	of 28 probs.

separately for θ-DEA-UIP and MOEA/DD, and the same has been used for their respective base variants. From Table 5.8, following can be observed.

- θ-DEA-UIP performs either statistically better than or equivalent to θ-DEA in all 28 out of 28 instances. In that θ-DEA-UIP performs statistically better than θ-DEA in 14 instances, respectively.

- MOEA/DD-UIP performs either statistically better than or equivalent to MOEA/DD in all 28 out of 28 instances. In that MOEA/DD-UIP performs statistically better than MOEA/DD in 12 instances, respectively.

Overall, it is fair to infer that both θ-DEA-UIP and MOEA/DD-UIP offer a better performance in terms of hypervolume than their respective base variants, on diversity-hard multi-objective problems.

5.6.2 Many-objective Problems

In this subsection, the performance of θ-DEA and MOEA/DD has been investigated on many-objective test problems.

Table 5.9. Hypervolume based comparison of θ-DEA-UIP with θ-DEA on many-objective (DTLZ and MaF) problems, with $M = 5, 8$ and 10. The termination generation (t_{term}) has been determined on-the-fly for θ-DEA-UIP. The entries are formatted as $median$ hypervolume values from 31 runs followed by '–', '=' or '+'. –/=/+ inform that θ-DEA-UIP performs statistically better than, equivalent to, or worse than the underlying algorithm, respectively.

Problem	$M = 5$			$M = 8$			$M = 10$		
	t_{term}	θ-DEA	θ-DEA-UIP	t_{term}	θ-DEA	θ-DEA-UIP	t_{term}	θ-DEA	θ-DEA-UIP
DTLZ1	748	2.157168=	2.155684	875	9.981730=	9.979700	833	0.000000=	0.000000
DTLZ2	941	2.121202=	2.117514	905	9.763755+	9.757405	877	17.591542=	17.592135
DTLZ3	765	2.082338=	2.012005	847	8.546232+	7.622917	819	0.000000=	0.000000
DTLZ4	969	2.129442=	2.128684	857	9.731912+	9.707987	763	17.219373=	17.165056
MaF1	844	0.056371=	0.056379	928	0.013510=	0.012873	773	0.001562–	0.001647
MaF2	366	0.988566–	0.995460	454	4.122041–	4.177972	521	7.475374–	7.613498
MaF3	1209	1.977794=	1.866041	612	0.000000=	0.000007	461	0.000000=	0.000000
MaF4	863	0.192197=	0.220878	963	0.032738=	0.045256	1222	0.026055–	0.034063
MaF5	1157	2.487719=	2.487765	1254	9.988717=	9.988717	1292	17.757727=	17.757727
MaF7	1090	0.956168=	0.955427	1022	4.222138–	4.262750	766	6.468072=	6.712384
MaF8	1767	0.000233–	0.000340	1429	0.000517–	0.000570	1317	0.000075=	0.000081
MaF9	2037	0.028336=	0.028005	849	0.000833=	0.001078	786	0.000014=	0.000042
MaF10	1049	0.830748–	0.877855	888	2.709276–	3.293281	962	5.286687–	5.609288
MaF11	1048	2.376759=	2.370576	1105	9.448397=	9.465672	1193	16.919760–	17.053469
MaF12	558	1.671026=	1.647667	431	6.450904=	6.438767	490	11.460954=	11.612789
MaF13	857	0.248616=	0.513728	966	0.269744=	0.297863	1211	0.160839–	0.213371
(–/=/+) $\longrightarrow$		**03**/13/00			**04**/09/03			**06**/10/00	of 16 probs.

5.6.2.1 θ-DEA-UIP

Table 5.9 reports the median hypervolume values from among the 31 randomly seeded runs of θ-DEA-UIP and θ-DEA, at the end of t_{term} generations. In that, t_{term} has been determined on-the-fly for θ-DEA-UIP, and the same has been used for θ-DEA. From Table 5.9, following may be observed.

- θ-DEA-UIP performs statistically better than θ-DEA in 13 out of 48 instances, spread over $M = 5$, 8 and 10.
- θ-DEA-UIP performs statistically better than or equivalent to θ-DEA in 45 out of 48 instances, spread over $M = 5$, 8 and 10.

From the above, it is fair to infer that θ-DEA-UIP offers an overall better performance than θ-DEA on considered many-objective problems, ranging from 5 to 10 objectives.

5.6.2.2 MOEA/DD-UIP

Table 5.10 reports the median hypervolume values from among the 31 randomly seeded runs of MOEA/DD-UIP and MOEA/DD, at the end of t_{term} generations. In that, t_{term} has been determined on-the-fly for MOEA/DD-UIP, and the same has been used for MOEA/DD. From Table 5.10, following may be observed.

- MOEA/DD-UIP performs statistically better than MOEA/DD in 14 out of 48 instances, spread over $M = 5$, 8 and 10.
- MOEA/DD-UIP performs statistically better than or equivalent to MOEA/DD in 44 out of 48 instances, spread over $M = 5$, 8 and 10.

From the above, it is fair to infer that MOEA/DD-UIP offers an overall better performance than MOEA/DD on considered many-objective problems, ranging from 5 to 10 objectives.

5.7 Run-time Analysis of the UIP Operator

For real-world problems, quite often, the time spent on solution evaluations constitutes a dominant fraction of the overall run time for 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. Hence, any methodological intervention that could help ensure

Table 5.10. Hypervolume based comparison of MOEA/DD-UIP with MOEA/DD on many-objective (DTLZ and MaF) problems, with $M = 5$, 8 and 10. The termination generation (t_{term}) has been determined on-the-fly for MOEA/DD-UIP. The entries are formatted as $median$ hypervolume values from 31 runs followed by '–', '=' or '+'. –/=/+ inform that MOEA/DD-UIP performs statistically better than, equivalent to, or worse than the underlying algorithm, respectively.

Problem	$M = 5$			$M = 8$			$M = 10$		
	t_{term}	MOEA/DD	MOEA/DD-UIP	t_{term}	MOEA/DD	MOEA/DD-UIP	t_{term}	MOEA/DD	MOEA/DD-UIP
DTLZ1	883	2.670244=	2.456018	853	6.960960+	5.976893	808	16.139124+	15.307207
DTLZ2	894	2.121843=	2.120594	839	9.758621=	9.755059	834	17.574223=	17.566918
DTLZ3	904	0.260120=	0.066750	803	8.319002=	8.116630	814	9.385090+	6.920623
DTLZ4	960	2.133188=	2.130794	784	9.713384=	9.714599	754	17.296162=	17.248217
MaF1	856	0.055339=	0.056772	839	0.013111=	0.013142	757	0.001787–	0.001837
MaF2	376	0.981635=	0.984138	455	4.039789–	4.096927	535	7.321452–	7.494955
MaF3	1842	2.483576=	2.378006	1504	0.000000=	0.000045	568	0.000000=	0.000237
MaF4	820	0.154150=	0.168207	833	8.957987=	8.748836	975	0.015330=	0.019916
MaF5	1173	2.487694–	2.487737	1167	9.988716–	9.988717	1090	17.757727=	17.757726
MaF7	1126	0.968569+	0.964265	959	4.291200=	4.295078	514	4.935445–	5.351254
MaF8	1853	0.000094–	0.000208	1508	0.000390=	0.000382	1438	0.000063=	0.000061
MaF9	1607	0.026104=	0.027067	745	0.000000=	0.000004	912	0.000022–	0.000071
MaF10	975	0.828694–	0.879025	931	2.599672–	3.326115	938	4.797117–	5.624205
MaF11	1020	2.361815=	2.364974	1073	9.368322=	9.367990	1149	16.694957–	16.906135
MaF12	505	1.637249=	1.622541	462	5.856193=	5.954883	496	10.449224=	10.762628
MaF13	944	0.409027–	0.515826	1132	0.621752=	1.007821	1164	1.268003–	1.918401
(–/=/+) $\longrightarrow$		**04**/11/01			**03**/12/01			**07**/07/02	of 16 probs.

a reasonably good PF-approximation in fewer solution evaluations could have immense utility. However, it cannot be ignored that any such intervention may require both additional solution evaluations and computational time, for the underlying methodology to come into effect. Hence, the total number of solution evaluations (required collectively by the original optimizer and the methodology), and the total time (required for original solution evaluations and methodology implementation), need to be factored in. In this context, it is notable that for a desired or prefixed 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 the total run time may be lower, and
- fewer total solution evaluations may be needed, but the total run time may be higher.

In the context of UIP operator, the results discussed above testify its promise for fewer total solution evaluations for a desired quality of PF-approximation. These include:

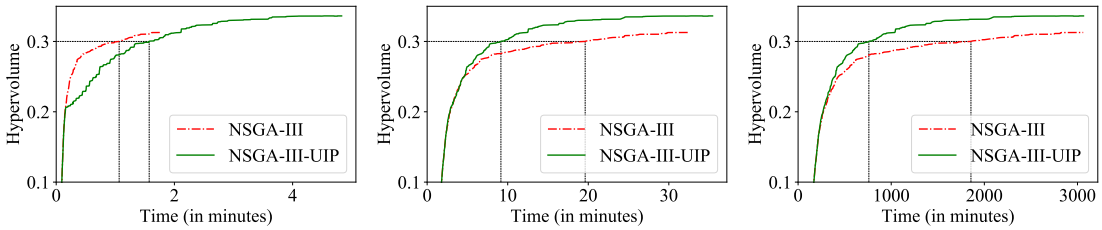
- the fact, that no new solution evaluations are required, and
- the fact, that when integrated with an RV-EMO algorithm, it promises a better or equivalent PF-approximation than the stand-alone RV-EMO algorithm, *at any given generation* (as in Figures 5.2 and 5.3a).

Critically, any specific generation of an RV-EMO-UIP run, where either or both of IP2 and IP3 are invoked, will take more time than any generation of the base RV-EMO (without UIP), which may be attributed to the construction of underlying training-dataset(s) and subsequently 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 Figures 5.2 and 5.3a, *may not necessarily translate to a better PF-approximation in lower total run time*. This sets up the motivation for investigating the UIP operator with regard to its associated run time. Towards that end, a sample analysis focusing on the performance of NSGA-III and NSGA-III-UIP on the $\tilde{\text{Z}}$DT6 problem is presented here, assisted by the following terminology. Let $\mathcal{T}_{\text{SE}}$ denote the time, in *seconds*, required for one solution evaluation. Also, let $\mathcal{T}_{\text{base}}$ and $\mathcal{T}_{\text{UIP}}$ denote the time, in *minutes*, required to complete one algorithmic run of NSGA-III and NSGA-III-UIP, respectively, till t_{term} generations. Under the computational set-up employed and experimental settings highlighted earlier, it turns out that $\mathcal{T}_{\text{SE}} = 1.05\text{e-}04$ (0.105 milliseconds), $\mathcal{T}_{\text{base}} = 1.752$, and $\mathcal{T}_{\text{UIP}} = 4.829$. Notably, $\rho = \mathcal{T}_{\text{UIP}}/\mathcal{T}_{\text{base}} = 2.756$, is significant, as base NSGA-III may lead to a better PF-approximation than NSGA-III-UIP, if allowed more generations with a run time equivalent of $(\mathcal{T}_{\text{UIP}} - \mathcal{T}_{\text{base}})$ minutes.

The above suggests the possibility that *even though a pre-fixed quality of PF approximation may be offered by NSGA-III-UIP in fewer generations or equivalently fewer solution evaluations, than NSGA-III, it may still require higher overall run time*. Figure 5.4a, which represents the median run of $\tilde{\text{Z}}$DT6, testifies that the above possibility has indeed come true. Notably,

- the horizontal axis in Figure 5.4a represents the total run time (instead of the number of generations, as in Figure 5.2a), since the current focus is to evaluate NSGA-III-UIP versus NSGA-III, based on the total run time, and

- the total time required by NSGA-III to complete $t_{\text{term}} = 1836$ generations in significantly lower than the corresponding time required by NSGA-III-UIP.



(a) $\mathcal{T}_{\text{SE}} \approx 0.0001$, $\rho = 2.756$ (b) $\mathcal{T}_{\text{SE}} = 0.01$, $\rho = 1.095$ (c) $\mathcal{T}_{\text{SE}} = 1.0$, $\rho = 1.001$

Figure 5.4. ZDT6: Total run time analysis for NSGA-III versus NSGA-III-UIP. The black horizontal lines mark a pre-fixed quality of PF-approximation to assess relative performance. Here $\rho = \mathcal{T}_{\text{UIP}}/\mathcal{T}_{\text{base}}$ and $t_{\text{term}} = 1836$.

The specific instance above could *mistakenly* lead to the inference that *for a pre-fixed quality of PF-approximation, though NSGA-III-UIP may require fewer generations or equivalently fewer solution evaluations, than NSGA-III, its total run time can never be comparable or lower than that of NSGA-III*. The basis for such a misconception has been countered below, through variation in $\mathcal{T}_{\text{SE}}$. To symbolically emulate real-world scenarios where each solution evaluation may take significantly larger time, two hypothetical values of $\mathcal{T}_{\text{SE}} = 0.01$ and $\mathcal{T}_{\text{SE}} = 1.0$ have been considered. Though, in the case of ZDT6, actual $\mathcal{T}_{\text{SE}} = 1.05\text{e-}04$, the solution evaluations have been inter-spaced by 0.1 and 10.0 seconds, respectively, (using the *sleep* function available in the computational set-up used) to emulate the two scenarios. It may be noted that, as $\mathcal{T}_{\text{SE}}$ values rise from $1.05\text{e-}04$, through 0.01, to 1.0 (seconds), the ρ values fall from 2.756, through 1.095, to 1.001. Interestingly, with $\mathcal{T}_{\text{SE}} = 0.1$ or 1.0, which could be quite common in real-world problems, the time required for a complete algorithmic run of NSGA-III-UIP becomes nearly comparable to that of NSGA-III. These instances also facilitate another insightful revelation. For a pre-fixed quality of PF-approximation, represented by, say a hypervolume of 0.3, as highlighted by horizontal lines in Figure 5.4:

- the solution evaluations required by NSGA-III-UIP are $45,400$ (independent of $\mathcal{T}_{\text{SE}}$ values), while standalone NSGA-III requires $111,200$ solution evaluations,
- the total run time for NSGA-III-UIP is relatively higher than that of NSGA-III, if $\mathcal{T}_{\text{SE}} = 1.05\text{e-}04$, and

- the total run time for NSGA-III-UIP is relatively lower than that of NSGA-III, if $\mathcal{T}_{\mathrm{SE}} = 0.1$ or $\mathcal{T}_{\mathrm{SE}} = 1.0$.

On the basis of the above investigation, it can be inferred that *NSGA-III-UIP promises to offer a pre-fixed quality of PF-approximation, in fewer generations or equivalently fewer solution evaluations, than NSGA-III, while its total run time may be higher or lower than that of NSGA-III depending on the time that each solution evaluation requires*. Clearly, the utility of incorporating the UIP operator may be far more significant in problems where each solution evaluation requires significant time.

Chapter 6

Conclusion and Future Research Directions

In this chapter, the research work presented in this thesis is concluded in Section 6.1, followed by a discussion on some potential future research directions in Section 6.2.

6.1 Conclusion

This thesis has proposed a generic and practicable ML-based framework to assist in the performance enhancement of RV-EMO algorithms. This framework relies on invocations of one of the three operators, including, IP2, IP3, and UIP, that learn from inter- and/or intra-generational solutions, and utilize that learning for the creation of pro-convergence and/or pro-diversity offspring solutions. The degree to which these operators are utilized in a particular generation of an RV-EMO algorithm, and the frequency with which these are invoked across the different generations, have been influenced by the overarching considerations of convergence-diversity balance that is critical for the success of all EMO/RV-EMO algorithms; and also the risk-rewards tradeoff associated with reliance on ML-based operators. Towards generality of the proposed framework, adhoc decisions on critical aspects, including, when to initiate learning, and how frequently to learn, have been avoided. Finally, considering the framework's practicability, evaluation of any additional solution, compared to the base RV-EMO algorithm (without ML assistance), has been successfully avoided. The efficacy of this framework has been tested through its integration with some RV-EMO algorithms, including NSGA-III, θ-DEA and MOEA/DD, on a wide range of test problems with different characteristics, such as convergence-hardness, diversity-hardness, multi-objectives, and many-objectives. In this thesis, while the first two chapters (Chapters 1–2) have laid the foundational concepts, the remaining chapters (Chapters 3–5) have presented the core contributions, which are concluded in the subsequent text.

Chapter 3 presents the IP2 operator for convergence-enhancement, and its associated frame-

work for integration with NSGA-III, leading to NSGA-III-IP2. The IP2 operator is constituted by three modules, including: *training-dataset construction*, *ML training*, and *offspring's advancement*. The first module *maps* the input-archive solutions to the target-archive solutions, *along* the RVs in *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 to *advance* a proportion $\mathcal{P}^{\text{IP2}}$ of the offspring ($\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ in number), originally created using natural variation operators, potentially improving their convergence properties. Notably, the *advanced* offspring replace the underlying offspring (that were advanced), ensuring that no additional solution evaluations are required. Further, in any NSGA-III-IP2 run: (i) the first invocation of the IP2 operator is guided by non-dominance of the population; and (ii) subsequent invocations of the IP2 operator are guided by its performance, implying that such decisions have not been made in an adhoc manner. The efficacy of IP2 operator has been established through a comparison of NSGA-III and NSGA-III-IP2 on several convergence-hard MOPs, where NSGA-III-IP2 performed: (i) better in about 29% instances, and (ii) either better or equivalent in about 95% instances, compared to NSGA-III. This further indicates that despite IP2 being a pro-convergence operator, NSGA-III-IP2 could maintain the required convergence-diversity balance in majority (about 95%) instances.

Chapter 4 presents the IP3 operator for diversity-enhancement, and its associated framework for integration with NSGA-III, leading to NSGA-III-IP3. The IP3 operator is also constituted by three modules, including: *training-datasets construction*, *ML training*, and *offspring creation*. The first module *maps* the solutions associated with neighbouring RVs in *objective-space*, and utilizes their underlying *variable-vectors* to construct M training-datasets, where each dataset facilitates improvement in a particular objective. The second module trains M ML models (one per dataset) to *learn* the search directions in *variable-space*, towards improvement in the corresponding objective. The third module *creates* $\lfloor \mathcal{P}^{\text{IP3}} N \rfloor$ pro-diversity offspring using appropriately selected ML models (one per offspring). In that, half of the offspring ($\lfloor \mathcal{P}^{\text{IP2}} N/2 \rfloor$) are created for improvement in spread, and rest $\lfloor \mathcal{P}^{\text{IP2}} N/2 \rfloor$ for improvement in uniformity of solutions. Notably, in generations where the IP3 operator is invoked, since $\lfloor \mathcal{P}^{\text{IP2}} N \rfloor$ offspring are already created using IP3, only the rest $\lceil (1 - \mathcal{P}^{\text{IP2}}) N \rceil$ offspring are created using natural variation operators, ensuring that no additional solution evaluations are required. Further, in any NSGA-III-IP3 run: (i) first invocation of the IP3 operator is guided by mild stabilization of the

population; and (ii) subsequent invocations of the IP3 operator are guided by its performance, implying that such decisions have not been made in an adhoc manner. The efficacy of IP3 operator has been established through a comparison of NSGA-III and NSGA-III-IP3 on several diversity-hard MOPs, where NSGA-III-IP3 performed: (i) better in about 46% instances, and (ii) either better or equivalent in about 96% instances, compared to NSGA-III. This further indicates that despite IP3 being a pro-diversity operator, NSGA-III-IP3 could maintain the required convergence-diversity balance in about 96% instances.

Notably, in Chapters 3 and 4, $\mathcal{P}^{\text{IP2}} = \mathcal{P}^{\text{IP3}} = 50\%$ has been used. This choice has been reasoned in Chapter 1, in the context of some key considerations, including the convergence-diversity balance and the risk-rewards tradeoff. It has also been highlighted that any setting of $\mathcal{P}^{\text{IP2}} > 50\%$ or $\mathcal{P}^{\text{IP3}} > 50\%$ may hamper the management of these considerations. The same could also be observed empirically in the sample investigations presented at the end of Chapters 3 and 4 for the IP2 and IP3 operators, respectively.

Chapter 5 presents the UIP operator for convergence- and diversity-enhancement, and its associated framework for integration with all three RV-EMO algorithms, leading to NSGA-III-UIP, θ-DEA-UIP and MOEA/DD-UIP. The UIP operator relies on independent invocations of the IP2 and IP3 operators for creation of both pro-convergence and pro-diversity offspring solutions. In any generation of RV-EMO-UIP, either or both of IP2 and IP3 operators maybe invoked. In any case, additional solution evaluations are not required, since each of IP2 and IP3 avoid these, individually. Further, the same criteria for (initial and subsequent) invocations of IP2 and IP3 operators have been retained, as in Chapters 3 and 4, respectively. This endorses, that adhoc decisions pertaining to invocations of UIP's constituents were successfully avoided. 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 and MOEA/DD. Notably, NSGA-III-UIP performed better than: (i) NSGA-III-IP2 in about 19% convergence-hard MOPs, and (ii) NSGA-III-IP3 in about 21% diversity-hard MOPs. This endorses that 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 possible otherwise. In general, the UIP variants of the three RV-EMO algorithms collectively performed: (i) better in about 34% instances, and (ii) either better or equivalent in about 95% instances, compared to their respective base variants on multi- and many-objective problems.

Although RV-EMO-UIP offers a better or equivalent PF-approximation compared to the base

RV-EMO algorithm, in the majority of test instances, it has been highlighted that the integration of the UIP operator imposes additional run time complexity. In that, the invocations of IP2 and/or IP3 operators may amount to a higher total run time of RV-EMO-UIP compared to the base RV-EMO, to meet a particular level of PF-approximation, if the solution evaluations are very fast. However, it has been established that the additional run time required to execute the invocations of IP2 and IP3 operators would tend to be negligible in real-world problems, where each solution evaluation may consume a significant amount of time. In a nutshell, the cited tradeoff between the extra total run time required by the UIP operator and the savings in costly solution evaluations, is likely to be in favour of the UIP operator in real-world problems.

A notable but unexplored aspect of the proposed operators is their ability to be integrated with the EMO algorithms that do not utilize an RV-based architecture, such as the dominance-based and indicator-based algorithms. In such cases, the proposed operators can still use a set of RVs to facilitate the mapping of solutions, while the underlying EMO algorithm can run without them. Even though this integration is theoretically feasible, the performance of these operators would depend on several factors, such as: (a) the availability of solutions in the neighbourhood of RVs for constructing the training-dataset(s), and (b) the probability of survival if a good offspring is created at any given RV. It would be intriguing to explore the above, as an extension to the proposed work.

As intended, this thesis has broadened the scope of *online innovization* by factoring both convergence and diversity, without the need to specify the relationship structures, a priori, and without incurring additional solution evaluations compared to the base RV-EMO algorithm. The author believes that this thesis has only scratched the surface of what may turn out to be one of the dominant research themes of ML-assisted evolutionary optimization. Some foreseeable research directions have been highlighted below.

6.2 Potential Future Research Directions

At a structural level, the focus in this thesis pertains to the *search* phase of RV-EMO algorithms, where solutions are iteratively *evolved* towards a good PF-approximation. Within this, the ML interventions have been restricted to the offspring produced by the natural variation operators. Clearly, even within the premise of RV-EMO algorithms, the utility of ML-based interventions could also be explored with regard to the other phases, including, *initialization* of solutions;

search phase with focus on *selection* operators; algorithm *termination*; and *decision making*. Towards a wider gamut of ML-assisted evolutionary optimization, similar interventions in the context of EMO algorithms that do not rely on the use of reference vectors, may also be explored. The utility of ML-based approaches 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.

- *ML-assisted Initialization:* conventionally, EMO algorithms are initialized with a population, constituted by a pre-defined number (N) of solutions. These solutions are usually created randomly or by utilizing some sampling methods such as Latin hypercube, in the variable-space. Even though these N solutions may represent a diverse-set in the variable-space, their representation in the objective-space remains unknown until they are evaluated. Intuitively, it may be useful if the initial population offers a good diversity in the objective-space, which may eventually lead to a better PF-approximation in fewer solution evaluations. Towards this, an ML method may be utilized to sample the initial population in such a manner that it also effects better initial diversity in the objective-space, eventually facilitating a better PF-approximation.

- *ML-assisted Termination:* this thesis has utilized an earlier proposed stabilization tracking algorithm [Saxena and Kapoor, 2019] to detect: (i) mild population stabilization, to *time* the first invocation of IP3 operator, and (ii) strict population stabilization, to terminate an RV-EMO-IP2/IP3/UIP run. Notably, this algorithm requires a set of two parameters that govern the degree of stabilization to be detected. While these parameters could be intuitively set, it would be better to have a stabilization tracking algorithm that does not require a priori fixation of parameters. Towards this, an ML method may be utilized to: (i) detect the performance trend of the underlying EMO algorithm 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 an algorithm's termination.

- *ML-assisted Transfer Learning for Multi-criterion Decision Making:* most EMO studies have focused on finding a set of Pareto-optimal solutions, and left the multi-criterion decision-making (MCDM) task of picking a single preferred solution, to the decision maker (DM). However, many existing MCDM methods can be combined with EMO algorithms,

such that the DM's preferences can influence *selection*, eventually leading to a single, preferred optimal solution at the end of an EMO 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 variable and objective combinations that were chosen. Such a model can then be applied to pick the preferred solution from a fresh set of optimal solutions.

- *ML-assisted Combinatorial Optimization:* combinatorial optimization problems are considered as one of the most challenging class of optimization problems, owing to the discrete nature of variables, and large-dimensional search-space. This explains why the applications of EMO algorithms on this class of problems, are rather sparse in literature. It would be worth investigating if the innovized progress operators introduced in this thesis can be extended to such problems, and the degree to which the search efficacy of EMO algorithms could be enhanced.

The author hopes that the innovized progress operators proposed in this thesis, the results presented, and some potent research directions cited, shall prompt more ML interventions towards enhancing the efficacy and scope of applications of EMO algorithms.

Bibliography

Abdi, H. Bonferroni and sidak corrections for multiple comparisons. *Encyclopedia of measurement and statistics*, pages 103–107, 2007. URL https://ci.nii.ac.jp/naid/20001381227/en/.

Anand, A., Suganthan, P., and Deb, K. A novel fuzzy and multiobjective evolutionary algorithm based gene assignment for clustering short time series expression data. In *2007 IEEE Congress on Evolutionary Computation*, pages 297–304, 2007. doi: 10.1109/CEC.2007.4424485.

Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL https://doi.org/10.1145/361002.361007.

Bertsekas, D. P., Nedi, A., and Ozdaglar, A. E. *Convex analysis and optimization*. Athena Scientific, 2003.

Bhattacharjee, K. S., Isaacs, A., and Ray, T. Multi-objective optimization using an evolutionary algorithm embedded with multiple spatially distributed surrogates. In *Multi-Objective Optimization*, pages 135–155. World Scientific, 2017. doi: 10.1142/9789813148239_0005.

Bora, T. C., Mariani, V. C., and dos Santos Coelho, L. Multi-objective optimization of the environmental-economic dispatch with reinforcement learning based on non-dominated sorting genetic algorithm. *Applied Thermal Engineering*, 146:688–700, 2019. ISSN 1359-4311. doi: https://doi.org/10.1016/j.applthermaleng.2018.10.020. URL https://www.sciencedirect.com/science/article/pii/S1359431118349317.

Chen, Y., Zhang, Y., and 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*, volume 4029. Springer, Berlin, Heidelberg, 2006.

Cheng, R., Jin, Y., Narukawa, K., and Sendhoff, B. A multiobjective evolutionary algorithm using gaussian process-based inverse modeling. *IEEE Transactions on Evolutionary Computation*, 19(6):838–856, 2015. doi: 10.1109/TEVC.2015.2395073.

Cheng, R., Jin, Y., Olhofer, M., and Sendhoff, B. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20 (5):773–791, 2016. doi: 10.1109/TEVC.2016.2519378.

Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., and Yao, X. A benchmark test suite for evolutionary many-objective optimization. *Complex & Intelligent Systems*, 3:67–81, 2017. doi: 10.1007/s40747-017-0039-7.

Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., and Sindhya, K. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):129–142, 2018.

Coello, C. A. C. and Lamont, G. B. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, 2004.

Coello Coello, C. A., González Brambila, S., Figueroa Gamboa, J., Castillo Tapia, M. G., and Hernández Gómez, R. Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, 6:221–236, 2020. doi: 10.1007/s40747-019-0113-4.

Cover, T. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14 (1):50–55, 1968. doi: 10.1109/TIT.1968.1054098.

Dai, C., Wang, Y., Ye, M., Xue, X., and Liu, H. An orthogonal evolutionary algorithm with learning automata for multiobjective optimization. *IEEE Transactions on Cybernetics*, 46(12): 3306–3319, 2016. doi: 10.1109/TCYB.2015.2503433.

Das, I. and Dennis, J. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal of Optimization*, 8(3): 631–657, 1998.

Deb, K. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.

Deb, K. Unveiling innovative design principles by means of multiple conflicting objectives. *Engineering Optimization*, 35(5):445–470, 2003. doi: 10.1080/0305215031000151256.

Deb, K. and Datta, R. Hybrid evolutionary multi-objective optimization and analysis of machining operations. *Engineering Optimization*, 44(6):685–706, 2012. doi: 10.1080/0305215X.2011.604316.

Deb, K. and Jain, H. An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

Deb, K. and Myburgh, C. A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables. *European Journal of Operational Research*, 261(2):460–474, 2017.

Deb, K. and Srinivasan, A. Innovization: Innovating design principles through optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 1629–1636, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595931864. doi: 10.1145/1143997.1144266.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pages 105–145, London, 2005. Springer London. ISBN 978-1-84628-137-2. doi: 10.1007/1-84628-137-7_6. URL https://doi.org/10.1007/1-84628-137-7_6.

Deb, K., Hussein, R., Roy, P., and Toscano, G. A taxonomy for metamodeling frameworks for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 23(1):104–116, 2019.

Du, Y., qing Li, J., Luo, C., and lei Meng, L. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations. *Swarm and Evolutionary Computation*, 62:100861, 2021. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2021.100861.

Dutta, S. and 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, pages 435–451, Cham, 2020. Springer International Publishing. ISBN 978-3-030-39831-6. doi: 10.1007/978-3-030-39831-6_29. URL https://doi.org/10.1007/978-3-030-39831-6_29.

El-Beltagy, M. A., Nair, P. B., and 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)*, pages 196–203. San Mateo, CA: Morgan Kaufman, 1999.

Emmerich, M., Giannakoglou, K. C., and Naujoks, B. Single and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

Fan, Z., Li, W., Cai, X., Li, H., Wei, C., Zhang, Q., Deb, K., and Goodman, E. Difficulty Adjustable and Scalable Constrained Multiobjective Test Problem Toolkit. *Evolutionary Computation*, 28(3):339–378, 09 2020. ISSN 1063-6560. doi: 10.1162/evco_a_00259. URL https://doi.org/10.1162/evco_a_00259.

Garg, K., Mukherjee, A., Mittal, S., Saxena, D. K., and Deb, K. A generic and computationally efficient automated innovization method for power-law design rules. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, page 161–162, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371278. URL https://doi.org/10.1145/3377929.3390022.

Gaur, A. and 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, page 75–76, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450343237. doi: 10.1145/2908961.2909019.

Gaur, A. and 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)*, pages 2177–2184, 2017. doi: 10.1109/CEC.2017.7969568.

Ge, H., Zhao, M., Sun, L., Wang, Z., Tan, G., Zhang, Q., and Chen, C. L. P. A many-objective evolutionary algorithm with two interacting processes: Cascade clustering and reference point

incremental learning. *IEEE Transactions on Evolutionary Computation*, 23(4):572–586, 2019. doi: 10.1109/TEVC.2018.2874465.

Goel, T. and Deb, K. Hybrid methods for multi-objective evolutionary algorithms. In *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, pages 188–192, 2002.

Goldberg, D. E. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

Gonçalves, R. A., Almeida, C. P., and Pozo, A. Upper confidence bound (ucb) algorithms for adaptive operator selection in moea/d. In Gaspar-Cunha, A., Henggeler Antunes, C., and Coello, C. C., editors, *Evolutionary Multi-Criterion Optimization*, pages 411–425, Cham, 2015. Springer International Publishing. ISBN 978-3-319-15934-8.

H. Seada, M. A. M. and Deb, K. Towards a better balance of diversity and convergence in NSGA-III: First results. In *Proceedings of the Nineth Inter- national Conference on Evolutionary Multi-Criterion Optimization (EMO-2017), (LNCS 10173)*, pages 545–559. Springer. (Munster, Germany), 2017.

He, C., Huang, S., Cheng, R., Tan, K. C., and Jin, Y. Evolutionary multiobjective optimization driven by generative adversarial networks (GANs). *IEEE Transactions on Cybernetics*, 51(6): 3129–3142, 2021. doi: 10.1109/TCYB.2020.2985081.

Holland, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: MIT Press, 1975.

Huang, W. and Li, H. On the differential evolution schemes in moea/d. In *2010 Sixth International Conference on Natural Computation*, volume 6, pages 2788–2792, 2010. doi: 10.1109/ICNC.2010.5583335.

Hussein, R., Roy, P., and Deb, K. Switching between metamodeling frameworks for efficient multi-objective optimization. In *IEEE Symposium Series on Computational Intelligence (SSCI-2018)*, pages 1–8. Piscatway, NJ: IEEE Press, 2018.

Inapakurthi, R. K. and Mitra, K. Optimal surrogate building using SVR for an industrial grinding process. *Materials and Manufacturing Processes*, 0(0):1–7, 2022. doi: 10.1080/10426914.2022.2039699. URL `https://doi.org/10.1080/10426914.2022.2039699`.

Ishibuchi, H. and Murata, T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(3):392–403, 1998. doi: 10.1109/5326.704576.

Ishibuchi, H. and Narukawa, K. Some issues on the implementation of local search in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004), (LNCS 3102)*, pages 1246–1258, 2004.

Ishibuchi, H., Sakane, Y., Tsukamoto, N., and Nojima, Y. Adaptation of scalarizing functions in moea/d: An adaptive scalarizing function-based multiobjective evolutionary algorithm. In Ehrgott, M., Fonseca, C. M., Gandibleux, X., Hao, J.-K., and Sevaux, M., editors, *Evolutionary Multi-Criterion Optimization*, pages 438–452, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-01020-0.

Ishibuchi, H., Sakane, Y., Tsukamoto, N., and Nojima, Y. Simultaneous use of different scalarizing functions in moea/d. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 519–526, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300728. doi: 10.1145/1830483.1830577. URL https://doi.org/10.1145/1830483.1830577.

Ishibuchi, H., Setoguchi, Y., Masuda, H., and Nojima, Y. Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190, 2017. doi: 10.1109/TEVC.2016.2587749.

Ishibuchi, H., Imada, R., Setoguchi, Y., and Nojima, Y. How to specify a reference point in hypervolume calculation for fair performance comparison. *Evol. Comput.*, 26(3):411–440, Sept. 2018. ISSN 1063-6560. doi: 10.1162/evco_a_00226. URL https://doi.org/10.1162/evco_a_00226.

Jain, H. and Deb, K. An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2014.

Jaskiewicz, A. Genetic local search for multiple objective combinatorial optimization. *Euoropean Journal of Operational Research*, 371(1):50–71, 2002.

Jones, D. R. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

Koçer, H. G. and Uymaz, S. A. A novel local search method for LSGO with golden ratio and dynamic search step. *Soft Computing*, 25:2115–2130, 2021. doi: 10.1007/s00500-020-05284-x.

Kruskal, W. H. and Wallis, W. A. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. ISSN 01621459. URL `http://www.jstor.org/stable/2280779`.

Kumar, R. and Singh, P. K. Pareto evolutionary algorithm hybridized with local search for biobjective TSP. In *Hybrid Evolutionary Algorithms*, pages 361–398, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-73297-6. doi: 10.1007/978-3-540-73297-6_14. URL `https://doi.org/10.1007/978-3-540-73297-6_14`.

Land, M. W. S. and Belew, R. K. *Evolutionary Algorithms with Local Search for Combinatorial Optimization*. PhD thesis, 1998. AAI9914083.

Lara, A., Sanchez, G., Coello, C. A. C., and Schütze, O. HCS: A new local search strategy for memetic multi-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):112–132, 2010.

Li, F., Gao, L., Shen, W., Cai, X., and Huang, S. A surrogate-assisted offspring generation method for expensive multi-objective optimization problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020. doi: 10.1109/CEC48606.2020.9185691.

Li, H. and Zhang, Q. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

Li, K., Fialho, A., Kwong, S., and Zhang, Q. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014a. doi: 10.1109/TEVC.2013.2239648.

Li, K., Zhang, Q., Kwong, S., Li, M., and Wang, R. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 18(6): 909–923, 2014b. doi: 10.1109/TEVC.2013.2293776.

Li, K., Zhang, Q., Kwong, S., Li, M., and Wang, R. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 18(6): 909–923, 2014c. doi: 10.1109/TEVC.2013.2293776.

Li, K., Deb, K., Zhang, Q., and Kwong, S. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716, 2015a. doi: 10.1109/TEVC.2014.2373386.

Li, K., Kwong, S., Zhang, Q., and Deb, K. Interrelationship-based selection for decomposition multiobjective optimization. *IEEE Transactions on Cybernetics*, 45(10):2076–2088, 2015b. doi: 10.1109/TCYB.2014.2365354.

Li, L., Chen, H., and et al., C. L. A robust hybrid approach based on estimation of distribution algorithm and support vector machine for hunting candidate disease genes. *The Scientific World Journal*, 2013(393570):7, 2013. doi: 10.1155/2013/393570.

Lian, Y. and Liou, M.-S. Multiobjective optimization using coupled response surface model and evolutionary algorithm. *AIAA Journal*, 43(6), 2005.

Lima, C., Pelikan, M., Sastry, K., Butz, M., Goldberg, D., and 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.*, volume 4193 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2006.

Lima, C., Pelikan, M., Lobo, F., and 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.*, volume 5752 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2009.

Lin, J., Liu, H., and 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)*, pages 82–86, 2017.

Louppe, G. *Understanding Random Forests: From Theory to Practice*. ArXiV, 2015. URL `https://arxiv.org/abs/1407.7502`.

Ma, H., Wei, H., Tian, Y., Cheng, R., and Zhang, X. A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints. *Information Sciences*, 560:68–91, 2021. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2021.01.029. URL https://www.sciencedirect.com/science/article/pii/S0020025521000566.

Ma, X., Yu, Y., Li, X., Qi, Y., and Zhu, Z. A survey of weight vector adjustment methods for decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 24(4):634–649, 2020. doi: 10.1109/TEVC.2020.2978158.

Ma, Z. and Wang, Y. Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons. *IEEE Transactions on Evolutionary Computation*, 23(6): 972–986, 2019. doi: 10.1109/TEVC.2019.2896967.

Mallipeddi, R. and Lee, M. Surrogate model assisted ensemble differential evolution algorithm. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, 2012. doi: 10.1109/CEC.2012.6256479.

Martí, L., García, J., Berlanga, A., and 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, pages 689–696, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581309. doi: 10.1145/1389095.1389230. URL https://doi.org/10.1145/1389095.1389230.

Martins, M. S. R., Yafrani, M. E., Delgado, M., Lüders, R., Santana, R., Siqueira, H. V., Akcay, H. G., and Ahiod, B. Analysis of bayesian network learning techniques for a hybrid multi-objective bayesian estimation of distribution algorithm: a case study on MNK landscape. *Journal of Heuristics*, 27:549–573, 2021. doi: 10.1007/s10732-021-09469-x.

Masood, A., Mei, Y., Chen, G., and Zhang, M. A pso-based reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling. In Wagner, M., Li, X., and Hendtlass, T., editors, *Artificial Life and Computational Intelligence*, pages 326–338, Cham, 2017. Springer International Publishing. ISBN 978-3-319-51691-2.

Miettinen, K. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1999.

Mittal, S., Saxena, D. K., and Deb, K. A unified automated *Innovization* framework using threshold-based clustering. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020. doi: 10.1109/CEC48606.2020.9185879.

Mittal, S., Saxena, D. K., Deb, K., and Goodman, E. D. Enhanced *Innovized*‘ progress operator for evolutionary multi-and many-objective optimization. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2021a. doi: 10.1109/TEVC.2021.3131952.

Mittal, S., Saxena, D. K., Deb, K., and Goodman, E. D. A learning-based *innovized* progress operator for faster convergence in evolutionary multi-objective optimization. *ACM Trans. Evol. Learn. Optim.*, 2(1), nov 2021b. ISSN 2688-299X. doi: 10.1145/3474059. URL `https://doi.org/10.1145/3474059`.

Miuhlenbein, H. and 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*, pages 178–187. London, UK, 1996.

Mullur, A. A. and Messac, A. Metamodeling using extended radial basis functions: A comparative approach. *Engineering with Computers*, 21(203), 2006. doi: https://doi.org/10.1007/s00366-005-0005-7.

Murata, T., Nozawa, H., Tsujimura, Y., Gen, M., and 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)*, pages 663–668, 2002.

Myburgh, C. and Deb, K. Derived heuristics-based consistent optimization of material flow in a gold processing plant. *Engineering Optimization*, 50(1):1–18, 2018. doi: 10.1080/0305215X.2017.1296436.

Ng, A., Deb, K., and Dudas, C. Simulation-based innovization for production systems improvement: An industrial case study. In *The International 3rd Swedish Production Symposium*, page 278–286, 2009.

Padhye, N., Deb, K., and Mittal, P. Boundary handling approaches in particle swarm optimization. In Bansal, J., Singh, P., Deep, K., Pant, M., and Nagar, A., editors, *Proceedings*

of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012). Advances in Intelligent Systems and Computing, volume 201, pages 287–298. Springer, India, 2013.

Pelican, M., Goldberg, D. E., and Cantu-Paz, E. BOA: The bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Conference, (GECCO-1999)*, pages 525–532, 1999.

Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, pages 525–532, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606114.

Pelikan, M., Goldberg, D., and Lobo, F. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002. doi: 10.1023/A:1013500812258.

Rachmawati, L. and Srinivasan, D. Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front. *IEEE Transactions on Evolutionary Computation*, 13(4):810–824, 2009.

Ren, Q., Luo, F., Ding, W., and 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)*, pages 176–181, 2019. doi: 10.1109/SSCI44817.2019.9002847.

Saini, N., Kumar, S., Saha, S., and Bhattacharyya, P. Scientific document summarization using citation context and multi-objective optimization. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4290–4295, 2021. doi: 10.1109/ICPR48806.2021.9412201.

Sato, H. Inverted pbi in moea/d and its impact on the search performance on multi and many-objective optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, page 645–652, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326629. doi: 10.1145/2576768.2598297. URL https://doi.org/10.1145/2576768.2598297.

Saxena, D. K. and 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., and Reed, P., editors, *Evolutionary Multi-Criterion Optimization*, pages 191–202, Cham, 2019. Springer International Publishing.

Seada, H., Abouhawwash, M., and Deb, K. Multiphase balance of diversity and convergence in multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(3):503–513, 2019. doi: 10.1109/TEVC.2018.2871362.

Shim, V. A., Tan, K. C., and Tan, K. K. A hybrid adaptive evolutionary algorithm in the domination-based and decomposition-based frameworks of multi-objective optimization. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, 2012. doi: 10.1109/CEC.2012.6256485.

Sindhya, K., Deb, K., and Miettinen, K. A local search based evolutionary multi-objective optimization technique for fast and accurate convergence. In *Proceedings of the Parallel Problem Solving From Nature (PPSN-2008)*. Berlin, Germany: Springer-Verlag, 2008.

Sinha, A., Bedi, S., and Deb, K. Bilevel optimization based on kriging approximations of lower level optimal value function. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018. doi: 10.1109/CEC.2018.8477763.

Stewart, T., Bandte, O., Braun, H., Chakraborti, N., Ehrgott, M., Göbelt, M., Jin, Y., Nakayama, H., Poles, S., and Di Stefano, D. Real-world applications of multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 285–327, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-88908-3. doi: 10.1007/978-3-540-88908-3_11.

Takagi, T., Takadama, K., and Sato, H. *Incremental Lattice Design of Weight Vector Set*, page 1486–1494. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450371278. URL https://doi.org/10.1145/3377929.3398082.

Tian, Y., He, C., Cheng, R., and Zhang, X. A multistage evolutionary algorithm for better diversity preservation in multiobjective optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(9):5880–5894, 2021. doi: 10.1109/TSMC.2019.2956288.

Trivedi, A., Srinivasan, D., Sanyal, K., and Ghosh, A. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3): 440–462, 2017. doi: 10.1109/TEVC.2016.2608507.

Wang, H. and Jin, Y. A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. *IEEE transactions on cybernetics*, 50(2):536–549, 2020.

Wang, L., Zhang, Q., Zhou, A., Gong, M., and Jiao, L. Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 20(3):475–480, 2016a. doi: 10.1109/TEVC.2015.2457616.

Wang, R., Zhang, Q., and Zhang, T. Decomposition-based algorithms using pareto adaptive scalarizing methods. *IEEE Transactions on Evolutionary Computation*, 20(6):821–837, 2016b. doi: 10.1109/TEVC.2016.2521175.

Wang, R., Zhou, Z., Ishibuchi, H., Liao, T., and Zhang, T. Localized weighted sum method for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 22(1):3–18, 2018. doi: 10.1109/TEVC.2016.2611642.

Wang, R., Dong, N.-J., Gong, D.-W., Zhou, Z.-B., Cheng, S., Wu, G.-H., and Wang, L. PCA-assisted reproduction for continuous multi-objective optimization with complicated pareto optimal set. *Swarm and Evolutionary Computation*, 60:100795, 2021. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2020.100795.

Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987. URL `http://www.jstor.org/stable/3001968`.

Wu, M., Kwong, S., Jia, Y., Li, K., and Zhang, Q. Adaptive weights generation for decomposition-based multi-objective optimization using gaussian process regression. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 641–648, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349208. doi: 10.1145/3071178.3071339. URL `https://doi.org/10.1145/3071178.3071339`.

Wu, M., Li, K., Kwong, S., Zhang, Q., and Zhang, J. Learning to decompose: A paradigm for decomposition-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(3):376–390, 2019. doi: 10.1109/TEVC.2018.2865931.

Xu, Q., Zhang, C., and Zhang, L. A fast elitism gaussian estimation of distribution algorithm and application for pid optimization. *The Scientific World Journal*, 2014(597278):14, 2014. doi: 10.1155/2014/597278.

Yang, S., Jiang, S., and Jiang, Y. Improving the multiobjective evolutionary algorithm based on decomposition with new penalty schemes. *Soft Comput*, 21:4677–4691, 2017. doi: 10.1007/s00500-016-2076-3. URL https://doi.org/10.1007/s00500-016-2076-3.

Yuan, Y., Xu, H., Wang, B., and Yao, X. A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):16–37, 2016. doi: 10.1109/TEVC.2015.2420112.

Zhang, Q. and Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi: 10.1109/TEVC.2007.892759.

Zhang, Q., Zhou, A., and Jin, Y. Rm-meda: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation*, 12(1):41–63, 2008. doi: 10.1109/TEVC.2007.894202.

Zhao, H. and Zhang, C. An online-learning-based evolutionary many-objective algorithm. *Information Sciences*, 509:1–21, 2020. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.08.069. URL https://www.sciencedirect.com/science/article/pii/S0020025519308187.

Zhou, A., Zhang, Q., Jin, Y., Tsang, E., and Okabe, T. A model-based evolutionary algorithm for bi-objective optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2568–2575 Vol. 3, 2005. doi: 10.1109/CEC.2005.1555016.

Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., and 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*, pages 892–899, 2006. doi: 10.1109/CEC.2006.1688406.

Zhou, Z., Wang, Z., Pang, T., Wei, J., and 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)*, pages 153–160, 2021. doi: 10.1109/CEC45853.2021.9504689.

Zitzler, E., Deb, K., and Thiele, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000. doi: 10.1162/106365600568202. URL https://doi.org/10.1162/106365600568202.

List of Publications

Articles in Pipeline

1. Mittal, S., Saxena, D. K., Deb, K., and Goodman, E. D. A Unified *Innovized* Progress Operator for Evolutionary Multi- and Many-objective Optimization.

Published Journal Articles

2. Mittal, S., Saxena, D. K., Deb, K., and Goodman, E. D. A Learning-based *Innovized* Progress Operator for Faster Convergence in Evolutionary Multi-objective Optimization, *ACM Trans. Evol. Learn. Optim.*, vol. 2, no. 1, pp. 1–29, 2022.

3. Mittal, S., Saxena, D. K., Deb, K., and Goodman, E. D. Enhanced *Innovized* Progress Operator for Evolutionary Multi- and Many-objective Optimization, *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021. [Issue yet to be assigned]

Presentations and Proceedings in International Conferences

4. Mittal, S., Saxena, D. K. and Deb, K. Learning-based multi-objective optimization through ANN-assisted online Innovization, In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, Association for Computing Machinery, New York, NY, USA, 171–172.

5. Garg, K., Mukherjee, A., Mittal, S., Saxena, D. K. and Deb, K. A generic and computationally efficient automated innovization method for power-law design rules, In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, Association for Computing Machinery, New York, NY, USA, 161–162.

6. Mittal, S., Saxena, D. K. and Deb, K. A Unified Automated Innovization Framework Using Threshold-based Clustering" *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1-8.

7. Deb, K., Mittal, S., Saxena, D. K. and Goodman, E. D. Embedding a Repair Operator in Evolutionary Single and Multi-objective Algorithms - An Exploitation-Exploration Perspective, In: *Ishibuchi H. et al. (eds) Evolutionary Multi-Criterion Optimization. EMO 2021*, Lecture Notes in Computer Science, vol 12654. Springer, Cham.