En route to automated extraction and transfer of knowledge in multitask optimization : an evolutionary perspective

Bali, Kavitesh Kumar

2021

Bali, K. K. (2021). En route to automated extraction and transfer of knowledge in multitask optimization : an evolutionary perspective. Doctoral thesis, Nanyang Technological University, Singapore. <https://hdl.handle.net/10356/152658>

<https://hdl.handle.net/10356/152658>

<https://doi.org/10.32657/10356/152658>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 22:19:17 SGT

En Route to Automated Extraction and Transfer of Knowledge in Multitask Optimization: An Evolutionary Perspective

Bali Kavitesh Kumar

School of Computer Science & Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the intellectual content of this thesis is the product of my original research work and has not been submitted for a higher degree to any other University or Institution.

18 Dec. 2020

.....

Date

.....

Bali Kavitesh Kumar

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

29 Dec. 2020

.....

Date

.....

Prof. Ong Yew Soon

Authorship Attribution Statement

This thesis contains material from 2 papers published in the following peer-reviewed journals in which I am listed as an author.

Chapters 3, 4 and part of Chapter 6 are published as “Bali, K.K., Ong, Y.S., Gupta, A. and Tan, P.S., 2019. Multifactorial evolutionary algorithm with on-line transfer parameter estimation: MFEA-II. IEEE Transactions on Evolutionary Computation, 24(1), pp.69-83.”

The contributions of the co-authors are as follows:

- Dr. Gupta, Prof. Ong, and myself shared our initial thoughts for this work. Dr. Gupta and Prof. Ong have contributed to the conception of the idea, theory and algorithmic manifestations of the proposed work. This particular work was envisioned to be one of the very first attempts towards a theoretically principled adaptive knowledge transfer framework in the context of evolutionary multitasking.
- I prepared the manuscript drafts. The manuscript was revised by Dr. Gupta, Prof. Ong as well Dr. Tan.
- I co-developed and verified the theoretical studies with Dr. Gupta and Prof Ong.
- I conducted the experimental studies. Prof. Ong and Dr. Gupta provided some great experimental suggestions for this work, particularly in the context of practical applications.
- Dr. Tan provided her feedbacks throughout the paper revision process.

Chapter 5 and part of Chapter 6 are published as “Bali, K.K., Gupta, A., Ong, Y.S. and Tan, P.S., 2020. Cognizant Multitasking in Multiobjective Multifactorial Evolution: MO-MFEA-II. IEEE Transactions on Cybernetics.”

The contributions of the co-authors are as follows:

- Prof. Ong, Dr. Gupta, myself and Dr. Tan discussed the initial research directions for this work.
- I prepared the manuscript drafts. The manuscript was revised by Dr. Gupta, Prof. Ong and Dr. Tan.
- Dr. Gupta and Prof. Ong helped verify the theoretical aspects of this work.
- I wrote the codes for the underlying algorithm, set up the experiments and analyzed the results. Prof. Ong, Dr. Gupta and Dr. Tan provided further suggestions to improve my experimental designs.

18 Dec. 2020

.....

Date

.....

Bali Kavitesh Kumar

Acknowledgements

Foremost, I thank God for knowledge, protection and resilience to conquer my PhD journey. Compiling this thesis has surely not been unaided, but a combined effort of everyone I have known during the course of my studies: my helpful advisers Prof. Ong Yew Soon and Dr. Tan Puay Siew, my brilliant mentor Dr. Abhishek Gupta, research colleagues, family and friends. These amazing souls deserve my wholehearted thanks.

To my dear parents,
Kishor and Reena Bali

Abstract

It is a conventional wisdom that real world problems seldom occur in isolation. The motivation for this work, inspired from the observation that humans rarely tackle every problem from scratch, is to improve optimization performance through adaptive knowledge transfer across related problems. The scope for spontaneous transfers under the simultaneous occurrence of multiple problems unveils the benefits of multitasking. Multitask optimization has recently demonstrated competence in solving multiple (related) optimization tasks concurrently. Notably, in the presence of underlying relationships between problems, the transfer of high quality solutions across them has shown to facilitate superior performance characteristics - as the cost of re-exploring overlapping regions of the search space is reduced. However, in the absence of any prior knowledge about the inter-task synergies (as is often the case with general black-box optimization), the threat of predominantly negative transfer prevails. Susceptibility to negative inter-task interactions can in fact be detrimental, often impeding the overall convergence behavior. To allay such fears, this thesis presents viable solutions towards automated extraction and transfer of (fruitful) knowledge such that any deleterious effects of otherwise negative inter-task exchanges are suppressed. To this end, an in-depth theoretical analysis is first conducted to unveil the primary caveats which concern the global convergence characteristics of the present day multitasking evolutionary optimization framework. Next, a novel evolutionary computation framework is proposed that enables *online* learning and exploitation of the similarities (and discrepancies) between distinct tasks in multitask settings via probabilistic mixture models. The proposed method is based on principled theoretical arguments that seek to minimize the tendency of harmful interactions between tasks, based on a purely data-driven learning of relationships among them. As a proof of concept, the method is initially validated experimentally on a wide range of synthetic discrete and continuous single-objective benchmarks. Thereafter, a realization of similar concepts of the proposed method is extended to the domain of multi-objective optimization (an omnipresent scenario in our daily lives). It is noteworthy that this work shall be among the first to utilize

probabilistic modeling to capture inter-task relationships between multi-objective optimization tasks in the context of evolutionary multitasking. Empirical studies on a series of benchmark test functions show that the method is able to decipher and adapt to the degree of similarity between distinct multi-objective optimization tasks on the fly. Finally, the practicality of the proposed methods are substantiated on various real world case studies including reinforcement learning, multi-fidelity optimization and evolutionary deep learning. Not only do the practical studies provide insights into the behavior of the methods in the face of several (many) complex tasks occurring at once, but also underscores the benefits of *omnidirectional* knowledge exchanges, the boon of *intentional-unintentional* problem solving capabilities as well as knowledge transfers from low fidelity optimization tasks to substantially reduce the cost of (otherwise expensive) high fidelity optimization.

Contents

Acknowledgements	ix
Abstract	xiii
List of Figures	xix
List of Tables	xxiii
1 Introduction	1
1.1 Premises	1
1.1.1 Motivations	3
1.2 Research Goals	4
1.3 Thesis Outline	5
2 Literature Review	7
2.1 Multitask Learning	7
2.1.1 Motivations for MTL: A quick recap	8
2.1.2 The Problems with Multitasking: No free lunch	9
2.1.3 When should we ideally multitask?	10
2.2 Multitask Optimization	10
2.3 Guises of Multitask Optimization	11
2.3.1 Multitask Bayesian optimization	11
2.3.2 Evolutionary multitasking	11
2.3.2.1 Preliminaries of Evolutionary Multitasking	12
2.3.3 The Multifactorial Evolutionary Algorithms	13
2.3.4 Other Algorithmic Realizations of Evolutionary Multitasking	14
2.4 Chapter Summary	15
3 Theoretical Foundations of Evolutionary Multitasking	17
3.1 Overview of Mixture models in the context of Multitask Optimization	17
3.1.1 Unified search space	18
3.1.2 Knowledge sharing via mixture distributions	19
3.1.3 Actualizing knowledge transfers in multitasking	19
3.2 Does Multitasking Prevent Global Convergence?	20

3.3	Chapter Summary	23
4	Multifactorial Evolutionary Algorithm with Online Transfer Parameter Estimation: MFEA-II	25
4.1	The basic multifactorial evolutionary algorithm	26
4.1.1	Preservation of parent distribution under parent-centric evolutionary operators	27
4.1.2	Preliminaries of the MFEA	27
4.1.2.1	Overview of the MFEA	28
4.2	Theoretical Foundations of the MFEA	29
4.2.0.1	Does the MFEA Converge?	30
4.3	Susceptibility to Negative Inter-Task Interactions in Evolutionary Multitasking	30
4.3.0.1	Analyzing the Rate of Convergence of the MFEA	31
4.4	Transfer Parameter Estimation via Optimal Mixture Modeling	32
4.4.1	Learning the RMP Matrix Online	33
4.4.2	Main Ingredients of MFEA-II	36
4.5	Experimental Study	38
4.5.1	Experimental Configuration	38
4.5.2	Pedagogical Case Study	39
4.5.3	Further Experiments	41
4.5.3.1	Case 1: Trap-5 + hamming distance minimization (HDM) problems	42
4.5.3.2	Case 2: Inverted trap-5 + HDM problems	43
4.5.3.3	Case 3: K randomly generated HDM tasks	45
4.5.3.4	Case 4: LeadingOnes + TrailingOnes problem	46
4.5.4	Synthetic continuous benchmark functions	48
4.6	Chapter Summary	51
5	Cognizant Multitasking in Multi-objective Multifactorial Evolution: MO-MFEA-II	53
5.1	Introduction	53
5.2	Background	54
5.2.1	Multi-objective Optimization	54
5.2.2	Multi-objective Evolutionary Algorithms (MOEAs)	55
5.2.3	Overview of the Existing MO-MFEA	55
5.2.3.1	Tuning the rmp parameter	56
5.3	Cognizant Multitasking MO-MFEA-II	56
5.3.1	RMP matrix	57
5.3.2	Latent Search Distributions in the MO-MFEA	57
5.3.3	Preliminaries on Probabilistic Model-based Search	58
5.3.4	Probabilistic Mixture Modeling in MO-MFEA-II	59
5.3.5	Learning the RMP Matrix Online in MO-MFEA-II	60
5.3.6	Summary of the MO-MFEA-II	61

5.3.6.1	Preliminary definitions	61
5.3.6.2	Algorithm summary	62
5.3.6.3	Selective evaluations in MO-MFEA-II	63
5.4	Experimental Study	64
5.4.1	Experimental Configuration	64
5.4.2	Specification of Benchmark MOOPs	65
5.4.2.1	Problem 1	66
5.4.2.2	Problem 2	66
5.4.2.3	Problem 3	66
5.4.2.4	Problem 4	67
5.4.2.5	Problem 5	67
5.4.3	Measuring Complementarity Between Constitutive Tasks . .	68
5.4.4	Experimental Results	68
5.4.4.1	Complete Intersection and High Similarity: CI-HS	68
5.4.4.2	No Intersection and Low Similarity: NI-LS	70
5.4.4.3	Study with more than two tasks	71
5.5	Chapter Summary	73
6	Applications of Evolutionary Multitasking	75
6.1	Introduction	75
6.2	Part I: The Double-Pole balancing Problem: A practical study of MFEA-II	76
6.2.1	Experimental Configuration	77
6.2.2	Preliminaries of the Double-Pole Balancing Problem	78
6.2.2.1	Omnidirectional Transfer in Multitask Optimization	79
6.2.2.2	Some insights on Many Tasking	80
6.3	Part II: Evolutionary Multitasking for High Fidelity Optimization: A practical study of MO-MFEA-II	82
6.3.1	Multi-objective Risk Averse UAV Path Planning	84
6.3.1.1	Simulation details	84
6.3.2	Experimental Configuration	86
6.4	Part III: Agile Reconfiguration of Deep Models via Evolutionary Multitasking	89
6.4.1	Premise	90
6.4.2	Multi-criteria deep reconfiguration	92
6.4.3	Multi-master multi-criteria optimization	94
6.4.3.1	Multitasking approach - a brief recap	94
6.4.3.2	Neuroevolutionary approach to deep reconfiguration	95
6.4.3.3	Standard deep reconfiguration	95
6.4.3.4	Deep reconfiguration via evolutionary multitasking	95
6.4.4	Experimental Study	98
6.4.5	Dataset	98

6.5 Chapter Summary	100
7 Conclusions and Future Work	103
7.1 Future Research Directions	104
7.1.1 Scalability to many-tasking	104
7.1.2 Improved genetic transfers and solution representations	105
7.1.3 Extensions to constrained optimization	106
7.1.4 What lies further ahead in the real world?	106
A Supplementary Material	109
A.1 Proof of Lemma 4.2.1	109
A.2 Visualization of Synthetic Benchmarks proposed in Chapter 4	111
A.2.1 Visualization of Complete Intersection (CI) Case in 1- D	111
A.2.2 Visualization of No Intersection (NI) Case in 1- D	112
List of Author's Publications	113
Bibliography	115

List of Figures

1.1	Process of learning and knowledge transfers.	1
2.1	The concept parameter sharing in MTL for a deep network. Instead of learning different models for tasks A, B and C independently, a single generic MTL network with task-specific output layers is preferred [1].	9
2.2	A comparison between (A) MTL, (B) Multitask optimization and (C) Multi-objective optimization. In MTL, multiple tasks are simultaneously learned by a shared model; Multitask optimization solves several optimization tasks in tandem via occasional inter-task knowledge transfers (e.g. genetic transfers); and multi-objective optimization establishes a Pareto trade-off between conflicting objectives (i.e., three objectives in this case) of a particular task. Herein, “KB” and “Alg” represent the knowledge base and the devised algorithm, respectively.	13
4.1	Left: General framework of the existing MFEA. Notice that MFEA employs an <i>offline</i> scalar <i>rm</i> p assignment. Right: MFEA-II with the added <i>online</i> <i>RMP</i> matrix learning module.	36
4.2	Left: Pairwise <i>rm</i> p’s learned between distinct tasks over successive generations. The shaded regions span one half standard deviation on either side of the mean. Right: Convergence trends (averaged over 30 independent runs) achieved by all the algorithms on the trap-5 problem. As shown, both multitasking variants are superior to CEA, with MFEA-II being the most dominant.	40
4.3	Trap-5 convergence trends.	42
4.4	Learned <i>rm</i> p trends across generations. The shaded regions span one-half standard deviation on either side of the mean.	43
4.5	Average learned <i>rm</i> p trend between trap-5 and a series of HDM problems dictated by ρ . The error bars span one-half standard deviation on either side of the mean.	44
4.6	Inverse trap-5 convergence trends.	44
4.7	Average learned <i>rm</i> p trend between the inverse trap-5 and a series of HDM problems dictated by ρ . The error bars span one-half standard deviation on either side of the mean.	45

4.8	Averaged performance of MFEA-II, MFEA-I and CEA on 10 randomly generated HDM tasks.	45
4.9	Few representative <i>rmf</i> curves between pairs of randomly generated HDM tasks. Other inter-task <i>rmf</i> values showcase similar trends.	46
4.10	Learned <i>rmf</i> trend between LeadingOnes and TrailingOnes problems over successive generations. The shaded regions span one-half standard deviation on either side of the mean.	47
4.11	Convergence trends of MFEA-II and MFEA-I.	47
4.12	Learned pairwise <i>rmf</i> values between pairs of tasks achieved by MFEA-II on different problem sets characterized by different degrees of inter-task similarities. The plots are averaged over 30 independent runs with the shaded regions spanning one-half standard deviation on either side of the mean.	50
4.13	Convergence trends (averaged over 30 independent runs) achieved by the three tested algorithms on the problem set CI+HS with high degrees of inter-task similarity.	51
5.1	Top: Convergence trends (averaged over 30 independent runs) achieved by all the algorithms in the CI-HS composite MOOP benchmark. Below: Pairwise <i>rmf</i> 's learned between tasks T_1 and T_2 over successive generations. The shaded regions span one half standard deviation on either side of the mean.	69
5.2	Top: Convergence trends of all the algorithms in the NI-LS composite MOOP benchmark. Bottom: Pairwise learned <i>rmf</i> 's.	71
5.3	Convergence trends (averaged over 30 independent runs) achieved by all the algorithms on the 5 tasks.	72
6.1	Double-pole balancing problem	78
6.2	Comparison of total number of function evaluations consumed by each of the algorithms in solving all the six different tasks.	81
6.3	Pairwise <i>rmf</i> 's of MFEA-II, learned between constitutive tasks of different short pole lengths. The plots are averaged over 30 independent runs and the shaded regions span one-half standard deviation on either side of the mean.	82
6.4	Multiple UAVs in action.	83
6.5	Left: Geographical region of interest. Minimum distance (dotted lines) and minimum risk (solid lines) trajectories for 5 different UAVs in an area with different signal strength, wind speeds and 1 geofence (red circumference). Right: Trajectory [defined by the cubic spline interpolation of 4 control points and the start (S) and destination (E) of a particular UAV's mission.	85
6.6	Sample analysis of 5 UAVs flying in an area with different signal strength and wind conditions.	85
6.7	Left: Convergence trends (averaged over 10 independent runs) achieved by NSGA-II and MO-MFEA-II for experimental Configuration 1. Right: Pairwise <i>rmf</i> 's learned between tasks T_1 and the helper task T_2 over total run time. The shaded regions span one half standard deviation on either side of the mean.	88

6.8	Left: Convergence trends achieved by NSGA-II and MO-MFEA-II for experimental Configuration 2. The MO-MFEA-II employs two helper tasks, respectively. Right: Pairwise <i>rmf</i> 's learned between tasks T_1 and the helper tasks T_2 and T_3 . The shaded regions span one half standard deviation on either side of the mean.	88
6.9	Elastic and agile cloud computing platform simultaneously catering for small, unknown or variable compute requirements at the edge. As per unique device requirements and user-specific preferences, diverse specialized (master) models such as 1-class, 2-class or multi-class output can be queried from the cloud hosting a gigantic, pretrained, generalized (Jack of all trades) deep model. These lightweight master models can then be conveniently accessed on portable devices with lesser compute powers.	90
6.10	A standard neuroevolutionary approach to deep model reconfiguration.	96
6.11	Multiple reconfiguration tasks (aimed at switching on and off filters of a deep CNN) are handled simultaneously via evolutionary multitasking. The scope for spontaneous knowledge transfers under the simultaneous occurrence of multiple tasks unveils the benefits of multitasking.	97
6.12	Convergence curves for all tasks $T_1 - T_{10}$. The bottom right plots show the learned inter-task <i>rmf</i> 's between distinct task pairs. . . .	100
A.1	In the complete intersection case, the global optimum of all tasks maps to the same point in the unified search space.	111
A.2	In the no intersection case, the global optima of the two tasks T_1 and T_2 are separated.	112

List of Tables

4.1	Benchmark problem sets [2]	48
4.2	Comparison of MFEA-II with MFEA-I and CEA on the continuous minimization benchmark problem sets. The mean and standard deviations of 30 independent runs are provided. Boldface entries marked with * are best and significantly the best (by Wilcoxon test with significance level 0.05). Best but not significantly the best entries are only boldface.	49
5.1	Summary of test problems	68
5.2	Inter-task similarities R_s	72
5.3	Average IGD values achieved by NSGA-II, MO-MFEA-II and MO-MFEA (with a range of $rm\mu$ values) for all $K = 5$ Tasks. Bold face entries are the best (by WILCOXON test with significance level 0.05).	72
5.4	Average IGD values for all $K = 4$ Tasks	73
5.5	Average IGD values for all $K = 3$ Tasks	74
6.1	Percentage success rate (out of 30 runs) of three different tasks. They are either solved separately (CEA), or merged into pairs or a triplet (MFEA-II).	80
6.2	Comparison of various methods on different short pole lengths l_s , for the double-pole balancing problem. The Table shows the percentage success rate (%) out of 30 independent runs. Best entries are highlighted in bold.	81
6.3	Practical study experimental configurations. Different task instances are generated by defining different discretization step sizes (d_{step})	87
6.4	IoT Device Categories	98
6.5	Reconfiguration tasks and the corresponding class labels	99
6.6	MNIST Example: Master Model accuracies vs Jack of all trade model	101

Chapter 1

Introduction

1.1 Premises

In our immensely complex world, having a conscious understanding of everything becomes practically impossible. Processes of learning and the transfer of learning are central to understanding how humans have evolved and developed important competencies. *“It is especially important to understand the kinds of learning experiences (experiential knowledge) that lead to transfer, defined as the ability to extend what has been learned in one context to new contexts”* [3].

(A) A baby mastering how to eventually walk.

(B) Math as a prerequisite for ML [4]

FIGURE 1.1: Process of learning and knowledge transfers.

Humans have the ability to identify recurring patterns in diverse situations encountered over a lifetime, constantly understanding relationships between tasks and efficiently solving them through knowledge reuse. For instance, a toddler learning how to walk is not an overnight process but a self-driven learning task. Figure 1.1a demonstrates the gradual acquisitions of basic reflexes such as crawling, calibrating balance, stepping reflex and eventually proficient enough to walk independently. Other intrinsic scenarios include learning to ride a motorcycle by leveraging on prior knowledge of bicycles, building strong mathematics foundations for a good grasp of machine learning (ML) techniques, etc.

The capacity of artificial intelligence systems to mimic such cognitive behaviors for effective problem solving is deemed invaluable, particularly when tackling real world problems where speed and accuracy are critical. In machine learning, the notion of knowledge transfer has manifested in many guises, ranging from *transfer learning* [5–9] to *multitask learning* [10]. Transfer learning leverages on a pool of available data from various source tasks to improve the learning efficacy of a related target task. Interestingly, while transfer learning mainly focuses on just the target task, multitask learning, in contrast, learns all tasks simultaneously by sharing information across the different tasks, in order to improve the *overall* generalization performance. Simply put: *what is learned for each task can help other tasks be learned better*. Fueled by data abundance, the concepts of knowledge transfer have mainly been prominent in the field of predictive analytics (e.g. classification and regression).

However, it is contended that real world optimization problems seldom occur in isolation. That is, the presence of latent similarities, even between distinct black-box optimization instances, are commonplace. This is particularly true when we are working within a particular domain of application. With this in mind, a plethora of information can potentially be shared across related problems. Arguably, it may seem unnecessarily laborious to isolate and solve any such related optimization problems from scratch (i.e., from a zero-state knowledge) [11]. Consequently, the existence of useful information across complementing problems, opens doors for fruitful knowledge transfer. In practical settings, leveraging on such a rich pool of knowledge, can yield substantial cost-saving benefits [12]. Thus, the ability to solve many tasks, with the scope of knowledge exchange among them, has been

a longstanding goal of artificial intelligence systems targeting improved problem-solving efficacy [13].

Over the years, a handful of success stories have surfaced in the relatively young field of *transfer optimization* (TO), encompassing sequential transfers [14–19] as well as multitasking [20–22]. Similar to transfer learning, sequential transfer optimization utilizes high quality solutions from various source problems to solve a target optimization task. Therein, the transfer of knowledge (from past optimization problems to present) is conducted in a *unidirectional* fashion. That is, only a single target task is optimized at a time; re-optimization of the source task(s) is irrelevant. Solving multiple tasks concurrently is thus, beyond the scope of sequential transfer optimization techniques.

Multitasking effectively fills this void. Surprisingly, in spite of the longevity of *multitask learning*, the idea has only recently been introduced into the arena of *optimization problem-solving* [20, 21]. The notion of *multitask optimization* aims to facilitate efficient resolution of multiple problems simultaneously, by promoting *omnidirectional* knowledge transfer for greater synergistic search. Simply put, all the optimization tasks can in principle mutually help each other.

1.1.1 Motivations

Despite its practical implications, preliminary works have indicated that the performance of multitask optimization is highly sensitive to the degree of underlying inter-task similarities. To elaborate, each optimization task may not always be closely related to all other available tasks. In such scenarios, blindly assuming relatedness and solving (any) unrelated optimization problems concurrently can potentially lead to performance slowdowns [2]. Overlooking such inter-task discrepancies and sharing knowledge across multiple diverse problems give rise to a phenomenon known as *negative transfer*. In fact, the threat of negative transfer has commonly been reported in the machine learning literature [10, 23], thereby highlighting a central pitfall of multitasking in general (i.e., be it in learning or in optimization).

Therefore, it is imperative to design multitasking algorithms adept at leveraging on positive transfers (when available) while suppressing any deleterious effects of negative exchanges otherwise. This thesis focuses on the use of evolutionary algorithms (EAs) since they generally allow flexible data representations (i.e., discrete and continuous domains), and have shown to combine well with various data modeling techniques including probabilistic models [24] as well as Bayesian networks [25], to name a few. In particular, attention is drawn towards the present day candidate evolutionary algorithms for multitasking, namely the *multifactorial evolutionary algorithm* (MFEA) [21] and the *multi-objective multifactorial evolutionary algorithm* (MO-MFEA) [22].

1.2 Research Goals

Based on the above facts, this thesis builds upon the following research objectives:

- Conduct a theoretical analysis of a generic population-based multitask optimization framework, focussing on how *inter-task* knowledge transfers can actuate and subsequently validate its global convergence guarantees. Thereafter, unveil the challenges of the present day MFEA, particularly highlighting its susceptibility to predominantly negative knowledge transfers - which often tends to impede the overall convergence characteristics.
- Propose a novel evolutionary computation framework, MFEA-II, adept at *online* learning and exploitation of the similarities (and discrepancies) between distinct tasks in multitask settings. In contrast to its predecessor, MFEA (which employs a simplistic *offline* knowledge transfer regime), the proposed MFEA-II shall learn inter-task relationships based on overlaps in the probabilistic search distributions derived from data generated during the course of multitasking - and accordingly adapt the extent of genetic transfers online. Subsequently, validate its efficacy on a wide range of discrete and continuous single-objective benchmarks.
- Thereafter, analyze how similar concepts above can apply to the domain of multi-objective optimization as well. As indicated previously, this work shall be among the first to utilize probabilistic modeling to capture inter-task relationships between multi-objective optimization tasks in the context

of evolutionary multitasking. Henceforth, the proposed work shall be referred to as MO-MFEA-II, a successor to the existing MO-MFEA.

- Finally, highlight the practicality of the aforementioned frameworks (i.e., MFEA-II and MO-MFEA-II) in real world scenarios, particularly in robotics and expensive optimization settings such as high-fidelity optimization and evolutionary deep learning.

1.3 Thesis Outline

The outline of the thesis is as follows:

- Chapter 1 outlines the motivations and research goals.
- Chapter 2 discusses the backgrounds of multi-task learning cum multi-task optimization. Further, it highlights the most notable approaches for multi-task optimization, spanning Bayesian and evolutionary methods, and their associated limitations.
- Chapter 3 presents a generic evolutionary multitask optimization framework from a probabilistic modelling perspective to realize how knowledge transfers can be actuated as a consequence of inter-task interactions. Next, an in-depth study of its convergence characteristics is conducted to theoretically validate that (under appropriate assumptions) evolutionary multitasking does not prevent global convergence.
- Chapter 4 analyzes the present day MFEA (a candidate evolutionary multitask algorithm) and highlights its shortcomings. Thereafter, it introduces the *multifactorial evolutionary algorithm with online transfer parameter estimation*, namely the MFEA-II, an adaptive successor to the former MFEA (with offline knowledge transfer regime).
- Chapter 5 puts forward the *cognizant multitasking algorithm*, MO-MFEA-II, focussed on processing multiple complex multi-objective optimization tasks simultaneously via adaptive knowledge transfers.

- Chapter 6 showcases the practicality evolutionary multitasking in diverse applications ranging from complex controller design tasks in robotics to the otherwise expensive high-fidelity optimization of the risk averse UAV path planning problem as well as evolutionary deep learning.
- Chapter 7 concludes the thesis with an overview of the results and analysis from Chapters 3-6 and discusses foreseeable future research directions.

Chapter 2

Literature Review

This chapter provides a review of *knowledge sharing* that has been predominant in the field of artificial intelligence research today. To begin with, it briefly describes the concepts of multitask learning. Thereafter, it presents a comparison between the most recent strides in multitask optimization, spanning Bayesian optimization and evolutionary multitasking. In addition to that, a survey of various existing evolutionary multitasking frameworks is conducted and their respective limitations are highlighted.

2.1 Multitask Learning

The human brain is a self-learning system. Among various other things, it has the general capability of comprehending its surroundings - being “*aware of what is going on*” and “*figuring out*” what to do [26, 27]. This intrinsic feature enables humans to draw inferences from what one experiences, recognize recurring patterns (relationships) between diverse tasks, and share knowledge across related problems [28]. It is indeed this ability to leverage on what we already know to tackle related tasks that plays a significant role in boosting our efficiency of problem-solving and decision-making [11].

The ability to learn from experience is considered invaluable for machines as well. Given modern technologies with large-scale data storage and seamless communication facilities, the development of machines that mimic human-like behaviors of autonomously extracting, processing and transferring learned knowledge across related problems is surely an attractive proposition. In this regard, the concept of *multitasking* has drawn much interest in the field of computational intelligence [10, 20, 21, 29, 30]. Not surprisingly, present day artificial intelligence systems are emerging as powerful tools for multitasking, capable of simultaneously handling myriad tasks with greater speed and accuracy. As has been emphasized in Chapter 1, the premise of multitask learning is simple; *what is learned for each task can help other tasks be learned better*. Given the data deluge (in the advent of internet), the notion of multitask learning (MTL) has been getting popular by the day, being mostly prominent in the field of predictive analytics, ranging from classification and regression tasks. In what follows, various motivations of MTL is provided. In particular, it provides context for current neural network-based methods by discussing the relevant literature.

2.1.1 Motivations for MTL: A quick recap

There are various reasons/motivations that warrant the use of MTL. Whether it is from classical biological inspirations of human learning and experiential knowledge sharing or purely from machine learning perspectives, the long term goals have been the same - learning new tasks by applying the knowledge base we have acquired for learning related tasks.

In the context of (deep) neural networks, it is noted that machine learning models generally require a large volume of data for training. However, we often end up with many tasks for which the individual datasets are insufficiently sized to achieve the desired performance. In this case, if some of these tasks are related (e.g. predicting many diseases and outcomes from a patient's profile), we can merge the features and labels into a single larger training dataset; thus, allowing us to take advantage of shared information from related tasks to build a sufficiently large dataset.

Further, it is a proven fact that MTL improves the generalization capability of a model. To elaborate, the information learned from related tasks improves the

model’s ability to learn a useful representation of the data, which reduces overfitting and enhances generalization. Also, MTL can substantially reduce the overall training times by training a single (generic) model instead of investing time training many models on multiple (related) tasks.

FIGURE 2.1: The concept parameter sharing in MTL for a deep network. Instead of learning different models for tasks A, B and C independently, a single generic MTL network with task-specific output layers is preferred [1].

Moreover, MTL is crucial in various practical scenarios, such as deploying a model in an environment with limited computational power. Notably, machine learning (particularly deep learning) models often have many (in the order of millions or billions) parameters that need to be stored/processed in memory. Thus, for applications where the computational power is limited (e.g. edge devices), it is preferable to have a single MTL network with some shared parameters, as opposed to multiple standalone single-tasking models processing related tasks separately [1, 31]. Figure 2.1 illustrates the notion of (hard) parameter sharing for MTL in deep networks. Herein, the hidden layers between all tasks of interests are shared while keeping several task-specific output layers; i.e., a multi-headed neural network with shared losses trained via back propagation.

2.1.2 The Problems with Multitasking: No free lunch

Like most other (supervised) machine learning algorithms, MTL does not defy the “no free lunch theorem” [32] as well; i.e., it is *not* universally the best-performing algorithm for all problems out in the wild. Despite the various advantages of MTL, there are some cases where the approach can actually hurt performance. During the

training of a MTL network, tasks can compete with each other in order to achieve a better learning representation i.e. one or more tasks can dominate the training process, often resulting in a biased model. For instance, consider a MTL scenario wherein instance segmentation (i.e., segmenting a separate mask for each individual object in an image) is trained alongside semantic segmentation (i.e., classifying of objects at pixel level). Under such settings, the latter task often dominates the learning process unless some task balancing mechanism is employed [33].

Furthermore, the loss function of MTL may also be more complex as a result of multiple summed losses, thereby making the optimization more difficult and exorbitantly expensive. In these cases, there is a negative effect of cooperating on multiple tasks (in the order of thousands or millions), and individually trained networks that are laser focussed on single tasks may just perform better.

2.1.3 When should we ideally multitask?

Over the years, answering this question has not been simple, and still remains to be an active area of research today. Because of the increased importance of multitask learning, a plethora of methods have been proposed to automatically learn which tasks should be jointly learned, and which tasks should be learned separately. Essentially the ultimate goal is towards mimicking the (cognizant) human brain to spawn next generation adaptive MTL algorithms that favour positive knowledge transfers and suppress negative transfers across thousands (if not millions) of tasks in realistic scenarios. Some recent advancements can be found in [34–36].

2.2 Multitask Optimization

While significant advancements have been made in the context of multitask learning, attempts made to transfer knowledge in the context of *optimization* problems (which typically start the search with *zero* prior data) [11], have relatively been scarce, and received little visibility. *Multitask optimization* has only recently been explored as a means for solving multiple optimization problems concurrently [20, 21, 23]. In what follows, a brief review of recent strides in the field of multitask optimization is presented.

2.3 Guises of Multitask Optimization

2.3.1 Multitask Bayesian optimization

Multitask Bayesian optimization has been a prominent advancement in the context of bolstering hyperparameter optimization for automatic machine learning [20]. Its prime success is achieved by utilizing multitask Gaussian process models [23] to adaptively learn inter-task dependencies and improve the overall search of a Bayesian optimization algorithm. However, Bayesian optimization is generally limited to problems with fairly low or moderate dimensions. This is because, as the dimensionality of the search space increases, the number of data points required to ensure a good coverage of the space (for learning accurate Gaussian process models) increases exponentially (often referred to as the cold start problem) [37]. Furthermore, the applications of Bayesian optimization are primarily limited to continuous search spaces and do not directly apply to combinatorial search spaces wherein indefinite kernels maybe difficult to deal with [38].

2.3.2 Evolutionary multitasking

In contrast, evolutionary algorithms (EAs) have been promising in bridging the aforementioned gaps of multitask bayesian optimization. EAs offer immense flexibility in accommodating diverse data representations (be it continuous or combinatorial [39]), and scale fairly well to higher dimensions [40, 41]. Recently, evolutionary multitasking, under the label of multifactorial optimization [21], has demonstrated various success stories in solving multiple optimization problems simultaneously, encompassing discrete, continuous, single-, as well as multi-objective optimization [21, 22, 42–44]. Notably, evolutionary multitasking is not only limited to [simplistic] unconstrained optimization domains, but has equally triumphed in more realistic scenarios faced with challenging (dynamic) constraints as well; whether it has been standard optimization settings with predefined equality and inequality constraints, or classical combinatorial settings such as capacity-constrained Knapsack problems [45], dynamic time-window constrained capacitated vehicle routing problems (CVRPs) [46], quadratic assignment problems (QAPs) [47], etc.

Beyond this, evolutionary multitasking has also found its way in the field of machine learning; from training an ensemble of decision trees [48] to feedforward neural networks with modular knowledge representation [49].

2.3.2.1 Preliminaries of Evolutionary Multitasking

This section provides a brief description of a multitask setting that deals with the co-existence of multiple related search/optimization problems, such that knowledge can be transmitted among them. Consider a series of K (related) optimization tasks that are labelled as $T_1, T_2, \dots, T_K$. Without loss of generality, all tasks are assumed to be minimization problems. The k^{th} task, denoted T_k , is considered to have a search space $\mathbf{X}_k$ on which the objective function is defined as $f_k : \mathbf{X}_k \rightarrow \mathbb{R}$. Here, f_k can either be a scalar, for a single-objective optimization problem, or even a vector constituting a multi-objective optimization problem. In addition, each of the tasks may be constrained by several equality and/or inequality conditions that must be satisfied for a solution to be considered feasible. In such a setting, evolutionary multitasking paradigm builds on the implicit parallelism of population-based search with the aim of finding $\{x_1, x_2, \dots, x_{K-1}, x_K\} = \operatorname{argmin}\{f_1(x), f_2(x), \dots, f_{K-1}(x), f_K(x)\}$, where x_k is a feasible solution in $\mathbf{X}_k$.

Evolutionary multitasking facilitates efficient resolution of multiple self-contained optimization tasks simultaneously by promoting omnidirectional knowledge transfers. That is, all tasks can benefit from each other via mutual knowledge sharing [12]. Notably, in the presence of underlying relationships between problems, the transfer of high-quality solutions (via implicit genetic transfers) across them has shown to facilitate superior performance characteristics.

As an important aside, it is worthwhile to highlight the distinction between evolutionary multitasking and multi-objective optimization. While the former aims to leverage the implicit parallelism of population-based search to exploit latent genetic complementarities between multiple different tasks, multi-objective optimization attempts to efficiently resolve conflicts among competing objectives of the same task; i.e., assuming the existence of a Pareto trade-off between the objectives, for which a particular algorithm produces an estimation in the form of a set of possible solutions. As such, no unique solution exists for each task, but rather

FIGURE 2.2: A comparison between (A) MTL, (B) Multitask optimization and (C) Multi-objective optimization. In MTL, multiple tasks are simultaneously learned by a shared model; Multitask optimization solves several optimization tasks in tandem via occasional inter-task knowledge transfers (e.g. genetic transfers); and multi-objective optimization establishes a Pareto trade-off between conflicting objectives (i.e., three objectives in this case) of a particular task. Herein, “KB” and “Alg” represent the knowledge base and the devised algorithm, respectively.

different solutions that meet every objective to a certain degree - providing practitioners the flexibility to choose desired ones. Interestingly, note that each task in an evolutionary multitasking environment may itself comprise multiple objectives, which highlights the greater generality of the evolutionary multitasking paradigm with respect to standard multi-objective optimization settings.

With the above, distinctions between multitask learning, (evolutionary) multitask optimization and multi-objective optimization can be clearly drawn. A recent survey by Osaba et al. [50] perfectly illustrates the core differences between these concepts (see Figure 2.2).

2.3.3 The Multifactorial Evolutionary Algorithms

Since the conceptualization of evolutionary multitasking, various algorithmic realizations have come to the fore [9, 12, 21, 22, 30, 51–57]. Notably, the *crossover-based* multifactorial evolutionary algorithms (i.e., MFEA and MO-MFEA) [21, 22] have so far been the most prominent ones. Herein, the primary mode of knowledge transmission herein is through implicit genetic transfers, facilitated by inter-task crossovers between solutions belonging to different tasks. These algorithms have shown that the crossover (or recombination) operator when applied in the unified genotype space, in conjunction with the effects of evolutionary selection pressure,

form the key ingredients driving implicit knowledge transfer across constitutive optimization tasks in multitasking environments for improved problem solving. For the sake of brevity, technical details of these algorithms have been omitted in this section. A thorough analysis of the same shall be conducted in the following chapter.

One of the key attributes owing to the success of the crossover-based MFEA variants is the use of a *unified genotype space* (i.e., a unified search space or a unified solution representation scheme) encompassing all tasks in a multitasking environment. For instance, in the case of box-constrained continuous optimization tasks, such unification can be achieved in a straightforward manner by a simple linear map between task-specific solution space and the unified space — which is also continuous and is often defined in the range $[0, 1]^D$. The unification forms the basis of implicit knowledge transfer across optimization tasks in the form of encoded genetic material. Interestingly, unified representation schemes can be defined to encompass a variety of tasks belonging to different domains (i.e., in continuous as well as discrete optimization), thereby laying the foundation for general-purpose cross-domain multitask optimization frameworks. In fact, unification, to some extent, elicits an added layer of privacy, particularly in distributed networking environments (as task specific solutions can only be decoded with its own mapping parameters/keys). The concepts of unification shall be covered in more detail in the subsequent chapter.

While the aforementioned multitasking solvers (such as the MFEA) have led to recent success stories, a known shortcoming of existing methods is their inability to adapt the extent of transfer in a principled manner. Thus, in the absence of any prior knowledge about the relationships between optimization functions, a threat of predominantly *negative* (harmful) transfer prevails.

2.3.4 Other Algorithmic Realizations of Evolutionary Multitasking

Beyond the basic MFEA variants, various other associated algorithmic developments have also surfaced recently [12, 21, 22, 58]. In particular, a handful of methods equipped with adaptive knowledge transfer capabilities have been proposed. For example, Feng *et. al.* [29] proposed to learn optimal linear mappings between different continuous multi-objective tasks using a denoising autoencoder.

In this method, the learned mappings serve as a bridge between tasks, providing a transformation of search spaces (and solutions) such that adaptive knowledge transfers can be conducted.

Further, in [59], a credit assignment-based technique was employed to adapt knowledge transfers between tasks in a manner that promotes exploration, thereby preventing solutions from getting trapped in local optima. More recently, Lin *et. al.* [60] have utilized incremental Naive Bayes classifiers to select valuable solutions to be transferred during the multitasking search. In their proposed method, the authors also use a randomized mapping that enhances the exploration capacity of transferred solutions, with the aim of improving overall convergence behavior.

While there has been growing interest in designing adaptive multitask solvers, many existing methods are designed based on prior assumptions of task relatedness, such that solutions from a source may be directly transferable to a target task. Some other methods have also been proposed that learn a mapping between tasks with the goal of inducing high (ordinal) correlation between their respective objective functions [61]. The learned mappings are then applied to transferred solutions, increasing the likelihood of them being beneficial to the target task. Nevertheless, it is noted that existing methods typically do not provide any explicit measures to guard against the threat of negative transfer; for example, when the learned mapping is inaccurate.

In light of the above, the key novelty of the proposed works in this thesis shall be to take a data-driven approach to explicitly guard against negative transfers plaguing the present day evolutionary multitasking engines, MFEA and MO-MFEA.

2.4 Chapter Summary

This chapter has provided the background of multitasking within the scope of computational intelligence, ranging from multitask learning to multitask optimization. Notably, an extensive review of the recent developments in the field of evolutionary multitasking is presented, highlighting its strengths and limitations. In particular, this chapter sheds light on the inefficacy of existing methods to automatically extract and transfer fruitful knowledge across tasks while curbing any threats of

negative inter-task influences. These surveys have arisen motivations for developing novel and efficient frameworks to improve the performance of the present day multitask optimization algorithms and combat their limitations. To this end, in what follows, an in-depth theoretical study of evolutionary multitasking is first conducted in Chapter 3. Accordingly, this study shall form the foundations for proposing theoretically principled adaptive multitasking frameworks in the subsequent chapters.

Chapter 3

Theoretical Foundations of Evolutionary Multitasking

This chapter serves to present the theoretical formalizations and analysis of the present day evolutionary multitasking framework(s)¹. To this end, an instantiation of a multitasking evolutionary optimization framework from a probabilistic modeling perspective is first described. Thereafter, it unveils how the principled integration of probabilistic models drawn from different tasks can actualize knowledge transfers while curbing negative inter-task interactions. The chapter concludes by analyzing the effect of the mixture distribution on the global convergence behavior of multitasking, unveiling that asymptotic global convergence properties of probabilistic modeling based EAs continue to be preserved.

3.1 Overview of Mixture models in the context of Multitask Optimization

Herein, we present an overview of constructing and sampling probabilistic mixture distributions—that combine search distributions drawn from different tasks—as a way of initializing knowledge exchange in a multitask setting. It must be noted that for such a combination to be possible, all distributions must be defined in a common/shared space, which we refer to as the *unified search space*.

¹The work in this chapter has been published in [62]

3.1.1 Unified search space

The importance of search space unification in multitasking is to encompass all the individual search spaces of different optimization tasks such that we have a shared platform on which the transfer of knowledge can seamlessly take place [11, 22, 63]. For the sake of brevity, consider K *maximization* tasks $\{T_1, T_2, \dots, T_K\}$ to be solved concurrently. The search space dimensionality of each of the optimization tasks is given by $D_1, D_2, \dots, D_K$ respectively. In such a scenario, we can define a *unified space* $\mathbf{X}$ of dimensionality $D_{unified} = \max\{D_1, D_2, \dots, D_K\}$, such that there is sufficient flexibility to encompass each task in the multitask setting. The motivation behind using such a unification scheme, instead of simply concatenating the K distinct search spaces such that $D_{unified} = (D_1 + D_2 + \dots + D_K)$, is two-fold:

- Firstly, it helps circumvent the challenges associated with the curse of dimensionality when solving several tasks with multidimensional search spaces simultaneously.
- Secondly, it is considered to be an effective means of accessing the power of population-based search, promoting the discovery and implicit transfer of useful genetic materials from one task to another in an efficient manner.

With this, we propose $\mathbf{X}$ to be limited to the range $[0, 1]^{D_{unified}}$, which serves as a *continuized* unified space into which all candidate solutions are mapped (*encoded*). Accordingly, while tackling the k^{th} task, a subset of D_k variables are extracted from a candidate solution $x \in \mathbf{X}$ and *decoded* (inverse of the mapping function) to a task-specific solution representation for evaluation. For the case of box-constrained continuous optimization problems, a simple linear scaling offers a suitable one-to-one mapping between the task-specific solution space and the unified space. On the other hand, for the variety of discrete/combinatorial problems, different encoding/decoding processes can be conceived - readers are referred to [63] for a comprehensive overview.

3.1.2 Knowledge sharing via mixture distributions

Without loss of generality, for all $k \in \{1, 2, \dots, K\}$, let the (sub)populations associated with the k^{th} task be denoted as P^k . The corresponding objective function is denoted as f_k and $f_k^* = f_k(x^*)$ denotes the global maximum of the k^{th} task. Note that we assumed x to be a candidate solution in the *unified search space* $\mathbf{X}$. With this background, a key assumption in the subsequent theoretical deductions of multitasking is that for any task T_k , the Borel measure of a set $H = \{x | x \in \mathbf{X}, f_k(x) > f'_k\}$ is positive for any $f'_k < f_k^*$. In an evolutionary multitasking algorithm, let the subpopulations associated with each of the K tasks at time step $t > 0$ originate from the underlying probability distributions $p^1(x, t), p^2(x, t), \dots, p^K(x, t)$ respectively. Accordingly, in the process of facilitating *inter-task* interactions, the offspring population generated for the k^{th} task at the t^{th} iteration is considered to be drawn from the following mixture distribution,

$$q^k(x, t) = \alpha_k \cdot p^k(x, t) + \sum_{j \neq k} \alpha_j \cdot p^j(x, t). \quad (3.1)$$

The finite mixture $q^k(x, t)$ is a linear combination of all K available distributions, with mixture coefficients given by the α 's [64]. Note that $\alpha_k + \sum_{j \neq k} \alpha_j = 1$, and $\alpha'_s \geq 0$.

3.1.3 Actualizing knowledge transfers in multitasking

The onset of knowledge transfer in multitasking occurs *during subsequent sampling of candidate solutions from the mixture distribution specified in Eq. (3.1)* [12, 65]. The degree of mixing (i.e., the amount of knowledge exchange) is mandated by the mixture coefficients α 's. It is noteworthy that in the absence of prior knowledge about the inter-task relationships, any inappropriate (blind) mixing of probability distributions will pose a threat of predominantly negative transfers by wrongly biasing the search distribution. To alleviate any such issues, it is imperative to facilitate the *optimal mixture* of the component densities by ensuring that the mixture coefficients (α 's) are appropriately learned *online*.

3.2 Does Multitasking Prevent Global Convergence?

In this section, we highlight the effects of the mixture distribution on the global convergence behavior of multitasking. In particular, we show that asymptotic global convergence properties of probabilistic modeling based EAs continue to be preserved. For mathematical convenience, the theoretical study is carried out for large (sub)population sizes (N), i.e., under the assumption that $N \rightarrow \infty$. Although such an assumption may not be completely practical, it is considered a reasonable simplification for developing the foundations of our proposals. Subsequently, the practical implications of our derived theoretical results are verified through experimental studies in Chapters 4 - 6.

In the limit of large values of N , it follows from the Glivenko-Cantelli theorem [66] that the empirical probability density of the (sub)population P^k closely approximates the true underlying distribution $p^k(x)$. Further, we assume that for the initial population, $p^k(x, t = 0)$ is positive and continuous in $\mathbf{X}$ for all $k \in \{1, 2, \dots, K\}$. With this background, asymptotic convergence to the global optimum(s) of each constitutive task k in a multitasking environment is said to occur if the following relation is satisfied,

$$\lim_{t \rightarrow \infty} \mathbb{E}[f_k(x)] = \lim_{t \rightarrow \infty} \int_{\mathbf{X}} f_k(x) \cdot p^k(x, t) \cdot dx = f_k^*, \forall k. \quad (3.2)$$

Loosely speaking, as time goes on in the optimization process, the distribution of the population must be gradually concentrated on only those points of the unified space that correspond to a global optimum.

For our theoretical deductions, we specifically consider the evolutionary multitasking algorithm to follow a (μ, λ) selection strategy whereby μ denotes the number of parents, and λ represents the number of offspring. Under this selection mechanism, the top μ individuals out of λ offspring (with $\mu < \lambda$), are selected for each task and taken as parents for the next iteration [67]. Note that the parent population of the k^{th} task is denoted as P^k . Further, the offspring population, of size $\lambda = N$, is denoted as P_c^k . Recall that the distribution of P_c^k is given by $q^k(x)$ as in Eq. (3.1). For now, no elitism is assumed to strengthen the implications of our result

- as randomized global search algorithms with elitism trivially possess asymptotic convergence properties.

To begin, let us define a parameter $\theta = \mu/\lambda$, such that $0 < \theta < 1$. Using the mixture distribution in Eq. (3.1), the (μ, λ) parent selection scheme can be modeled as [68],

$$p^k(x, t+1) = \begin{cases} \frac{q^k(x, t)}{\theta} & \text{if } f_k(x) \geq \beta^k(t+1) \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

where $\beta^k(t+1)$ represents a real number that satisfies,

$$\int_{f_k(x) \geq \beta^k(t+1)} q^k(x, t) \cdot dx = \theta. \quad (3.4)$$

Thus, the transition from $p^k(x, t)$ to $p^k(x, t+1)$ is based on Eqs. (3.1), (3.3) and (3.4).

Theorem 3.2.1. Given $p^k(x, 0)$ is positive and continuous in $\mathbf{X}$ for all k , and $N \rightarrow \infty$, the asymptotic convergence to the global optimum(s) of each constitutive task is guaranteed during multitasking if $\alpha_k > \theta$ in Eq. (3.1).

Proof. Our deductions follow a similar style as [68]. At any iteration $t > 0$, the μ parent samples associated with task k map to a set of objective values corresponding to f_k . Let this set be denoted as $F^k(t)$, and the infimum of the set as per Eq. (3.3) is $\beta^k(t)$, i.e., $\beta^k(t) = \inf F^k(t)$. In other words, we have,

$$p^k(x, t) = 0 \Leftrightarrow f_k(x) < \beta^k(t). \quad (3.5)$$

This implies that a point is selected if and only if its objective function is not less than $\beta^k(t)$. We shall first show that $\beta^k(t) < \beta^k(t+1)$. This condition provides the intuition that the *spread of the population must be contracting*, gradually zooming in on the most promising regions of the unified space.

Based on Eq. (3.4), while selecting parents $P^k(t)$ from offspring $P_c^k(t-1)$, $\beta^k(t)$ satisfies,

$$\int_{f_k(x) \geq \beta^k(t)} q^k(x, t-1) \cdot dx = \theta. \quad (3.6)$$

Notice that if the following inequality holds,

$$\int_{f_k(x) \geq \beta^k(t)} q^k(x, t) \cdot dx > \int_{f_k(x) \geq \beta^k(t+1)} q^k(x, t) \cdot dx, \quad (3.7)$$

then $\beta^k(t) < \beta^k(t+1)$. Invoking Eq. (3.1) for the left hand side of Eq. (3.7), we have,

$$\int_{f_k(x) \geq \beta^k(t)} q^k(x, t) \cdot dx \geq \int_{f_k(x) \geq \beta^k(t)} \alpha_k \cdot p^k(x, t) \cdot dx. \quad (3.8)$$

Moreover, Eq. (3.5) implies $\int_{f_k(x) \geq \beta^k(t)} p^k(x, t) \cdot dx = 1$, which on combining with Eq. (3.8) leads to the relation,

$$\int_{f_k(x) \geq \beta^k(t)} q^k(x, t) \cdot dx \geq \alpha_k. \quad (3.9)$$

Combining Eqs. (3.4), (3.7) and (3.9), we conclude that $\beta^k(t) < \beta^k(t+1)$ holds if $\alpha_k > \theta$. The relation $\beta^k(t) < \beta^k(t+1)$, and the fact that β^k can not be greater than f_k^* implies that there exists a limit $\lim_{t \rightarrow \infty} \beta^k(t) = f'_k$ such that $f'_k \leq f_k^*$. Hereafter, we present a proof that $f'_k = f_k^*$ via contradiction by first assuming,

$$f'_k < f_k^*. \quad (3.10)$$

Based on the proposed transition procedure from $p^k(x, t)$ to $p^k(x, t+1)$ and per Eqs. (3.1) and (3.3), note,

$$p^k(x, t) \geq p^k(x, 0) \left[\frac{\alpha_k}{\theta} \right]^t \quad \forall x : f_k(x) > f'_k. \quad (3.11)$$

Let $H = \{x | x \in \mathbf{X}, f_k(x) > f'_k\}$. Since $p^k(x, 0) > 0$ for any $x \in \mathbf{X}$, and considering $\frac{\alpha_k}{\theta} > 1$, we have,

$$\lim_{t \rightarrow \infty} p^k(x, t) = +\infty, \forall x \in H. \quad (3.12)$$

Noting that the Borel measure of H is positive, and applying Fatou's lemma [69] to get,

$$\lim_{t \rightarrow \infty} \int_H p^k(x, t) \cdot dx = +\infty, \quad (3.13)$$

it is clear that Eq. (3.13) contradicts the fact that $p^k(x, t)$ is a probability density function. Thus, the contradiction implies the conclusion,

$$\lim_{t \rightarrow \infty} \beta^k(t) = f_k^*. \quad (3.14)$$

□

3.3 Chapter Summary

This chapter presented an instantiation of an evolutionary multitask optimization framework from a probabilistic modelling perspective. Accordingly, it underscored how knowledge transfers can be actuated through inter-task interactions (i.e., via principled integration of probabilistic models drawn from different tasks). However, the threats of negative transfers (particularly between unrelated tasks) may still persist. Finally, a study of its convergence characteristics is conducted to theoretically validate that under appropriate assumptions, evolutionary multitasking indeed does not prevent convergence.

Chapter 4

Multifactorial Evolutionary Algorithm with Online Transfer Parameter Estimation: MFEA-II

Chapter 3 theoretically analyzed the effect of mixture distribution on the global convergence behaviour of probabilistic model based evolutionary multitasking frameworks. Under appropriate assumptions, it was shown that asymptotic global convergence properties of multitasking EAs continue to be preserved.

While multitasking solvers have led to recent success stories, a known shortcoming of existing methods is their inability to adapt the extent of transfer in a principled manner. Notably, in the absence of any prior knowledge about the relationships between distinct optimization tasks, a threat of predominantly negative (harmful) transfer prevails. In the context of probabilistic model based EAs, it thus becomes vital to facilitate the *optimal mixture* of the component probability densities drawn from different tasks.

To alleviate the threat of negative transfers plaguing the present day evolutionary multitasking frameworks, this chapter proposes a novel *data-driven* multitasking approach adept at *online* learning of *how much* knowledge to transfer across distinct optimization problems¹.

¹The work in this chapter has been published in [62]

It is noteworthy that the ideas described herein directly encompass a previously proposed *multifactorial evolutionary algorithm* (MFEA) for multitasking [21]. In comparison to the MFEA, the main contributions in this work are twofold:

- While the *mode* of transfer remains the same as MFEA, the parameter inducing the *extent* of transfers is modified to take the form of a symmetric matrix. This allows effective multitasking across more than two tasks with possibly diverse inter-task relationships.
- The transfer parameter matrix is continuously learned and adapted *online* based on the data generated during the course of the multitasking search. As the result of a theoretically principled learning paradigm, the need for extensive manual parameter tuning is avoided.

With this, the novel algorithm put forward in this chapter is labelled as MFEA-II.

The organization of this chapter is as follows. Section 4.1 presents an instantiation of a candidate evolutionary algorithm for multitask optimization, namely the MFEA. The theoretical foundations of the MFEA and its associated shortcomings are analyzed in Sections 4.2 and 4.3, respectively. Thereafter, the proposed online parameter estimation strategy for MFEA-II is proposed in Section 4.4. Experimental results and analyses on a series of discrete and continuous toy/benchmark examples are provided in Section 4.5. Section 4.6 concludes the chapter.

4.1 The basic multifactorial evolutionary algorithm

In this section, we present an overview of an instantiation of a crossover and mutation based EA for multitask optimization. Recently proposed, the so-called multifactorial evolutionary algorithm (MFEA) [21] leverages on the implicit parallelism of population-based search to solve multiple problems simultaneously with *omnidirectional* transfer of knowledge across them. Under the relatively strong restriction of *parent-centric* evolutionary operators, crossover and mutation based EAs can be thought of as having resemblance to stochastic sampling-based optimization algorithms [24]. To elaborate, parent-centric operators tend to bias the offspring to be

created nearer to the parents with high probability [70]. In the literature, there exists a family of operators that satisfy this assumption in the context of real-coded EAs. Common examples include the simulated binary crossover (SBX) [71], polynomial mutation (PM) [72] and Gaussian mutation with small variance [73], to name a few. In such settings, it is not unreasonable (as shown below in Section 4.1.1) to consider that the empirical density of the offspring population $P_c(t)$ tends to closely approximate that of the parent population $P(t)$; which can be expressed as $p_c(x, t) \approx p(x, t)$. While such an assumption will not necessarily hold under arbitrary evolutionary operators, it provides sufficient simplification for an intuitive theoretical analysis that builds on the result of Section 3.1. The practical implications of the same shall be demonstrated in the experimental sections of the subsequent chapters.

4.1.1 Preservation of parent distribution under parent-centric evolutionary operators

For simplicity, consider the probability distribution of a parent population to follow a multivariate Gaussian with mean m and covariance Σ . Additionally, let the mean and covariance of the offspring population be m_c and Σ_c respectively. It is noteworthy that parent-centric crossover operators already ensure $m_c = m$ [74]. In general, we may write $p_c(x, t) = p(x, t) + \delta_m$, where δ_m is a random vector with zero mean and covariance $\delta\Sigma$. Restricting the evolutionary operators to treat each variable independently (through variable-wise crossovers and/or mutations), we may write $\Sigma_c = \Sigma + \delta\Sigma$, where $\delta\Sigma$ is a diagonal matrix of the form $\delta\Sigma_{(i,i)} = \sigma_i^2$. Here σ_i^2 is the induced variance of the i^{th} variable during offspring creation. The specified high probability of creating offspring near the parents implies that $\sigma_i^2 \ll \Sigma_{(i,i)}$. Therefore, $\Sigma_c \approx \Sigma$, which suggests that the underlying distribution of the parent and offspring populations remain largely unaltered under variable-wise parent-centric operations.

4.1.2 Preliminaries of the MFEA

The MFEA employs a single population P of individuals to solve K optimization tasks $\{T_1, T_2, \dots, T_K\}$ simultaneously, where each task is seen as an added *factor*

influencing the evolution of the population. The subpopulation associated with the k^{th} task is denoted as P^k . Given this background, certain terminologies commonly encountered in the context of the MFEA are as follows.

- **Definition 1** (*Skill Factor*): The *skill factor* τ_i of the i^{th} individual is the one task, amongst all K tasks in the multitasking environment, with which the individual is associated.
- **Definition 2** (*Scalar Fitness*): The *scalar fitness* of the i^{th} individual in a multitasking environment is given by $\varphi_i = 1/r_{\tau_i}^i$, where $r_{\tau_i}^i$ is the rank of the i^{th} individual on task τ_i .

4.1.2.1 Overview of the MFEA

Algorithm 1 summarizes the general work flow of the present-day MFEA. The procedure evolves a population of $N \cdot K$ individuals encoded in a unified space $\mathbf{X}$. Specifically, each of the K tasks is allocated equal resource of N individuals in its subpopulation. During the evolutionary process, in addition to the standard *intra-task crossover* between parents of the same skill factor, *inter-task crossover* can also occur between candidate solutions associated with distinct optimization tasks, i.e., possessing different skill factors. It is this simple feature that opens the door for knowledge transfers through crossover-based genetic exchanges across tasks.

The extent of knowledge transfer in the MFEA is governed by a scalar (user-defined) parameter labeled as the *random mating probability* (rpm). To elaborate on the role of the rpm , we refer to Step 14 in Algorithm 1. If two individuals selected from the parent pool correspond to different tasks, they can undergo crossover under a probability specified by $rpm \in [0, 1]$. During inter-task crossover, the genetic material created for one optimization task gets copied into the solution chromosome of an individual associated with a different task. As a result, implicit transfer of information is actualized across tasks. Clearly, if rpm is large, the transfer is more frequent than otherwise.

Algorithm 1 Pseudocode of a basic MFEA

```

1 Randomly sample  $N \cdot K$  individuals in  $\mathbf{X}$  to form initial population  $P(0)$  ;
2 for every individual  $p_i$  in  $P(0)$  do
3     Assign skill factor  $\tau_i = \text{mod}(i, K) + 1$ , for the case of  $K$  tasks ;
4     Evaluate  $p_i$  for task  $\tau_i$  only ;
5 Set  $t = 1$  ;
6 while stopping conditions are not satisfied do
7      $P(t) \leftarrow$  Select top  $N \cdot K/2$  ( $\leq 50\%$ ) of generated individuals (based on scalar fitness  $\varphi$ );
8     Configure offspring population  $P_c(t) = \emptyset$ ;
9     while offspring generated for each task  $< N$  do
10        Sample two individuals uniformly at random (without replacement):  $x_i$  and  $x_j$  from  $P(t)$ ;
11        if  $\tau_i == \tau_j$  then
12             $[x_a, x_b] \leftarrow$  Intra-task crossover between  $x_i$  and  $x_j$  ;
13            Assign offspring  $x_a$  and  $x_b$  skill factor  $\tau_i$ ;
14        else if  $\text{rand} \leq \text{rmp}$  then
15             $[x_a, x_b] \leftarrow$  Inter-task crossover between  $x_i$  and  $x_j$  ;
16            Each offspring is randomly assigned skill factor  $\tau_i$  or  $\tau_j$ ;
17        else
18             $[x_a] \leftarrow$  local variation (mutation) of  $x_i$  ;
19            Assign offspring  $x_a$  skill factor  $\tau_i$ ;
20             $[x_b] \leftarrow$  local variation (mutation) of  $x_j$ ;
21            Assign offspring  $x_b$  skill factor  $\tau_j$ ;
22        Evaluate  $[x_a, x_b]$  for their assigned skill factors only;
23         $P_c(t) = P_c(t) \cup [x_a, x_b]$ 
24     $t = t + 1$ ;

```

4.2 Theoretical Foundations of the MFEA

At any generation t of the MFEA, the probability density function $p(x, t)$ of the parent population $P(t)$ is essentially a mixture of K probability densities $p^k(x, t)$ representing the K tasks in the multitask setting. Herein, the k^{th} mixture component corresponds to subpopulation $P^k(t) \subset P(t)$. Importantly, while the *overall* offspring population distribution may satisfy $p_c(x, t) \approx p(x, t)$ under appropriate assumptions on the evolutionary operators, the inter-task crossovers indicate that distribution $p_c^k(x, t)$ need not necessarily be equivalent to $p^k(x, t)$. In particular, we have the following result on the mixture distribution occurring in the MFEA.

Lemma 4.2.1. Under the stated assumptions of parent-centric evolutionary operators, the offspring mixture distribution $p_c^k(x, t)$ of the k^{th} task in the MFEA can be expressed as,

$$p_c^k(x, t) = \left[1 - \frac{0.5 \cdot (K-1) \cdot rmp}{K}\right] \cdot p^k(x, t) + \sum_{j \neq k} \frac{0.5 \cdot rmp}{K} \cdot p^j(x, t). \quad (4.1)$$

Proof. Refer to Appendix A.1. □

Note the resemblance between Eq. (4.1) and Eq. (3.1), with $p_c^k(x, t)$ being analogous to $q^k(x, t)$.

4.2.0.1 Does the MFEA Converge?

Herein, we prove that the MFEA asymptotically converges to the global optimum(s) of each constitutive task regardless of the choice of rmp .

Theorem 4.2.2. Under the large population assumption (i.e., $N \rightarrow \infty$), the MFEA with parent-centric evolutionary operators and (μ, λ) selection possesses asymptotic global convergence properties across all tasks, given $\theta (= \mu/\lambda) \leq 0.5$.

Proof. According to Theorem 3.2.1, for asymptotic global convergence of the k^{th} task in the multitasking environment, it suffices to ensure that $\alpha_k > \theta$. With reference to Lemma 4.2.1, the mixture coefficient α_k corresponds to,

$$\alpha_k = 1 - \frac{0.5 \cdot (K-1) \cdot rmp}{K}. \quad (4.2)$$

Since $0 \leq rmp \leq 1$, it follows that $0.5 < \alpha_k \leq 1$, and thus α_k is indeed greater than θ . The above holds for all k . □

4.3 Susceptibility to Negative Inter-Task Interactions in Evolutionary Multitasking

As is clear from Algorithm 1, the performance of MFEA relies on the choice of rmp . Given the lack of prior knowledge about the inter-task relationships, tuning an appropriate rmp (via trial and error) thus becomes burdensome. Essentially, poor prescriptions of rmp risk the possibility of either predominantly negative inter-task interactions, or the potential loss of fruitful knowledge exchange.

In this section, we study the possible deleterious effects of negative inter-task interactions in evolutionary multitasking. In particular, we analyze the *rate of convergence* of the MFEA under the adverse case of no complementarity between constitutive tasks in a multitasking environment. A direct approach in this regard is to measure the evolution of the *spread* of subpopulations from generation t to $t + 1$ as $\beta^k(t + 1) - \beta^k(t)$. Clearly, a large difference indicates a faster rate of convergence.

4.3.0.1 Analyzing the Rate of Convergence of the MFEA

Without loss of generality, we consider only the k^{th} optimization task as the same arguments hold across all K tasks. Given $N \rightarrow \infty$, we simulate the adverse scenario of no complementarity between constitutive tasks, at some generation t (> 0) of the MFEA, by assuming that wherever $p^k(x, t) > 0$, we have $p^j(x, t) = 0$, for all $j \neq k$. By extension, the following also holds,

$$f_k(x) \geq \beta^k(t) \Rightarrow p^j(x, t) = 0, \text{ for all } j \neq k. \quad (4.3)$$

Qualitatively speaking, the optimum solution(s) of the k^{th} task do not lie in the neighborhood of the optimum solution(s) of any other task being optimized in the same multitasking environment. With this, the effects of negative interactions between optimization tasks can be analyzed by comparing the expected progress (from generation t to $t + 1$) of a traditional single-tasking solver (denoted by ST) against its multitasking counterpart. We make the simplification that the population distribution at the t^{th} generation is identical for both solvers; i.e., $p^{k,ST}(x, t) \approx p^k(x, t)$ and $\beta^{k,ST}(t) \approx \beta^k(t)$.

Notably, due to the absence of inter-task crossovers in single-tasking, the offspring distribution $p_c^{k,ST}(x, t)$ is *not* a mixture model (unlike MFEA). Further, Section 4.1.1 shows that *parent-centric* operators result in $p_c^{k,ST}(x, t) \approx p^k(x, t)$. Thus, for the (μ, λ) selection with $\theta \leq 0.50$, $\beta^{k,ST}(t + 1)$ satisfies,

$$\int_{f_k(x) \geq \beta^{k,ST}(t+1)} p_c^{k,ST}(x, t) \cdot dx \approx \int_{f_k(x) \geq \beta^{k,ST}(t+1)} p^k(x, t) \cdot dx = \theta (= 0.5 - \epsilon), \text{ where } \epsilon \in [0, 0.5). \quad (4.4)$$

On the other hand, since $\beta^k(t + 1) > \beta^k(t)$ (Theorem 3.2.1), Eq. (4.1) together with Eq. (4.3) imply that $\beta^k(t + 1)$ must satisfy,

$$\int_{f_k(x) \geq \beta^k(t+1)} p_c^k(x, t) \cdot dx = \int_{f_k(x) \geq \beta^k(t+1)} \left[1 - \frac{0.5 \cdot (K-1) \cdot rmp}{K} \right] \cdot p^k(x, t) \cdot dx = 0.5 - \epsilon. \quad (4.5)$$

For *strictly positive* values of rmp , we have the inequality $0.5 < [1 - \frac{0.5 \cdot (K-1) \cdot rmp}{K}] < 1$. Then, combining Eqs. (4.4) and (4.5) leads to,

$$\int_{f_k(x) \geq \beta^k(t+1)} p^k(x, t) \cdot dx = [0.5 - \epsilon] \cdot \left[1 - \frac{0.5 \cdot (K-1) \cdot rmp}{K} \right]^{-1} > \int_{f_k(x) \geq \beta^{k,ST}(t+1)} p^k(x, t) \cdot dx, \quad (4.6)$$

which indicates $\beta^{k,ST}(t+1) > \beta^k(t+1)$. Therefore,

$$\beta^{k,ST}(t+1) - \beta^{k,ST}(t) > \beta^k(t+1) - \beta^k(t). \quad (4.7)$$

Eq. (4.7) suggests that in the case where there is no complementarity between tasks, single-tasking may lead to faster convergence characteristics than multitasking. Further, the result of Eq. (4.6) shows that the reduction in the convergence rate of MFEA depends on the chosen rmp .

4.4 Transfer Parameter Estimation via Optimal Mixture Modeling

In this section, we propose an online rmp estimation technique that theoretically guarantees to minimize the negative (harmful) interactions between distinct optimization tasks.

From the convergence rate analysis of MFEA we infer that negative transfers in multitasking can be suppressed by forcing $p_c^k(x, t)$ to replicate $p^k(x, t)$ as closely as possible. Based on Lemma 4.2.1, this can easily be accomplished by assigning $rmp=0$. However, such a naive strategy prohibits inter-task crossovers and, as a result, prohibits any knowledge transfer from occurring (see Algorithm 1). With this in mind, our primary goal is to put forward a strategy that promotes knowledge transfer ($rmp > 0$), while simultaneously mitigating negative inter-task interactions. To this end, we propose a novel data-driven approach for online learning of rmp values that lead to an optimal mixture distribution in Eq. (4.1).

It is worth mentioning that hereafter, RMP is no longer a scalar parameter, but takes the form of a symmetric $K \times K$ matrix (given K optimization tasks) of pairwise values, represented as,

$$RMP = \begin{bmatrix} rmp_{1,1} & rmp_{1,2} & \cdot & \cdot \\ rmp_{2,1} & rmp_{2,2} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \quad (4.8)$$

where $rmp_{j,k} = rmp_{k,j}$ captures the synergy (or the extent of transfer) between the j^{th} and k^{th} tasks. Accordingly, it is naturally enforced that $rmp_{j,j} = 1, \forall j$. *This enhancement offers the distinct advantage that in many practical scenarios, where the complementarity between tasks may not be uniform across different task-pairs, our proposed method shall be capable of capturing non-uniform inter-task synergies.*

At generation t , consider $g^k(x, t)$ to be a probabilistic model (estimate) of the true distribution $p^k(x, t)$ of any k^{th} task of interest. Herein, $g^k(x, t)$ is built from the dataset of subpopulation $P^k(t)$. By replacing the true density functions with the learned probabilistic models, and using the elements of the RMP matrix instead of the scalar rmp , Eq. (4.1) can be rewritten as,

$$g_c^k(x, t) = \left[1 - \frac{0.5}{K} \cdot \sum_{k \neq j} rmp_{k,j} \right] \cdot g^k(x, t) + \frac{0.5}{K} \sum_{j \neq k} rmp_{k,j} \cdot g^j(x, t). \quad (4.9)$$

wherein $g_c^k(x, t)$ is a probabilistic mixture model approximating the distribution of the offspring population: $p_c^k(x, t)$.

Thus, in order to force $p_c^k(x, t)$ to closely replicate $p^k(x, t)$, it is equivalent to learning the RMP matrix such that the offspring probabilistic model $g_c^k(x, t)$ accurately models the parent distribution $p^k(x, t)$, for all $k \in \{1, 2, \dots, K\}$.

4.4.1 Learning the RMP Matrix Online

The first step is to build probabilistic models $g^k(x, t)$ on each subpopulation $P^k(t)$, for all $k \in \{1, 2, \dots, K\}$. From here onwards, consider the parameter $\theta (= \mu/\lambda)$ to be 0.5. Then, given the $N/2$ parent solutions in each subpopulation $P^k(t)$ at some iteration t , we propose to learn the RMP matrix by maximizing the following

log-likelihood function:

$$\max_{RMP} \sum_{k=1}^K \sum_{i=1}^{N/2} \log g_c^k(x_{ik}, t). \quad (4.10)$$

where x_{ik} is the i^{th} sample (individual) in the dataset of $P^k(t)$.

We claim that the data-driven learning paradigm of Eq. (4.10) serves to reduce the “gap” between $g_c^k(x, t)$ and $p^k(x, t)$, measured over all k . In this regard, we note that a commonly encountered quantifier of distribution gaps is the Kullback Leibler (KL) divergence [75]. Specifically, the KL divergence provides the amount of information lost when an arbitrary density function g is used to approximate another density p . The mathematical equation corresponding to this definition is,

$$KL(p||g) = \int_{\mathbf{X}} p(x) \cdot [\log p(x) - \log g(x)] \cdot dx. \quad (4.11)$$

In what follows, we prove our claim in the context of the KL divergence.

Theorem 4.4.1. Maximizing the likelihood function in Eq. (4.10) is equivalent to minimizing the gap between the probability densities $g_c^k(x, t)$ and $p^k(x, t)$ averaged across all tasks $\{T_1, T_2, \dots, T_k, \dots, T_K\}$.

Proof. $\sum_{i=1}^{N/2} \log g_c^k(x_{ik}, t)$ can also be written as,

$$(N/2) \frac{\sum_{i=1}^{N/2} \log g_c^k(x_{ik}, t)}{N/2} = \frac{N}{2} \mathbb{E} [\log g_c^k(x_{ik}, t)], \quad (4.12)$$

where $\mathbb{E}[\cdot]$ is the expected value. Under the assumption of large N , the empirical probability density function induced by the samples in $P^k(t)$ converges to the underlying probability density $p^k(x, t)$ [66]. Thus, we have,

$$\mathbb{E} [\log g_c^k(x, t)] = \int_{\mathbf{X}} p^k(x, t) \cdot \log g_c^k(x, t) \cdot dx. \quad (4.13)$$

Accordingly, we can restate the mathematical formulation of Eq. (4.10) as,

$$\max_{RMP} \sum_{k=1}^K \int_{\mathbf{X}} p^k(x, t) \cdot \log g_c^k(x, t) \cdot dx. \quad (4.14)$$

Note that maximizing Eq. (4.14) is equivalent to the following minimization problem, given $p^k(x, t)$ to be fixed,

$$\min_{RMP} \sum_{k=1}^K \int_{\mathbf{X}} p^k(x, t) \cdot [\log p^k(x, t) - \log g_c^k(x, t)] \cdot dx. \quad (4.15)$$

Referring to Eq. (4.11), we can express Eq. (4.15) in terms of the *KL* divergence as:

$$\min_{RMP} \sum_{k=1}^K KL(p^k(x, t) || g_c^k(x, t)). \quad (4.16)$$

In other words, the mathematical program of Eq. (4.10) indeed minimizes the gap (given by the *KL* divergence) between the mixture models $g_c^k(x, t)$ and the true density functions $p^k(x, t)$, averaged across all K optimization tasks. $\square$

Theorem 4.4.1 highlights that through the proposed data-driven learning of the *RMP* matrix, the various tasks in a multitask setting can be directly brought to bear upon one another, while facilitating the suppression of negative transfers. In other words, the method results in a theoretically principled adaptive online transfer parameter estimation scheme.

With regard to the online learning paradigm, an important matter of “practical” interest is the choice of probabilistic model type for building $g^k(x, t)$, for all $P^k(t) \in \{P^1(t), P^2(t), \dots, P^K(t)\}$. To this end, observe that choosing a complex probabilistic model that may tend to overfit the (in practice reasonably small) dataset $P^k(t)$, potentially leads to the cancellation of inter-task transfers. This is in fact a well documented phenomenon even in the classical transfer/multitask learning literature, which is typically overcome by internal cross-validation steps or noise incorporation for preventing overfitting [23, 76]. Along similar lines, in this chapter we propose to use simple and fast probabilistic modeling approaches (e.g., univariate marginal distributions) to help prevent the overfitting issue; which in turn allows the various other models in the multitask setting to be mobilized for filling in distribution gaps.

4.4.2 Main Ingredients of MFEA-II

This section introduces the enhanced MFEA-II incorporated with the proposed online *RMP* matrix learning scheme. Algorithm 2 provides the pseudocode of the *RMP* learning procedure (Section 4.4.1). The general steps associated with MFEA-II are illustrated by Figure 4.1 (right side).

Algorithm 2 Online *RMP* learning

Build simple and fast models $g^k(x, t)$ on each subpopulation $P^k(t)$, $\forall k \in \{1, 2, \dots, K\}$.

Optimize mixture models: $\max_{RMP} \sum_{k=1}^K \sum_{i=1}^{N/2} \log g_c^k(x_{i,k}, t)$.

FIGURE 4.1: **Left:** General framework of the existing MFEA. Notice that MFEA employs an *offline* scalar *rmp* assignment. **Right:** MFEA-II with the added *online* *RMP* matrix learning module.

As depicted, the initial phases of MFEA-II are basically similar to the existing MFEA. The distinguishing facet of the MFEA-II is the incorporation of the online *RMP* learning module within the overall optimization algorithm. It is important to highlight that given built probabilistic models $g^k(x, t)$, the mathematical problem of Eq. (4.10) is *convex upward*. As a result, it is found to be solved to optimality at little computational overhead using classical optimizers.

To summarize, in MFEA-II, the offspring population $P_c(t)$ is generated based on the learned *RMP* matrix, which automatically ascertains the extent of genetic transfer across distinct tasks. Note that the main modification compared to the original MFEA occurs during inter-task crossover in steps 14 – 21 of Algorithm 1 (Chapter 3). The modified steps of MFEA-II are summarized in Algorithm 3, wherein solutions corresponding to different tasks undergo crossover based on the learned *RMP* matrix. Further, note that during inter-task crossovers, a parent-centric operator (without additional uniform crossover-like variable swap) is applied so as to ensure that this key assumption of our theoretical deduction does indeed hold in the present instantiation of MFEA-II. Each generated offspring also undergoes small parent-centric mutation. It is worthwhile to mention that if the learned *RMP* matrix is full of 0's, the MFEA-II works similar to multiple canonical EAs executing single-tasking in parallel. The ability of the MFEA-II to identify discrepancies and interpret similarities across multiple tasks on the fly is exemplified through subsequent experimental studies².

Algorithm 3 Inter-task crossover in MFEA-II

```

Randomly sample two individuals  $x_i$  and  $x_j$  from  $P(t)$ ;
if  $\tau_i \neq \tau_j$  then
    if  $\text{rand} \leq \text{rmp}_{\tau_i, \tau_j}$  then
         $[x_a, x_b] \leftarrow \text{Inter-task crossover between } x_i \text{ and } x_j$   Each offspring is randomly assigned skill factor  $\tau_i$  or  $\tau_j$ 
    end
    else
        Randomly select  $x'_i$  with skill factor  $\tau_i$  ;
         $[x_a] \leftarrow \text{Intra-task crossover between } x_i \text{ and } x'_i$   Assign offspring  $x_a$  skill factor  $\tau_i$  ;

        Randomly select  $x'_j$  with skill factor  $\tau_j$  ;
         $[x_b] \leftarrow \text{Intra-task crossover between } x_j \text{ and } x'_j$   Assign offspring  $x_b$  skill factor  $\tau_j$  ;
    end
end
    
```

²Source codes are available via <http://www.bdsc.site/websites/MTO/index.html>

4.5 Experimental Study

In this section, numerical illustrations are presented that showcase the efficacy of the proposed online parameter estimation scheme of the MFEA-II. Herein, we conduct experimental studies on a variety of problems, ranging from discrete to continuous benchmark optimization. For a proof of concept analysis, the performance of the MFEA-II is compared against the basic counterparts, i.e, the present day multifactorial evolutionary algorithm (denoted as MFEA-I) [21] and the standalone canonical EA (CEA). Further, the practicality of MFEA-II on increasingly challenging variants of the controller design task (benchmarked against state-of-the-art optimizers) shall be highlighted in Chapter 6.

4.5.1 Experimental Configuration

The experimental setup is outlined in what follows. The population size N (per task) is kept the same for all algorithms, ensuring consistency. Specifically, if a single-tasking algorithm utilizes a population size N for each of the K tasks, then a multitasking algorithm tackling the same K tasks is assigned a population size of $N \cdot K$. The general algorithmic settings for all the problems are:

1. Continuized unified representation with range $[0, 1]^{D_{unified}}$;
2. Evolutionary operators for MFEA-II, MFEA-I and CEA:
 - (a) SBX crossover with probability $(p_c) = 1$ and distribution index $\eta_c = 15$;
 - (b) Polynomial mutation with probability $(p_m) = 1/d$ and distribution index $\eta_m = 15$;
3. Probabilistic model for MFEA-II: variable-wise (marginal) normal distribution;
4. For the pedagogical case study:
 - (a) Population size (N): 100;
 - (b) Maximum function evaluations: 15,000;
5. For the synthetic continuous benchmark set:

- (a) Population size (N): 50
- (b) Maximum function evaluations: 500,000;
- 6. *rmp*: Learned online for MFEA-II. For MFEA-I:
 - (a) 0.3 for the pedagogical study and the synthetic continuous benchmarks (as per [2]);

Note that we utilize the SBX and polynomial mutation operators since they agree with the underlying assumptions of the theoretical derivations in Section 4.1.

4.5.2 Pedagogical Case Study

We begin by considering a set of binary optimization problems as toy examples. While decoding candidate solutions from the continuous unified space ($\mathbf{X} = [0, 1]^{D_{unified}}$) to the discrete/binary space, a variable is assigned a value of 0 if its encoded counterpart is less than or equal to ≤ 0.5 . Else, it is assigned a value of 1.

For this particular study, three popular binary problems from literature are studied, namely, *onemax*, *zeromax* and the highly deceptive *trap-5* problem [77]. While the optimum of *onemax* is the string of all 1's, that of *zeromax* is the string of all 0's. In *trap-5*, the string is first partitioned into consecutive groups of 5 non-overlapping bits. Further, a 5-bit trap function is applied to each of the groups, and the contribution of each group towards the combined objective function is given as follows.

$$trap_5(u) = \begin{cases} 5 & \text{if } u = 5 \\ (4 - u) & \text{if } u < 5 \end{cases} \quad (4.17)$$

where u is the number of 1's in the group. Observe there exists only one global optimum of the *trap-5* problem, when all the bits in the D -dimensional input string are 1's, and $(2^{\frac{D}{5}} - 1)$ other local optima.

The rationale behind the choice of the above set of problems is that they provide an ideal platform to showcase the potential advantages of the proposed online parameter estimation scheme of MFEA-II. To elaborate, the optimum solution of *onemax* is seen to intersect with the global optimum of *trap-5*, implying that the crossover-based exchange of genetic material across tasks is indeed a suitable mode

FIGURE 4.2: **Left:** Pairwise rmp 's learned between distinct tasks over successive generations. The shaded regions span one half standard deviation on either side of the mean. **Right:** Convergence trends (averaged over 30 independent runs) achieved by all the algorithms on the trap-5 problem. As shown, both multitasking variants are superior to CEA, with MFEA-II being the most dominant.

of transfer. However, due to the presence of a large number of local optimum in trap-5, it is deemed that the uninhibited inter-task crossovers between the two tasks is unfavorable, since this can potentially hinder the convergence characteristics of onemax. Additionally, the zeromax problem is clearly conflicting with onemax in terms of solution similarity, and by extension, with the trap-5 problem as well. Nonetheless, the worst (highly deceptive) local optimum of trap-5 corresponds to the global optimum of zeromax. Therefore, it is contended that solving the three tasks concurrently using the existing MFEA-I (with manually specified scalar rmp), would be susceptible to negative transfers.

In this study, we solve dimensionality $D = 200$ variants of each of the three problems. The multitasking variants MFEA-I and MFEA-II tackle all the three tasks simultaneously, while CEA solves each in isolation. A summary of the experimental results averaged over 30 independent runs are contained in Figure 4.2. Note that a 3×3 symmetrical RMP matrix is learned for MFEA-II, capturing the pairwise inter-task similarities over successive generations. The learned RMP matrix elements are presented in Figure 4.2a.

Figure 4.2b shows the convergence trends of the highly deceptive trap-5 problem. For brevity, the convergence trends of one-max and zero-max problems are omitted.

In general, it can be observed that the scope for genetic transfer in both multitasking variants (MFEA-I and MFEA-II) has been beneficial, while the single-tasking CEA gets constantly trapped in the deceptive local optimum. Further, upon a closer look at Figure 4.2b, it can be seen that the convergence behavior of MFEA-I has been fairly inferior to that of MFEA-II. Notice that MFEA-I does not always converge to the global optimum. This is primarily due to the uninhabited exchange of genetic materials in the MFEA-I, making it vulnerable to negative transfers. However, this is clearly not the case with MFEA-II, which exploits the relationships learned from task datasets to optimally mandate knowledge transfers (at different stages of the optimization phase). As is depicted by Figure 4.2a, during the initial stages, when the transfer across the three tasks is unlikely to be helpful, the learned *RMP* matrix values are indeed low. Progressively, once high quality genetic material is available for transfer from onemax to the trap-5 problem (around after 50 generations), the online parameter estimation scheme starts prescribing higher values of the corresponding inter-task *rmP*. As indicated by the high *rmP* values, the positive transfers from onemax guide the trap-5 to consistently (and quickly) converge to the global optimum. We also note that relatively low to medium *rmP* values are learned between zeromax and trap-5, implying that the search is to some extent drawn towards the highly deceptive local optimum of trap-5 (under the influence of zeromax). In contrast, negligible (near zero) *rmP* values are learned between conflicting problem pairs onemax and zeromax throughout the evolutionary search. As a result, the MFEA-II successfully identifies and exploits positive transfer (when available), while circumventing the negative exchanges plaguing MFEA-I; see Figure 4.2b.

Further analysis of the performance of MFEA-II on various other toy examples is provided in the following section.

4.5.3 Further Experiments

Herein, we extend our experiments for the pedagogical case study. In what follows, we provide numerical illustrations for a series of binary optimization problems. For the purpose of decoding candidate solutions from the continuous unified space ($\mathbf{X} = [0, 1]^{D_{unified}}$) to the respective discrete/binary representations, a variable (of

an encoded solution) is assigned a value of 0 if its value in the unified space is less than or equal to 0.5. Else, it is assigned a value of 1.

4.5.3.1 Case 1: Trap-5 + hamming distance minimization (HDM) problems

In this particular study, we conduct experiments to solve the trap-5 problem alongside a series of hamming distance minimization (HDM) binary problems using MFEA-II. Herein, we generate a range of target strings for the HDM problems by sampling them from a Bernoulli parameter ρ , whereby a single bit takes the value 1 with probability ρ (and the value 0 with probability $1 - \rho$). For each successive experiment, the Bernoulli parameter is gradually decremented from $\rho = 1.00$ to $\rho = 0.00$. It is noteworthy that if ρ is higher, the HDM problems behave similar to the onemax problem and thus bear higher degrees of intersection (overlap), in terms of the global optimum, with the trap-5 problem. In contrast, if ρ is lower, the HDM problems behave more like the zeromax problem. Hence, the global optimum of these HDM problems are further away from the global optimum of trap-5. For this set of experiments, we solve 100 dimensional variants of the trap-5 and the HDM problems.

FIGURE 4.3: Trap-5 convergence trends.

The convergence trends for trap-5 are shown in Figure 4.3 (results are averaged over 30 independent runs). From Figure 4.3, it can be seen that the performance

of MFEA-II is superior to CEA when multitasked with HDM problems sampled from higher ρ 's (e.g. $\rho = 1.00$ to $\rho = 0.92$). Under the positive influence of genetic transfers from these HDM tasks, the MFEA-II is unlikely to get trapped in the deceptive local optimum of trap-5. However, as ρ decreases, the convergence trends of MFEA-II gradually drops and approaches that of the CEA (i.e., the MFEA-II performs identical to the single tasking CEA). Most importantly, no negative transfer is observed even in the extreme case of multitasking trap-5 alongside a HDM problem with $\rho = 0.00$. Further, the learned rmp trends in Figures 4.4 and 4.5 reveal that MFEA-II is adept at capturing the inter-task relationships online and adapts genetic transfers in a manner that leverages on positive transfers between overlapping tasks (dictated by higher ρ 's) while suppressing any deleterious effects of negative exchanges otherwise. Notice that for distant (unrelated) tasks (e.g. $\rho = 0.80$), the rmp values learned (throughout the multitasking search) are near zero, implying that the amount of resources wasted in transferring and evaluating solutions across unrelated tasks is negligible.

FIGURE 4.4: Learned rmp trends across generations. The shaded regions span one-half standard deviation on either side of the mean.

4.5.3.2 Case 2: Inverted trap-5 + HDM problems

In this case study, we invert the roles of zeros and ones in the definition of the original trap-5 function such that the global optimum is changed to a string of all zero bits. The experimental setup is similar to Section 4.5.3.1 above.

FIGURE 4.5: Average learned *rmp* trend between trap-5 and a series of HDM problems dictated by ρ . The error bars span one-half standard deviation on either side of the mean.

FIGURE 4.6: Inverse trap-5 convergence trends.

In contrast to trap-5, the HDM problems sampled from lower ρ 's (e.g. $\rho = 0.00$ to $\rho = 0.08$) would, herein, induce (higher) positive influence towards improving the search of the inverted trap-5 problem - showing greater potential for fruitful genetic transfers as demonstrated by Figure 4.6. The extent of genetic transfers between the inverted trap-5 problem and a series of HDM problems (with varying ρ 's) is mandated by the learned *rmp* trend in Figure 4.7.

FIGURE 4.7: Average learned rmp trend between the inverse trap-5 and a series of HDM problems dictated by ρ . The error bars span one-half standard deviation on either side of the mean.

4.5.3.3 Case 3: K randomly generated HDM tasks

In this study, we consider solving K tasks, each minimizing the hamming distance to a given target string generated at random ($\rho = 0.5$). Herein, we set $K = 10$ and the dimensionality of each HDM task is set to $D = 50$. Figure 4.8 shows the

FIGURE 4.8: Averaged performance of MFEA-II, MFEA-I and CEA on 10 randomly generated HDM tasks.

convergence trends averaged across all the 10 tasks. It can be observed the MFEA-II and the single tasking CEA perform identical to each other. This is primarily because the randomly generated tasks tend to bear low degrees of intersection of global optima. As such, their population datasets may not overlap as the search progresses and genetic transfers could in fact be detrimental. While MFEA-II prevents the harmful exchange of genetic material, we observe that MFEA-I is plagued by significant negative transfers. The convergence trend from Figure 4.8 shows that the lack of an online transfer parameter learning strategy in MFEA-I hampers its performance on the 10 HDM tasks. The ability of MFEA-II to overcome this issue marks the key contribution of the present chapter.

Figure 4.9 shows a few representative *rmf* trends learned between pairs of randomly generated tasks. As is revealed by the relatively low (near zero) *rmf* trends, the learning module of MFEA-II is able to figure out that there is no utility for genetic transfers herein, thereby ensuring that the convergence characteristics are not hampered due to harmful transfers.

FIGURE 4.9: Few representative *rmf* curves between pairs of randomly generated HDM tasks. Other inter-task *rmf* values showcase similar trends.

4.5.3.4 Case 4: LeadingOnes + TrailingOnes problem

For this particular study, we consider solving the LeadingOnes and TrailingOnes problems simultaneously using MFEA-II. The LeadingOnes problem essentially aims to maximize the number of consecutive 1-bits counting from the left of a

solution string. In contrast, the TrailingOnes problem starts the search by scanning the solution from the right towards the left. Both the problems share the same global optimum (i.e., the string with all 1's). For this experimental study, we set the problem dimensionality and the population size to $D = 50$ and $N = 150$ for each of the tasks, respectively.

FIGURE 4.10: Learned *rmp* trend between LeadingOnes and TrailingOnes problems over successive generations. The shaded regions span one-half standard deviation on either side of the mean.

FIGURE 4.11: Convergence trends of MFEA-II and MFEA-I.

Notably, while the problems do share a common global optimum, transfers in the early stage of the optimization phase is unlikely to be beneficial since each of

the tasks begin the search from opposite ends of the search space. In particular, Figure 4.10 demonstrates that MFEA-II is able to identify this and prescribes lower (near zero) *rmp* values during the initial stages. As the search progresses and the tasks becomes more similar, MFEA-II gradually learns higher *rmp* values in order to facilitate fruitful knowledge transfers. As shown by Figure 4.11, the transfer adaptation strategy of MFEA-II enables both the tasks to progress towards the common global optimum. In contrast, the convergence trends of MFEA-I are significantly hampered due to continuous (unregulated) inter-task genetic transfers.

4.5.4 Synthetic continuous benchmark functions

In the next set of examples, we showcase the *rmp* learned given *known* inter-task relationships between task pairs. To this end, we utilize some of the CEC composite minimization benchmark functions comprising of high, medium and low inter-task similarities [2]. The inter-task similarity measure between a pair of tasks

TABLE 4.1: Benchmark problem sets [2]

Category	Task	D	Landscape	R_s
1 CI+HS	Griewank (T_1)	50	multimodal, nonseparable	1.0000
	Rastrigin (T_2)	50	multimodal, nonseparable	
2 CI+MS	Ackley (T_1)	50	multimodal, nonseparable	0.2261
	Rastrigin (T_2)	50	multimodal, nonseparable	
3 CI+LS	Ackley (T_1)	50	multimodal, nonseparable	0.0002
	Schwefel (T_2)	50	multimodal, separable	
4 NI+HS	Rosenbrock (T_1)	50	multimodal, nonseparable	0.9434
	Rastrigin (T_2)	50	multimodal, nonseparable	
5 NI+MS	Griewank (T_1)	50	multimodal, nonseparable	0.3669
	Weierstrass (T_2)	50	multimodal, nonseparable	

(R_s) is based on the Spearman’s rank correlation between their respective function landscapes. Accordingly, the problem pairs with $R_s < 0.2$ are considered to have low similarity (LS), problem pairs with $0.2 \leq R_s < 0.8$ are defined as medium similarity (MS), and pairs with $R_s \geq 0.8$ imply high inter-task similarity (HS). Further, the global optima of each of the task-pairs may *completely intersect* in the unified search space, denoted as CI, or may *not intersect*, denoted as NI. The dimensions of each of the tasks is represented by D in Table 4.1.

To begin with, we analyze Table 4.2 containing the numerical results achieved by MFEA-II, MFEA-I and CEA on the composite benchmark problems. A task is

assumed to be solved if it reaches an objective function value within a convergence tolerance threshold of $1e - 04$, hence recorded as 0.00.

TABLE 4.2: Comparison of MFEA-II with MFEA-I and CEA on the continuous minimization benchmark problem sets. The mean and standard deviations of 30 independent runs are provided. Boldface entries marked with * are best and significantly the best (by Wilcoxon test with significance level 0.05). Best but not significantly the best entries are only boldface.

Problem set	MFEA-II		MFEA-I		CEA	
	T_1	T_2	T_1	T_2	T_1	T_2
1 CI+HS	0.00	0.00	0.00	0.00	0.00	82.35
	0.00	0.00	0.00	0.00	0.00	11.51
2 CI+MS	0.00	0.00*	0.00	5.57	18.82	82.17
	0.00	0.00	0.00	7.26	0.92	12.17
3 CI+LS	19.98	1.67E+03*	20.00	3.60e+03	20.01	1.94E+03
	0.04	5.18E+02	0.002	7.26E+02	0.01	4.34E+02
4 NI+HS	55.42	4.08*	58.51	13.10	65.99	83.89
	29.91	9.73	47.99	34.33	32.52	13.06
5 NI+MS	0.00	20.61	0.00	21.08	0.00	31.52
	0.00	1.19	0.00	1.72	0.00	4.96

It is seen that MFEA-II demonstrates consistently superior performance overall. Not only has MFEA-II performed generally better in the cases of high similarities, it has also outperformed MFEA-I and CEA for the problem pair with low inter-task similarity (CI+LS). In particular, it can be observed that MFEA-II has improved the performance of MFEA-I on T_2 of CI+LS. This is because the lack of inter-task similarities makes the MFEA-I susceptible to predominantly negative transfer of genetic materials. On the contrary, MFEA-II dynamically modulates the extent of genetic transfers in order to minimize such harmful negative inter-task interactions. Most importantly, the same performance benefits continues to hold for the no intersection (NI) cases as well.

In order to demonstrate the efficacy of the online *RMP* learning module, we analyze the learned pairwise *rmP* curves of MFEA-II across various task-pairs characterized by different degrees of similarities. Figure 4.12 reveals that the MFEA-II successfully deciphers the underlying inter-task similarities *online*. Notice from Figures 4.12a and 4.12b that high values of *rmP* are learned (close to 1) for problem sets CI+HS and NI+HS that were known to have high inter-task similarities ($R_s \approx 1$). Similarly, relatively lower *rmP* values are learned for the problem pairs NI+MS and CI+LS with lower inter-task similarity; see Figures 4.12c and 4.12d.

FIGURE 4.12: Learned pairwise *rmp* values between pairs of tasks achieved by MFEA-II on different problem sets characterized by different degrees of inter-task similarities. The plots are averaged over 30 independent runs with the shaded regions spanning one-half standard deviation on either side of the mean.

Intuitively, based on the above observations, we conjecture that the online *rmp* estimation scheme indeed captures synergies (and identifies differences) across diverse optimization tasks in multitask settings.

To give a concrete example of the added benefits of online parameter estimation, we show the convergence behavior of MFEA-II against its counterparts MFEA-I and CEA for the CI+HS case. As is clear from Figure 4.13, MFEA-II is overall superior to MFEA-I and CEA. It can be seen that MFEA-II converges significantly faster (within 100,000 FE's) in comparison to MFEA-I on both tasks, T_1 and T_2 . The reason behind this accelerated convergence of MFEA-II can be inferred from the learned *rmp* trend in Figure 4.12a.

FIGURE 4.13: Convergence trends (averaged over 30 independent runs) achieved by the three tested algorithms on the problem set CI+HS with high degrees of inter-task similarity.

4.6 Chapter Summary

In this chapter, we proposed a novel online transfer parameter estimation scheme for evolutionary multitasking, capable of learning and exploitation of inter-task synergies on the fly. As a first step to our proposition, we theoretically analyze the shortcomings of the existing evolutionary multitasking framework MFEA from the standpoint of its susceptibility to negative transfers. Accordingly, we introduce the MFEA-II, integrated with *online transfer parameter estimation* to dynamically control the extent of knowledge exchange across tasks. Specifically, the extent of transfer is adapted based on the optimal blending (mixing) of probabilistic models in order to capture inter-task similarities *online* in a purely data-driven manner. The algorithmic contributions of MFEA-II can be summarized from two perspectives:

- The transfer parameter takes the form of a symmetric matrix for effectively multitasking across more than two tasks with possibly diverse inter-task relationships.
- The transfer parameter matrix is learned and adapted *online* during the course of the multitasking search.

The efficacy of the MFEA-II was demonstrated experimentally on a series of discrete and continuous synthetic benchmarks. Essentially, the experimental results

reveal that the algorithm is indeed adept at exploiting the similarities and discrepancies between tasks during multitask optimization, thereby allaying any fear of harmful inter-task interactions. Furthermore, the proposed method is automatic, alleviates human intervention (for ascertaining inter-task synergies), and yields consistently superior performance in comparison to the present day baseline MFEA.

Chapter 5

Cognizant Multitasking in Multi-objective Multifactorial Evolution: MO-MFEA-II

5.1 Introduction

Chapter 4 proposed the MFEA-II, that learns inter-task relationships based on overlaps in the probabilistic search distributions derived from data generated during the course of multitasking - and accordingly adapts the extent of genetic transfers *online*. However, the applicability of MFEA-II purely focussed on scalar valued single-objective optimization problems. With this in mind, the current chapter presents a realization of a *cognizant* evolutionary multitasking engine within the domain of multi-objective optimization. In particular, this chapter seeks to analyze how similar concepts of MFEA-II can apply to the domain of multi-objective optimization as well¹. Notably, this work shall be among the first to utilize probabilistic modeling to capture inter-task relationships between multi-objective optimization tasks in the context of evolutionary multitasking. The efficacy of the proposed method is substantiated on a variety of multi-objective benchmark test problems.

The organization of the remainder of this chapter is as follows. Section 5.2 presents the backgrounds of multi-objective optimization and associated multi-objective

¹The work in this chapter has been published in [78]

evolutionary algorithms. In addition, it provides a brief review of related works in the literature and summarizes the existing evolutionary multitasking algorithm for multi-objective optimization; i.e., the multi-objective multifactorial evolutionary algorithm (MO-MFEA). The cognizant multitasking engine, namely MO-MFEA-II, with online transfer adaptation capability is presented in Section 5.3. Experimental results and analyses on a wide range of multi-objective benchmarks are provided in Section 5.4. Section 5.5 concludes the Chapter with a summary of the proposed work.

5.2 Background

In this section, preliminaries of multi-objective optimization and associated single-task evolutionary algorithms are first presented. Subsequently, the existing MO-MFEA is discussed and its potential limitations highlighted.

5.2.1 Multi-objective Optimization

The goal of solving multi-objective optimization problems (MOOPs) is to find multiple trade-off solutions that offer an optimal compromise between conflicting objectives of interest. In what follows, we provide a general formulation for a MOOP.

Given a decision space $\mathbf{X} \subset \mathbb{R}^D$, a multi-objective minimization problem can be defined (without loss of generality) as follows:

$$\text{minimize } F(x) = (f_1(x), f_2(x), \dots, f_M(x)), \quad (5.1)$$

where $f_1(x), f_2(x), \dots, f_M(x)$ are M different minimization objectives, $F(x)$ is the overall objective vector, and $x \in \mathbf{X}$ is a decision vector. For the sake of brevity, additional constraint functions have been suppressed in Eq. (5.1). Following the concept of *Pareto dominance* [22], a solution x_1 is said to *dominate* another solution x_2 (i.e. $x_1 \prec x_2$) iff $\forall i \in \{1, 2, \dots, M\}: f_i(x_1) \leq f_i(x_2)$, and $\exists j \in \{1, 2, \dots, M\}: f_j(x_1) < f_j(x_2)$. Accordingly, a solution x^* is considered to be Pareto optimal if x^* is not dominated by any other solution in $\mathbf{X}$. The objective function values corresponding to all the Pareto optimal solutions collectively form the Pareto front [79].

5.2.2 Multi-objective Evolutionary Algorithms (MOEAs)

Evolutionary algorithms (EAs) are considered a popular choice for solving MOOPs. In particular, the implicit parallelism offered by a population enables simultaneous sampling, evaluation and processing of multiple regions of the search space, leading to a reasonable representation of the entire set of Pareto optimal solutions to be found in only a single run of the solver [30, 80]. In the literature, some examples of Multi-objective Evolutionary Algorithms (MOEAs) that are commonly used today include NSGA-II [81], NSGA-III [82, 83], SPEA2 [84] and [14], MOEA/D [85–87], to name a few. While a plethora of MOEAs have been proposed over the years, the focus has mostly been to efficiently solve only one MOOP at a time. On the other hand, attempts to tackle multiple related MOOPs simultaneously (in the spirit of evolutionary multitasking) have been relatively scarce. Some recent works have however been discussed in Chapter 2.

5.2.3 Overview of the Existing MO-MFEA

The MO-MFEA [22] is a recently proposed method that allows efficient resolution of multiple MOOPs simultaneously by promoting *omnidirectional* knowledge transfers [12]. By this we mean that all tasks can benefit from each other via mutual knowledge sharing. For this to be possible, a *unified search space* encoding solutions to all constituent optimization tasks is first defined. At least in the case of box-constrained continuous optimization, such unification is achieved in a straightforward manner by a simple linear map between the task-specific solution space and the unified space [88] - which is also continuous and is defined in the range $[0, 1]^D$.

In the MO-MFEA, the primary mode of knowledge transmission is through implicit genetic transfers during *inter-task crossovers* between parent solutions belonging to different tasks. Essentially, the extent of genetic transfer is mandated by a user-defined and fixed scalar transfer parameter defined as the *random mating probability* (*rmpr*). For further details about the MO-MFEA, the reader is referred to [22].

5.2.3.1 Tuning the *rpm* parameter

It is noteworthy that the *rpm* parameter in MO-MFEA is *manually* specified based on the intuition of a decision maker. That is, if the optimization tasks are known to have a high degree of inter-task similarity, the *rpm* is usually set to a value close to 1. On the other hand, a low *rpm* value (closer to 0) is prescribed between optimization tasks that are known to bear low inter-task similarity. It is indeed patently clear that such an *offline rpm assignment* scheme is heavily dependent on the existence of prior knowledge in the mind of the practitioner about the relationships between different optimization problems. In practical settings, access to domain specific knowledge about different optimization problems is often limited, particularly in general black-box optimization. Given the possible lack of prior knowledge about inter-task relationships, the appropriate selection of *rpm* can become a significant hurdle for a practitioner. In fact, it has also been shown that inappropriate (blind) prescription of *rpm* values (via trial and error) risks the possibility of harmful genetic transfers, thereby leading to significant performance slowdowns [62]. Loosely speaking, the present day MO-MFEA typically does not provide any explicit measures to guard against the threat of negative transfer.

5.3 Cognizant Multitasking MO-MFEA-II

This section provides details of the proposed MO-MFEA-II, which offers a data-driven and cognitive enhancement of its predecessor MO-MFEA. Specifically, we would want the algorithm to be able to automatically decide *when* and *how much* knowledge to transfer such that the threat of negative transfers is minimized. To this end, the MO-MFEA-II learns the *rpm* parameter *online* (in place of an *offline rpm* assignment) and thus adapts the extent of genetic transfers between diverse tasks in a multitask setting. In order to develop an online *rpm* estimation technique with knowledge transfer adaptation capabilities, two key ideas have been incorporated in the existing MO-MFEA. These include:

- Utilizing a *RMP* matrix in place of a scalar *rpm* to effectively multitask across more than two tasks with possibly diverse inter-task relationships. For example, given three different optimization tasks, the degree of similarity

between the first and the second task maybe different from that between the first and the third. Therefore, using the same rmf value for all task pairs is considered to be too restrictive.

- Defining the MO-MFEA framework from a *probabilistic modeling perspective*, so as to allow capturing of inter-task similarities based on the quantification of overlaps in search distributions. The MO-MFEA-II is equipped to model the data generated online during the course of a multitasking search to unveil recurring solution patterns across different tasks.

5.3.1 RMP matrix

Given K optimization tasks in a multitask setting, the RMP takes the form of a symmetric $K \times K$ matrix of pairwise rmf values represented as,

$$RMP = \begin{bmatrix} rmp_{1,1} & rmp_{1,2} & \cdot & \cdot \\ rmp_{2,1} & rmp_{2,2} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad (5.2)$$

where $rmf_{k,j} = rmp_{j,k}$ reflects the extent of transfer between the k^{th} and j^{th} task-pair. Note that $rmf_{k,k} = 1, \forall k$. Unlike the scalar rmf of the present day MO-MFEA, the RMP matrix offers the distinct advantage of adapting the extent of knowledge transmissions between diverse task-pairs with possibly non-uniform inter-task similarities.

5.3.2 Latent Search Distributions in the MO-MFEA

Here, we briefly describe how knowledge transfers are actualized in the original MO-MFEA, through the *implicit mixture of search distributions drawn from different tasks*.

For $k \in \{1, 2, \dots, K\}$, consider the parent subpopulation associated with the k^{th} task T_k to be denoted as $P^k(t)'$ at iteration t . Further, let the subpopulations associated with each of the K tasks originate from the *true* (but unknown) underlying probability distributions $p^1(x, t), p^2(x, t), \dots, p^K(x, t)$ respectively, i.e., $P^k(t)' \sim p^k(x, t)$ for all $k \in \{1, 2, \dots, K\}$.

As a consequence of the *inter-task* interactions in the MO-MFEA (through inter-task crossovers), the offspring population generated for the k^{th} task at the t^{th} iteration can be shown to be drawn from the following mixture distribution, under the simplifying assumption of *parent-centric* evolutionary operations (see Appendix A.1).

$$p_c^k(x, t) = \left[1 - \frac{0.5}{K} \cdot \sum_{k \neq j} rmp_{k,j} \right] \cdot p^k(x, t) + \frac{0.5}{K} \sum_{j \neq k} rmp_{k,j} \cdot p^j(x, t). \quad (5.3)$$

The finite mixture $p_c^k(x, t)$ is a linear combination of all K task distributions, with the degree of mixing given by $rmp_{k,j}$'s. Note that $0 \leq rmp_{k,j} \leq 1$. Clearly, the most trivial way to prevent any harmful (negative) inter-task interactions is to simply set the $rmp_{k,j}$ values to 0, leading to the cancellation of all knowledge transfers. In this case, Eq. (5.3) implies that $p_c^k(x, t) = p^k(x, t)$.

5.3.3 Preliminaries on Probabilistic Model-based Search

For an ideal multitasking engine, the goal is to promote fruitful transfers whenever possible (i.e., by learning positive pairwise rmp values) while also minimizing negative inter-task interactions. The proposed idea is thus to leverage on the data generated during the course of a multitask optimization run, and to learn an RMP matrix that leads to an effective mixture distribution in Eq. (5.3). For instance, if the search distributions of the respective tasks are far from each other in a unified search space, such that no relationship can be inferred between them, we would want the corresponding pairwise rmp values to be automatically very low (near zero). On the other hand, if the search distributions are overlapping, higher rmp values should ideally be prescribed. Such a learning strategy is expected to benefit related tasks while not hurting the performance of unrelated tasks.

In order to do this learning, we draw from existing theories in the field of *probabilistic model-based search algorithms*. These algorithms guide the optimization search by iteratively building probabilistic models of parent solutions, and subsequently generating new offspring by sampling these models. The typical steps can be summarized as follows.

- **Step 1:** Start with a uniform distribution model $q(x, 0)$ over the search space

- **Step 2:** At iteration t , sample N offspring solutions from the current model $q(x, t - 1)$
- **Step 3:** Based on the fitness of sampled solutions, *select* $n < N$ parent solutions
- **Step 4:** Build a probabilistic model $q(x, t)$ of the parent solutions
- **Step 5:** Increment iteration count and go to Step 2, unless a termination condition is satisfied.

A well known result which supports this class of algorithms is summarized in Theorem 5.3.1. Briefly, the result highlights the benefit of learning probabilistic models $q(x, t)$ that capture the true underlying parent distribution $p(x, t)$ accurately, in order to attain asymptotic convergence guarantees. It is this theorem that also serves as the basis of the mathematical program proposed in the chapter to learn the *RMP* matrix online, as shall be detailed in Section (5.3.5).

Theorem 5.3.1. Consider a continuous optimization task T with objective function f and global optimum f^* . The prior distribution $q(x, 0)$ from which initial solutions are sampled is assumed to be positive and continuous everywhere in the search space $\mathbf{X}$, with $N \rightarrow \infty$. With this, at every iteration “ t ” of a probabilistic model-based EA, if the learned model $q(x, t)$ – from which offspring are sampled – is identical to the true parent distribution $p(x, t)$, then the search asymptotically converges to a distribution $p(x, \infty)$ such that $\int_{\mathbf{X}} f(x) \cdot p(x, \infty) \cdot dx = f^*$.

The proof of the above can be found in [68]. We note that this result originally pertains to the case of single-objective optimization, and has recently been used in [62] for learning optimized mixture models in the context of multitasking with single-objective optimization tasks (Chapter 4). In contrast, in this chapter, we investigate how incorporation of the same result can lead to performance gains in the MO-MFEA as well. Details of the resultant MO-MFEA-II are presented next.

5.3.4 Probabilistic Mixture Modeling in MO-MFEA-II

At any iteration t of MO-MFEA-II, we start by building probabilistic models $q^k(x, t)$ for parent subpopulations $P^k(t)' \in \{P^1(t)', P^2(t)', \dots, P^K(t)'\}$. The model

$q^k(x, t)$ is seen as a limited data approximation of the *true* parent distribution $p^k(x, t)$. Given the learned models above, Eq. (5.3) can equivalently be written as,

$$q_c^k(x, t) = \left[1 - \frac{0.5}{K} \cdot \sum_{k \neq j} rmp_{k,j} \right] \cdot q^k(x, t) + \frac{0.5}{K} \sum_{j \neq k} rmp_{k,j} \cdot q^j(x, t), \quad (5.4)$$

where $q_c^k(x, t)$ is a mixture model approximating the *true* $p_c^k(x, t)$ in Eq. (5.3).

From Theorem 1, we know that in this probabilistic interpretation, asymptotic convergence is guaranteed in the limit $N \rightarrow \infty$ if the offspring distribution is equal to the parent distribution. This result suggests that learning the *RMP* matrix such that mixture model $q_c^k(x, t)$ in Eq. (5.4) accurately replicates $p^k(x, t)$, for all $k \in \{1, 2, \dots, K\}$, shall implicitly decrease the tendency of negative inter-task interactions.

5.3.5 Learning the RMP Matrix Online in MO-MFEA-II

To this end, given the probabilistic models $q^k(x, t)$ for all tasks $k \in \{1, 2, \dots, K\}$, the following mathematical program is formulated to learn the *RMP* matrix,

$$\max_{RMP} \sum_{k=1}^K \sum_{i=1}^n \log q_c^k(x_{ik}, t), \quad (5.5)$$

where x_{ik} is the i^{th} sample (individual) in parent subpopulation $P^k(t)$. The rationale behind this formulation can be seen from the following theorem.

Theorem 5.3.2. Maximizing the likelihood function in Eq. (5.5), in the limit $n \rightarrow \infty$, is equivalent to minimizing the Kullback-Leibler divergence $KL(p^k(x, t) || q_c^k(x, t))$ averaged across all tasks $\{T_1, T_2, \dots, T_k, \dots, T_K\}$.

The proof of the above can be found in [62] (and Section 4.4.1), where the Kullback-Leibler divergence is used as a standard measure of the discrepancy between probability distributions. With regards to the computational tractability of optimizing the *RMP* matrix, we note that the mathematical program in Eq. (5.5) is convex upward. Therefore, in practice, the *RMP* matrix can be learned at little computational overhead (as long as K is not large) using classical convex optimization solvers.

It is also worth highlighting the importance of the type of constituent probabilistic models $q^k(x, t)$ to be chosen for approximating the parent distributions $p^k(x, t)$. If we choose to build a complex (highly expressive) model on a typically small subpopulation dataset $P^k(t)'$, then this may cause model *overfitting* - which in turn can be shown to result in the cancellation of inter-task knowledge transfers in Eq. (5.4) as the trivial solution of $rm p_{k,j} = 0 \forall k \neq j$ optimizes the program in Eq. (5.5). To alleviate this overfitting issue, we propose to use simple (less expressive) and fast probabilistic models, e.g., univariate marginal distributions ignoring variable dependencies, to approximate the parent distributions; thus, allowing all other models in the multitask setting to be mobilized to fill in the distribution gaps and actualize knowledge transfers. An additional step that actively prevents the cancellation of inter-task transfers is to inject some random noise into the available subpopulation datasets. Specifically, a simple modification that is found to work well in practice is to add a small amount of uniformly sampled noise to the dataset $P^k(t)' \forall k$, resulting in a corrupted version $P^k(t)''$. Thereafter, building the model $q^k(x, t)$ on the corrupted dataset $P^k(t)''$ naturally prevents overfitting to the original dataset $P^k(t)'$ [89].

5.3.6 Summary of the MO-MFEA-II

Given all the enhancements above, we can now summarize the MO-MFEA-II algorithm.

5.3.6.1 Preliminary definitions

Consider the case of solving K multi-objective optimization tasks $\{T_1, T_2, \dots, T_K\}$ concurrently using a single population P of evolving individuals, encoded in a unified space (denoted as $\mathbf{Y}$). Let the subpopulation associated with the k^{th} task be denoted as P^k . An approach for comparing the fitness of candidate solutions in such a multitask setting is formulated by defining the following properties for each individual in P .

- **Definition 1** (*Skill Factor*): The *skill factor* τ_i corresponds to the *one* task (out of the available K tasks) the i^{th} individual is associated with in the multitasking environment.

- **Definition 2** (*Scalar Fitness*): The *scalar fitness* of the i^{th} individual is defined as $\varphi_i = 1/r_{\tau_i}^i$, where $r_{\tau_i}^i$ corresponds to the rank of the i^{th} individual on task τ_i . Note that $\max\{\varphi_i\} = 1$.

5.3.6.2 Algorithm summary

Algorithm 4 provides the pseudocode for MO-MFEA-II. The algorithm is built upon the *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [81], which utilizes *lexicographic ranking* based on each candidate solution's *non-dominated front* (NF) and *crowding distance* (CD) to guide the population as a whole through complex multi-objective search spaces. For the sake of brevity, further details of the NSGA-II have not been reproduced in this chapter.

Algorithm 4 Pseudocode of the MO-MFEA-II

```

1 Randomly sample  $N \cdot K$  individuals in  $\mathbf{Y}$  to form initial population  $P(0)$ 
2 for every individual  $p_i$  in  $P(0)$  do
3   Assign skill factor  $\tau_i = \text{mod}(i, K) + 1$ , for the case of  $K$  tasks
4   Evaluate  $p_i$  for task  $\tau_i$ 
5 Compute scalar fitness  $\varphi_i \forall p_i$  via lexicographic ranking of  $NF_i, CD_i$ 
6 Set  $t = 1$ 
7 while stopping conditions are not satisfied do
8   Configure offspring population  $P_c(t) = \emptyset$ 
9    $P(t)' = \text{Tournament Selection of } n \cdot K \text{ parent solutions from } P(t)$ 
10  Learn  $RMP(t)$  as per Eq. (5.5)
11  while offspring generated for each task  $< N$  do
12    Sample two individuals  $x_i$  and  $x_j$  at random from  $P(t)'$ 
13    if  $\tau_i == \tau_j$  then
14       $[x_a, x_b] \leftarrow \text{Intra-task crossover} + \text{mutate}(x_i, x_j)$ 
15      Assign offspring  $x_a$  and  $x_b$  skill factor  $\tau_i$ 
16    else if  $\text{rand} \leq \text{rmp}_{\tau_i, \tau_j}$  then
17       $[x_a, x_b] \leftarrow \text{Inter-task crossover} + \text{mutate}(x_i, x_j)$ 
18      Each offspring is randomly assigned skill factor  $\tau_i$  or  $\tau_j$ 
19    else
20      Randomly select  $x'_i$  with skill factor  $\tau_i$ 
21       $[x_a] \leftarrow \text{Intra-task crossover} + \text{mutate}(x_i, x'_i)$ 
22      Assign offspring  $x_a$  skill factor  $\tau_i$ 
23      Randomly select  $x'_j$  with skill factor  $\tau_j$  ;
24       $[x_b] \leftarrow \text{Intra-task crossover} + \text{mutate}(x_j, x'_j)$ 
25      Assign offspring  $x_b$  skill factor  $\tau_j$ 
26    Evaluate  $[x_a, x_b]$  for their assigned skill factors only
27     $P_c(t) = P_c(t) \cup [x_a, x_b]$ 
28   $R(t) = P_c(t) \cup P(t)$ 
29  Update scalar fitness of all individuals in  $R(t)$ ;
30  Select  $N \cdot K$  fittest members from  $R(t)$  to form  $P(t + 1)$ 
31   $t = t + 1$ 

```

The MO-MFEA-II evolves a single population of $N \cdot K$ individuals encoded in a unified space $\mathbf{Y}$. A uniform resource allocation scheme is adopted whereby each task $T_k \in \{T_1, T_2, \dots, T_K\}$ is equally assigned N individuals to comprise its subpopulation P^k . Lines 7–31 in Algorithm 1 form the main evolutionary multitasking loop, describing the steps associated with offspring generation, task-specific fitness evaluation and environmental selection.

For offspring creation, a pool of parent candidates $P(t)'$ is first selected through binary tournament selection (line 9). The selection is based on the scalar fitness of each of the individuals such that $P^1(t)' \cup P^2(t)' \cup \dots, P^K(t)' = P(t)'$. The selected parent subpopulations are then used to build probabilistic models and learn the *RMP* matrix according to Eq. (5.5). Notice that the *RMP* matrix is currently learned and adapted online at every iteration t (line 10).

Apart from the conventional *intra-task crossover* between parent solutions belonging to the same task (lines 13 – 15), *inter-task crossover* (lines 16 – 18) also takes place between parent solutions belonging to different tasks (i.e., parents having different skill factors). In particular, the *extent* of implicit genetic transfer in the MO-MFEA-II is governed by the pairwise *rmp* values in the learned *RMP* matrix - which mandates the probability of inter-task crossover between two solutions x_i and x_j that may have different skill factors. Clearly, if the learned *RMP* is the identity matrix, then no knowledge transfer will be actualized.

5.3.6.3 Selective evaluations in MO-MFEA-II

Each of the offspring individuals is evaluated with respect to only one task (its assigned skill factor) amongst all other tasks in the multitasking environment (line 26). This salient feature combats the issue of otherwise exhaustively evaluating every individual for every task, which would often be computationally demanding. Post evaluation, the fittest $N \cdot K$ individuals survive for the next iteration of the evolutionary search (line 30).

The aforementioned steps (lines 7 – 31 in Algorithm 4) repeat until some stopping condition is met².

²Source codes are available via <http://www.bdsc.site/websites/MTO/index.html>

5.4 Experimental Study

In this section, experimental results are presented that showcase the efficacy of the proposed MO-MFEA-II in the domain of multi-objective optimization. The performance of MO-MFEA-II is compared against its predecessor, the MO-MFEA and its baseline algorithm, i.e., NSGA-II performing a single task at a time. For the case of synthetic benchmarks tackling two MOOPs at a time (Sections 5.4.4.1 & 5.4.4.2), the *evolutionary multitasking via explicit autoencoding* (EMEA) [29] algorithm is considered as an additional state-of-the-art multitasking baseline for comparison. Essentially, the EMEA leverages on the search biases from two different MOEAs (i.e., NSGA-II & SPEA2) acting upon two distinct tasks, respectively. During knowledge transfers, candidate solutions are first transformed via a linear mapping before being transferred from one task to the other.

5.4.1 Experimental Configuration

The experimental setup is outlined in what follows. To ensure consistency, the population size N (per task) is kept the same for all algorithms. For all the test problems under consideration, the following general settings are applied:

1. Continuous unified search space $[0, 1]^D$;
2. Evolutionary operators used for MO-MFEA-II, MO-MFEA, and NSGA-II are:
 - (a) SBX crossover with probability $(p_c) = 1$ and distribution index $\eta_c = 10$;
 - (b) Polynomial mutation with probability $(p_m) = 1/D$ and distribution index $\eta_m = 10$;
3. Probability model in MO-MFEA-II: Normal distribution;
4. Population size (N):
 - (a) For synthetic benchmarks: 50;
5. Maximum number of generations: 250;

6. *RMP* is learned online for MO-MFEA-II. For MO-MFEA, *rmp* values set as either 0.9, or 0.6, or 0.3;
7. Transfer interval in the EMEA: 10 generations.

Note that no uniform crossover like variable swap is applied between offspring to ensure parent-centric crossover operations with minimum chromosome disruptions [90], thus allowing us to clearly observe the effects of inter-task implicit genetic transfers.

5.4.2 Specification of Benchmark MOOPs

We highlight that the following benchmarks are created based on prior knowledge of the relationships between tasks in multitask settings. The relationship is measured in terms of the overlap between their optimal solutions and the rank (ordinal) correlations between their function landscapes [80]. Given such prior knowledge, evolutionary multitasking is naturally expected to benefit from the scope for genetic transfers between tasks that share a degree of inter-task similarity. On the other hand, it may be less effective to conduct knowledge exchange across tasks that have little in common, as this could potentially lead to negative transfers and impede the optimization process. In the latter scenario, an ideal multitasking algorithm should be able to automatically reduce inter-task interactions, thereby performing no worse than its baseline single-tasking algorithm.

In this particular study, we consider a variety of MOOP benchmarks proposed for evolutionary multitask optimization [22, 80]. These problems are characterized by different properties, including the type of the Pareto front, multimodality and separability.

5.4.2.1 Problem 1

In this multi-objective problem, $g(\mathbf{x})$ takes the form of the Sphere function,

$$\begin{aligned}
 \min f_1(\mathbf{x}) &= g(\mathbf{x}) \cos\left(\frac{\pi x_1}{2}\right), \\
 \min f_2(\mathbf{x}) &= g(\mathbf{x}) \sin\left(\frac{\pi x_1}{2}\right), \\
 g(\mathbf{x}) &= 1 + \sum_{i=2}^D x_i^2, \\
 x_1 &\in [0, 1], x_i \in [-100, 100], i = 2, 3, \dots, D.
 \end{aligned} \tag{5.6}$$

5.4.2.2 Problem 2

This particular MOOP is defined as follows,

$$\begin{aligned}
 \min f_1(\mathbf{x}) &= x_1; \\
 \min f_2(\mathbf{x}) &= g(\mathbf{x}) \left(1 - \left(\frac{x_1}{g(\mathbf{x})}\right)^2\right), \\
 g(\mathbf{x}) &= 1 + \frac{9}{n-1} \sum_{i=2}^D |x_i|, \\
 x_1 &\in [0, 1], x_i \in [-100, 100], i = 2, 3, \dots, D.
 \end{aligned} \tag{5.7}$$

5.4.2.3 Problem 3

Here, $g(\mathbf{x})$ takes the form of the Griewank function [91],

$$\begin{aligned}
 \min f_1(\mathbf{x}) &= g(\mathbf{x}) \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right), \\
 \min f_2(\mathbf{x}) &= g(\mathbf{x}) \cos\left(\frac{\pi x_1}{2}\right) \sin\left(\frac{\pi x_2}{2}\right), \\
 \min f_3(\mathbf{x}) &= g(\mathbf{x}) \sin\left(\frac{\pi x_1}{2}\right), \\
 g(\mathbf{x}) &= 2 + \frac{1}{4000} \sum_{i=3}^D (z_i)^2 - \prod_{i=3}^D \cos\left(\frac{z_i}{\sqrt{i-2}}\right) \\
 (z_3, z_4, \dots, z_D)^T &= (x_3, x_4, \dots, x_D)^T - \mathbf{s}, \\
 x_1 &\in [0, 1], x_2 \in [0, 1], x_i \in [-50, 50], i = 3, 4, \dots, D,
 \end{aligned} \tag{5.8}$$

where $\mathbf{s} = (20, 20, \dots, 20)$ represents a shift vector.

5.4.2.4 Problem 4

Here, $g(\mathbf{x})$ takes the form of the Ackley function [92],

$$\begin{aligned}
 \min f_1(\mathbf{x}) &= \frac{1}{2}(x_1 + x_2), \\
 \min f_2(\mathbf{x}) &= g(\mathbf{x}) \left(1 - \left(\frac{x_1 + x_2}{2 \cdot g(\mathbf{x})}\right)^2\right), \\
 g(\mathbf{x}) &= -20 \exp\left(-0.2 \sqrt{\frac{1}{D-2} \sum_{i=3}^D x_i^2}\right) \\
 &\quad - \exp\left(\frac{1}{D-2} \sum_{i=3}^D \cos(2\pi x_i)\right) + 21 + e \\
 x_1 &\in [0, 1], x_2 \in [0, 1], x_i \in [-100, 100], i = 3, 4, \dots, D.
 \end{aligned} \tag{5.9}$$

5.4.2.5 Problem 5

In this case, $g(\mathbf{x})$ takes the form of the Rastrigin function [93, 94],

$$\begin{aligned}
 \min f_1(\mathbf{x}) &= x_1, \\
 \min f_2(\mathbf{x}) &= g(\mathbf{x}) \left(1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}\right), \\
 g(\mathbf{x}) &= 1 + 10(D-1) + \sum_{i=1}^{D-1} z_i^2 - 10 \cos(4\pi z_i) \\
 (z_2, z_3, \dots, z_D)^T &= M \cdot (x_2, x_3, \dots, x_D)^T, \\
 x_1 &\in [0, 1], x_i \in [-5, 5], i = 2, 3, \dots, D,
 \end{aligned} \tag{5.10}$$

where M is a randomly generated $(D-1) \times (D-1)$ rotation matrix. The properties of these MOOPs are summarized in Table 5.1. In all cases, the dimensionality of the search space is set to $D = 10$. The Inverted Generational Distance (IGD) measure [95] is used for performance comparison between all the algorithms under consideration.

TABLE 5.1: Summary of test problems

	Pareto Front	Properties
Problem 1	$f_1^2 + f_2^2 = 1$ $f_1 \geq 0, f_2 \geq 0$	concave, unimodal, separable
Problem 2	$f_2 = 1 - f_1^2$ $0 \leq f_1 \leq 1$	concave, unimodal, separable
Problem 3	$\sum_{i=1}^3 f_i^2 = 1$ $f_i \geq 0, i = 1, 2, 3.$	concave, multimodal, nonseparable
Problem 4	$f_2 = 1 - f_1^2$ $0 \leq f_1 \leq 1$	concave, multimodal, nonseparable
Problem 5	$f_2 = 1 - \sqrt{f_1}$ $0 \leq f_1 \leq 1$	convex, multimodal, nonseparable

5.4.3 Measuring Complementarity Between Constitutive Tasks

In [80], two quantities have been proposed to measure the complementarity between different multi-objective optimization tasks in a multitask setting. Firstly, the degree of overlap of the global optima of the $g(\mathbf{x})$ functions of constitutive tasks is considered. A scenario in which there is complete overlap between the respective global optima in the unified space is defined as *complete intersection*, and is denoted as CI. On the other hand, the global optima of task-pairs may also *not intersect*, which we denote as NI. Secondly, a Spearman's rank correlation (R_s) measure is utilized to capture the similarity between the overall trends of the fitness landscape of tasks. The task-pairs for which the Spearman rank correlation $R_s < 0.2$ are characterized as having low inter-task similarity (LS); the task-pairs in the range $0.2 \leq R_s < 0.8$ are defined as having medium similarity (MS); and $R_s \geq 0.8$ implies high inter-task similarity (HS).

5.4.4 Experimental Results

5.4.4.1 Complete Intersection and High Similarity: CI-HS

We begin by considering a composite MOOP set with complete intersection and high similarity. In this setup, task T_1 is represented by *Problem: 1* (Eq. (5.6)) while task T_2 is represented by *Problem: 2* (Eq. (5.7)). The inter-task similarity

FIGURE 5.1: **Top:** Convergence trends (averaged over 30 independent runs) achieved by all the algorithms in the CI-HS composite MOOP benchmark. **Below:** Pairwise rmp 's learned between tasks T_1 and T_2 over successive generations. The shaded regions span one half standard deviation on either side of the mean.

is measured to be $R_s = 0.97$. The convergence trends (IGD metric averaged over 30 independent runs) for the first 100 generations are shown in Figure 5.1.

Since the composite MOOPs share high degrees of inter-task similarities, the effect of knowledge transfers in all the multitasking algorithms is observed to be beneficial. The results indicate that both MO-MFEA-II and MO-MFEA outperform the single tasking NSGA-II. Figures 5.1a and 5.1b show that MO-MFEA variants with high specified rmp values (i.e., 0.9 and 0.6, inducing high frequency of genetic transfers) showcase better convergence trends than MO-MFEA with a lower prescribed rmp value of 0.3. As such, note that the MO-MFEA requires careful rmp tuning

in order to achieve the desired performance. In contrast, the MO-MFEA-II alleviates the hurdle of manual parameter tuning. By learning the pairwise $rm\bar{p}$ values online and adapting knowledge transfers on the fly, the MO-MFEA-II has recorded improved convergence characteristics for both tasks T_1 and T_2 . Figure 5.1c reveals that the learning module of MO-MFEA-II is able to deduce $rm\bar{p}$ values that agree well with the *a priori* known inter-task similarity; high $rm\bar{p}$ values are learned throughout the optimization process implying high frequency of transfers.

We further compare the MO-MFEA-II with the results achieved by the EMEA algorithm for the CI-HS case. Recall that the EMEA employs two different solvers to tackle the two different tasks; T_1 is optimized by NSGA-II while T_2 is optimized by SPEA2. As observed from Figure 5.1, the EMEA has shown similar performance behavior as the MO-MFEA-II and other MO-MFEA variants. It leverages on the scope of knowledge transfers to improve its baseline algorithms for both the tasks.

5.4.4.2 No Intersection and Low Similarity: NI-LS

Next, we consider the extreme case of tackling two unrelated MOOPs with no intersection of the global optima and having low inter-task similarity. In this setup, task T_1 is represented by *Problem: 3* (Eq. (5.8)) while task T_2 is represented by *Problem: 4* (Eq. (5.9)). The inter-task similarity is measured to be $R_s = 0.08$. Experimental results are reported in Figure 5.2.

The lack of inter-task similarity in this scenario suggests that there exists little scope for crossover-based inter-task knowledge transfers. However, consider the convergence trends of the MO-MFEA variants with higher frequency of transfers (i.e $rm\bar{p} = 0.6$ and $rm\bar{p} = 0.9$). While these configurations seem to have improved the performance of task T_2 in comparison to NSGA-II (Figure 5.2b), we observe that their performance deteriorates significantly on task T_1 (Figure 5.2a). This may be a counter-effect of the *omnidirectional* transfer mechanism, which, if not controlled, could lead to undesired negative transfers for one of the tasks. This conjecture is also supported by the results of EMEA on T_1 , which is found to be hampered by negative transfers as well. As such, the MO-MFEA variants are seen to improve one task at the expense of worsening the other, but, with a decreased $rm\bar{p}$ value (0.3), such counter-effects are minimized. On the other hand, MO-MFEA-II has recorded superior convergence trends for both tasks. As shown by

FIGURE 5.2: **Top:** Convergence trends of all the algorithms in the NI-LS composite MOOP benchmark. **Bottom:** Pairwise learned rmp 's.

the learned rmp curve in Figure 5.2c the frequency of transfer is low, thus making the algorithm less prone to the impact of negative transfers.

5.4.4.3 Study with more than two tasks

Studies focusing on many-tasking have been carried out in the domain single-objective optimization [96, 97]. Along similar lines, we extend our current experimental study to more than two multi-objective tasks. In particular, we attempt to analyze the effects of omnidirectional transfers in the context of pooling several tasks (up to five MOOPs) in the same multitasking environment. For this set of experiments, all the five problems defined in section 5.4.2 are considered. Tasks $T_1, T_2, \dots, T_5$ are represented by *Problems: 1, 2, \dots, 5*, respectively. The inter-task similarities are given in Table 5.2.

TABLE 5.2: Inter-task similarities R_s

	T_1	T_2	T_3	T_4	T_5
T_1	1	0.970	0.298	0.778	0.402
T_2	-	1	0.299	0.753	0.391
T_3	-	-	1	0.08	0.130
T_4	-	-	-	1	0.817
T_5	-	-	-	-	1

TABLE 5.3: Average IGD values achieved by NSGA-II, MO-MFEA-II and MO-MFEA (with a range of rmf values) for all $K = 5$ Tasks. Bold face entries are the best (by WILCOXON test with significance level 0.05).

	Tasks				
	T_1	T_2	T_3	T_4	T_5
NSGA-II	$7.75e-02$ ($3.31e-02$)	$3.46e-01$ ($1.45e-01$)	$1.90e+01$ ($4.88e+00$)	$3.97e-01$ ($5.65e-02$)	$1.70e+00$ ($8.21e-01$)
MO-MFEA-II	$1.98e-02$ ($5.70e-03$)	$2.27e-01$ ($6.94e-02$)	$1.67e+01$ ($7.40e+00$)	$3.74e-01$ ($5.14e-02$)	$1.58e+00$ ($7.88e-01$)
MO-MFEA	($rmf = 0.90$)	$1.24e-01$ ($5.92e-02$)	$8.11e-01$ ($2.47e-01$)	$2.04e+01$ ($5.14e-02$)	$8.86e-01$ ($1.98e-01$)
	($rmf = 0.60$)	$5.24e-02$ ($2.31e-02$)	$5.07e-01$ ($1.30e-01$)	$1.90e+01$ ($7.40e+00$)	$5.60e-01$ ($5.68e-02$)
	($rmf = 0.30$)	$3.57e-02$ ($1.39e-02$)	$3.91e-01$ ($1.06e-01$)	$1.88e+01$ ($4.77e+00$)	$4.38e-01$ ($5.44e-02$)

FIGURE 5.3: Convergence trends (averaged over 30 independent runs) achieved by all the algorithms on the 5 tasks.

Experimental results are reported in Table 5.3 and Figure 5.3. According to the results, MO-MFEA-II records the overall best performance. More importantly, no harmful effects of negative transfer is seen for Mo-MFEA-II in any of the five tasks. Clearly, such superior performance is attributed to the fact that MO-MFEA-II incorporates a *RMP* matrix, capable of capturing diverse inter-task relationships between various task pairs and adapting the extent of transfers accordingly.

In contrast, the MO-MFEA variants that employ a user-specified scalar *rmp* are found to suffer. Since the *rmp* is fixed, the extent of transfer remains the same (uniform) across all task pairs (irrespective of the existence of non-uniform inter-task synergies reported in Table 5.2). In such a restrictive approach, the threats of negative transfers are observed to be predominant. For instance, while analyzing the convergence trends in Figure 5.3, it can be observed that prescribing a relatively high *rmp* (e.g., 0.6) may bolster performance in a few tasks while hampering others (in most cases, the conventional MO-MFEA performs worse than single-tasking NSGA-II).

Furthermore, Tables 5.4 and 5.5 show numerical results for pooling $K = 4$ and $K = 3$ tasks (MOOPs) together, respectively. Performance trends similar to Table 5.3 are observed.

TABLE 5.4: Average IGD values for all $K = 4$ Tasks

		Tasks			
		T_1	T_2	T_3	T_4
NSGA-II		$7.75e-02$ ($3.31e-02$)	$3.46e-01$ ($1.45e-01$)	$1.90e+01$ ($4.88e+00$)	$3.97e-01$ ($5.65e-02$)
MO-MFEA-II		$2.44e-02$ ($1.12e-02$)	$2.83e-01$ ($9.23e-02$)	$1.83e+01$ ($6.34e+00$)	$3.92e-01$ ($7.46e-02$)
MO-MFEA	$(rmp = 0.90)$	$1.21e-02$ ($1.12e-02$)	$7.46e-01$ ($1.77e-01$)	$2.04e+01$ ($1.30e+01$)	$9.00e-01$ ($8.57e-02$)
	$(rmp = 0.60)$	$4.38e-02$ ($1.42e-02$)	$4.75e-01$ ($1.03e-01$)	$1.87e+01$ ($5.24e+00$)	$5.30e-01$ ($1.27e-01$)
	$(rmp = 0.30)$	$3.43e-02$ ($8.30e-03$)	$3.42e-01$ ($7.47e-02$)	$2.03e+01$ ($2.19e+02$)	$4.44e-01$ ($6.97e-02$)

5.5 Chapter Summary

In this Chapter, we presented a cognizant multitasking multi-objective multifactorial evolution algorithm, namely, MO-MFEA-II. The salient feature of MO-MFEA-II is its *self awareness*, constantly modeling diverse relationships in the optimization

TABLE 5.5: Average IGD values for all $K = 3$ Tasks

	Tasks		
	T_1	T_2	T_3
NSGA-II	$7.75e - 02$ ($3.31e - 02$)	$3.46e - 01$ ($1.45e - 01$)	$1.90e + 01$ ($4.88e + 00$)
MO-MFEA-II	$2.27e - 02$ ($6.20e - 03$)	$2.75e - 01$ ($7.35e - 02$)	$1.31e + 01$ ($9.45e + 00$)
MO-MFEA	($rmp = 0.90$)	$4.19e - 02$ ($1.27e - 02$)	$4.26e - 01$ ($7.50e - 02$)
	($rmp = 0.60$)	$3.85e - 02$ ($1.77e - 02$)	$1.88e + 01$ ($4.87e + 00$)
	($rmp = 0.30$)	$4.02e - 01$ ($1.03e - 01$)	$1.86e + 01$ ($5.67e + 00$)
		$2.74e - 02$ ($8.80e - 03$)	$2.04e + 01$ ($6.56e - 02$)

environment and adapting its own mechanisms accordingly. We demonstrate the efficacy of MO-MFEA-II experimentally on various benchmark MOOPs. The experimental results show that the proposed method is capable of learning inter-task relationships between different MOOPs, demonstrating the algorithm's ability to decipher *when* and *how much* knowledge to transfer across different tasks. As opposed to the restrictive scalar transfer parameter in the present day MO-MFEA, the flexibility of MO-MFEA-II to capture diverse inter-task relationships is enhanced through a *RMP* matrix representation. The practical implications of MO-MFEA-II shall be showcased in the next chapter.

Chapter 6

Applications of Evolutionary Multitasking

6.1 Introduction

In previous studies, the applicability of multitasking in diverse real-world domains, including operations research [98], engineering [22], machine learning [99], etc., has been demonstrated. Accordingly, in this chapter, the goal is more towards analyzing the behavior of the newly proposed MFEA-II and MO-MFEA-II in practical scenarios¹. Notably, this chapter aims to highlight the following contributions of adaptive knowledge transfers in evolutionary multitask settings:

- *Omnidirectional knowledge transfer* - a feature unique to multitasking that allows mutual knowledge sharing between all distinct tasks of interest (unlike sequential transfer techniques where knowledge sharing is unidirectional)
- *The boon of intentional-unintentional problem solving capabilities* - a scenario that arises when intentionally attempting to solve the harder tasks, other simpler tasks can be implicitly (unintentionally) solved [13].
- *Scalability to many tasking* - the ability to handle more than two (i.e., upto six) complex tasks at once.

¹The works in this chapter have been published in [62, 78]

- *Speeding up expensive optimization* - leveraging on cheaper (helper) tasks to substantially reduce the cost of expensive optimization tasks.

The organization of the rest of the chapter is as follows. Section 6.2 considers a neuroevolution-based approach to simultaneously solving multiple single-objective variants of the double-pole balancing controller design task. In particular, we show that in a multitasking environment (via MFEA-II), not only can complex tasks benefit from simpler ones, but the reverse can also hold (as a consequence of omnidirectional knowledge transfer). Notably, this is a feature unique to multitasking that is not available in sequential transfer procedures. We also showcase the efficacy of the MFEA-II in the face of *many* tasks occurring at once. Further, Section 6.3 considers another practical case study in the domain of multi-objective risk averse UAV Path Planning. Importantly, the MO-MFEA-II demonstrates the benefits of knowledge transfers from low fidelity optimization tasks to substantially reduce the cost of high fidelity optimization. Thereafter, the applicability of multitasking is extended to the otherwise expensive domain of evolutionary deep learning in Section 6.4. Finally, Section 6.5 concludes the chapter.

6.2 Part I: The Double-Pole balancing Problem: A practical study of MFEA-II

In this section, we investigate the practicality of MFEA-II on increasingly challenging variants of the double-pole balancing controller design task [100]. For rigorous analysis, the performance of the MFEA-II is compared against a number of baseline solvers. These include the basic counterparts, i.e, the present day multifactorial evolutionary algorithm (denoted as MFEA-I) [21], the standalone canonical EA (CEA) and two state-of-the-art variants of *natural evolutionary strategies* (NES) [101]: the *separable natural evolutionary strategy* (SNES) [102] and the *exponential natural evolutionary strategy* (xNES) [103]. It is worthwhile to mention that NES has garnered much recent attention as a powerful evolutionary solver, with notable applications in OpenAI² projects on reinforcement learning [104].

²<https://openai.com/>

6.2.1 Experimental Configuration

The experimental setup is outlined in what follows. The population size N (per task) is kept the same for all algorithms, ensuring consistency. As stated in previous chapters, if a single-tasking algorithm utilizes a population size N for each of the K tasks, then a multitasking algorithm tackling the same K tasks is assigned a population size of $N \cdot K$. The general algorithmic settings for all the problems are:

1. Continuized unified representation with range $[0, 1]^{D_{unified}}$;
2. Evolutionary operators for MFEA-II, MFEA-I and CEA:
 - (a) SBX crossover with probability $(p_c) = 1$ and distribution index $\eta_c = 15$;
 - (b) Polynomial mutation with probability $(p_m) = 1/d$ and distribution index $\eta_m = 15$;
3. Probabilistic model for MFEA-II: variable-wise (marginal) normal distribution;
4. For the double-pole balancing problem:
 - (a) Population size (N): 100;
 - (b) Maximum function evaluations: 10,000;
5. rpm : Learned online for MFEA-II. For MFEA-I:
 - (a) 0.1 (tuned for best performance).

The implementations of the NES variants are obtained from [105]. Based on [106], the optimized set of parameters are found to be:

1. For SNES, the learning rates for the search parameters mean and standard deviation are $\eta_\mu = 1$ and $\eta_\sigma = 0.1058$, respectively.
2. For xNES, the learning rates for the mean and covariance components are $\eta_\mu = 1$ and $\eta_\sigma = \eta_B = 0.0017$.

We note that the aforementioned optimized NES parameters are close to default settings. Both NES variants incorporate rank based fitness shaping, as was suggested in [101].

6.2.2 Preliminaries of the Double-Pole Balancing Problem

In literature, the double-pole balancing problem (or inverted pendulum system) has been a standard benchmark for artificial learning systems [100, 107–110]. The problem involves balancing two poles (of different lengths) hinged on a cart, using a neural network controller, such that the poles do not fall beyond a predefined vertical angle and the cart remains within the bounds of a horizontal track. Typically, the length of the longer pole remains fixed to $1.0m$ while the length of the shorter pole l_s maybe varied.

FIGURE 6.1: Double-pole balancing problem

As substantiated in literature [100], the task of balancing both the poles become progressively harder as the poles assume similar lengths. In other words, the system becomes difficult to control as the length of the shorter pole l_s approaches to $1.0m$.

As such, it gets challenging (even for state-of-the-art optimizers) to solve such difficult control tasks from scratch (zero-state knowledge). Accelerating the acquisition of grounded knowledge by learning with many goals (tasks) has been a promising approach in the domain of reinforcement learning systems. It has been shown that while intentionally attempting to solve the harder tasks, other simpler tasks can be implicitly (unintentionally) solved [13]. *Furthermore, in this study we demonstrate that during multitasking, both simple and hard tasks can mutually benefit from one another as a unique consequence of the omnidirectional transfer of knowledge.*

In our experiments, we consider the Markovian case of the double-pole task. To elaborate, all state variables, including, the cart position- x , the cart velocity- $\dot{x}$, angle of longer pole- θ_1 , angular velocity of longer pole- $\dot{\theta}_1$, angle of shorter pole- θ_2 , and the angular velocity of shorter pole- $\dot{\theta}_2$, are available as inputs to the neural

network. The associated equations of motions and experimental parameters are based on the descriptions provided in [109, 111].

The dynamical system is evaluated using the fourth-order Runge-Kutta numerical integration scheme with step size $\tau = 0.01s$. That is to say, this is a simulation-based black-box optimization problem since the analytical form of the objective function is unknown. A simple feedforward neural network (FNN) controller, with 10 hidden neurons, is used. The optimization problem is about automatically tuning the weights of the FNN controller. Accordingly, the network outputs a force between $[-10, 10]N$ every 0.02 seconds. This force is then applied to the cart and the next state is updated, which is reintroduced as the new input to the neural network. The process continues until a failure occurs or the desired control action has been performed for the required amount of time. The amount of time before failure thus becomes the measurable fitness. The initial position of the longer pole is set to $\theta_1 = 1^\circ$ and the shorter pole remains upright ($\theta_2 = 0^\circ$). The track was set to 4.8m long. For the task to be solved, the angles of the poles have to be in the specified range $[-36^\circ, 36^\circ]$ for 10^5 time steps, where each step corresponds to 0.02s. This is equivalent to over 30 minutes of simulated time.

6.2.2.1 Omnidirectional Transfer in Multitask Optimization

In this particular study, we highlight the benefits of *omnidirectional* transfer in multitasking. As opposed to *sequential transfer* methods which tackle *one* task at a time through *unidirectional* knowledge transfers (from past to present), multitasking offers the flexibility of handling multiple tasks concurrently. With this in mind, we demonstrate that the omnidirectional genetic transfer regime of MFEA-II makes it possible for all tasks to gain maximum benefits from one another. To showcase this facet of multitasking, we refer to the numerical results contained in Table 6.1. Therein, three tasks of different short pole lengths l_s (with increasing difficulty levels) are considered. The percentage of successful runs (out of 30 independent runs) is used as a performance measure to compare MFEA-II against CEA.

According to the results, it can be seen that by tackling all the three tasks separately, the CEA can only solve task T_1 (27%) and fails to find suitable control

TABLE 6.1: Percentage success rate (out of 30 runs) of three different tasks. They are either solved separately (CEA), or merged into pairs or a triplet (MFEA-II).

Task	l_s	CEA	MFEA-II			
			(T_1, T_2)	(T_1, T_3)	(T_2, T_3)	(T_1, T_2, T_3)
T_1	$0.60m$	27%	30%	30%	-	47%
T_2	$0.65m$	0%	27%	-	27%	37%
T_3	$0.70m$	0%	-	7%	27%	17%

strategies for the more challenging variants T_2 and T_3 , within the allocated computational budget. respectively. On the other hand, the performance consistently improves if we tackle two distinct tasks simultaneously using MFEA-II. The dash symbol (-) in Table 6.1 implies the *absence* of a particular task in the MFEA-II. In addition to improving the success rate of task T_1 (30%), multitasking also helps to improve the performance of harder instances T_2 and T_3 when compared to single tasking. Eventually, pooling all the three tasks together (T_1, T_2, T_3) , significantly improves the overall performance of tasks T_1 (47%) and T_2 (37%). This highlights the fact that genetic transfers from complex tasks such as T_3 ($l_s = 0.70m$), can also help to boost the performance of simpler tasks T_1 and T_2 .

6.2.2.2 Some insights on Many Tasking

We extend our experimental setup to a series of six different tasks of different short pole lengths l_s , ranging from $0.45m$ to $0.70m$. Further, two state-of-the-art solvers SNES [102] and xNES [103] are included as additional baselines for comparison. The percentage success rate (out of 30 runs) of each of the algorithms, on each task is summarized in Table 6.2.

Overall, the numerical results reveal that the MFEA-II is superior to other methods, on almost all the tasks under consideration. While MFEA-I, SNES and xNES have demonstrated competitive performance on the first three tasks (T_1, T_2, T_3) , their performances rapidly drop as the length l_s increases. Apart from MFEA-II, the success rate of all other methods is significantly low on the increasingly challenging variants T_5 ($l_s = 0.65m$) and T_6 ($l_s = 0.70m$). MFEA-II has improved the success for the last two more difficult control tasks T_5 and T_6 , with 63% and 37% success respectively. On a closer look, it can be observed that MFEA-I has also performed better than the standalone solvers on the last two difficult task instances.

In fact, SNES and xNES are found to be unsuccessful in effectively balancing both the poles for T_6 with $l_s = 0.70m$.

TABLE 6.2: Comparison of various methods on different short pole lengths l_s , for the double-pole balancing problem. The Table shows the percentage success rate (%) out of 30 independent runs. Best entries are highlighted in bold.

Task	l_s	Success Rate (%)			
		MFEA-II	MFEA-I	SNES [102]	xNES [103]
T_1	$0.45m$	100%	80%	100%	97%
T_2	$0.50m$	100%	80%	77%	97%
T_3	$0.55m$	97%	60%	73%	100%
T_4	$0.60m$	83%	50%	27%	43%
T_5	$0.65m$	63%	20%	13%	10%
T_6	$0.70m$	37%	10%	0%	0%

FIGURE 6.2: Comparison of total number of function evaluations consumed by each of the algorithms in solving all the six different tasks.

As an important aside, the total number of function evaluations required by MFEA-II to solve all the six different tasks is measurably less than the rest of the solvers, as shown by Figure 6.2.

Further, Figure 6.3 shows the pairwise $rm p_{j,k}$ values learned between distinct task-pairs T_j and T_k , during the course of the optimization phase in MFEA-II. The curves highlight the existence of underlying inter-task synergies between 15 possible task-pairs of different short pole lengths l_s . Accordingly, the amount of transfers is regulated by the learned $rm p$'s in the RMP matrix, which are seen to vary across different stages of the optimization process. Recall that if $rm p$ is high, the frequency of transfer is high. In contrast, the frequency of transfer in MFEA-I remains fixed by the preset value of $rm p$.

Importantly, Figure 6.3 highlights that MFEA-II is able to autonomously distinguish similar tasks from dissimilar ones. To elaborate, it is worth noting that tasks

FIGURE 6.3: Pairwise *rmp*'s of MFEA-II, learned between constitutive tasks of different short pole lengths. The plots are averaged over 30 independent runs and the shaded regions span one-half standard deviation on either side of the mean.

with short poles of similar lengths are intuitively expected to have more synergy. Accordingly, from Table 6.2, it can be said that T_1 is more similar to T_2 than it is to $T_3/T_4/T_5/T_6$. Similarly, T_2 is more similar to T_3 than it is to $T_4/T_5/T_6$. Figure 6.3 shows that MFEA-II precisely captures this trend. In other words, as the expected inter-task similarity decreases, so do the learned *rmp* values.

6.3 Part II: Evolutionary Multitasking for High Fidelity Optimization: A practical study of MO-MFEA-II

One of the key beneficiaries of our proposed MO-MFEA-II framework is expected to be in the domain of high fidelity optimization settings. The fact that real world high fidelity numerical simulations can take anywhere between several minutes to several

hours for a single run, the use of traditional single-task optimization approaches still remain too costly. On the other hand, knowledge exploitation from cheaper optimization problems to cut down the computational costs of (related) expensive problems has shown to be promising [20, 51]. In this section, we demonstrate the efficacy of MO-MFEA-II in leveraging knowledge transfers from low fidelity tasks to speed up the performance of higher fidelity tasks.

FIGURE 6.4: Multiple UAVs in action.

In particular, we consider multi-objective optimization of the unmanned aerial vehicle (UAV) path planning problem. Before we proceed to the details of our experimental settings, we present a brief overview of the UAV path planning procedure. UAVs have emerged as promising tools for autonomous inspection of large geographical areas and performing missions in hazardous environments, such as military surveillance, nuclear power plant operations and outer-space exploration [112, 113]. Autonomous navigation by UAVs through unknown environments is a challenging task that requires safe and efficient path planning strategies, such that operational risks (in the presence of hazards including obstacles or environmental factors) are minimized. The risks associated with UAVs may vary from collisions to loss of control. Collisions are usually caused by static or dynamic obstacles while loss of control can result from system failure, navigating beyond signal range (e.g. radio, WiFi or GSM networks) or adverse environmental conditions (e.g. bad weather or bad GPS reception) [114]. In our experimental study, we consider a realistic scenario with multiple UAVs operating in a specified region in the southwest of Singapore. The operational environment is characterized by different types of hazards and environmental factors (e.g. signal strength and weather conditions).

For our simulations, a data-driven multitask Gaussian process (GP) model of the environment is created. The probabilistic predictions from such a GP model are

then used to define a *path-integral* risk metric, which gives the probability of an unsafe outcome along a UAV's traversed path. In what follows, background details of safety risk assessment in UAVs and the associated path efficiencies, the proposed optimization problem as well as GP modeling of the observed environment are briefly described - as per [114].

6.3.1 Multi-objective Risk Averse UAV Path Planning

The problem of minimizing operational risk for multiple UAVs in uncertain environments can be defined in a multi-objective optimization framework, addressing the trade-off between the path-integral risk measure and path-efficiency (travel distance). The considered MOOP can be represented as follows,

$$\text{minimize } (Risk, Distance), \quad (6.1)$$

where the first objective corresponds to minimizing the probability of an unsafe event occurring for at least one UAV, while the second objective accounts for the total distance travelled by all the UAVs from their respective origins to destinations.

With regard to the risk measure, the path-integral over a continuous path for a particular UAV is evaluated through a discretization of its trajectory. We note that the runtime of evaluating the risk metric along a finite discretized path is highly dependent on the discretization step size. To elaborate, a smaller step size implies numerical integrations of high fidelity, making the optimization process computationally expensive. Such high fidelity simulations, albeit more accurate, can take up several minutes or even hours. On the other hand, lower fidelity simulations with larger discretization step sizes can be much faster, although they do not offer the same level of accuracy in the risk estimations.

6.3.1.1 Simulation details

In this demonstration, we consider five UAVs operating in a region of about $10 \times 7 \text{ km}^2$ in the southwest of Singapore as shown by Figure 6.5a. Notice that the region includes geofence (indicated by red circumferences) which UAVs cannot

FIGURE 6.5: **Left:** Geographical region of interest. Minimum distance (dotted lines) and minimum risk (solid lines) trajectories for 5 different UAVs in an area with different signal strength, wind speeds and 1 geofence (red circumference). **Right:** Trajectory Γ defined by the cubic spline interpolation of 4 control points and the start (S) and destination (E) of a particular UAV's mission.

FIGURE 6.6: Sample analysis of 5 UAVs flying in an area with different signal strength and wind conditions.

enter. The trajectories of UAVs are defined using cubic spline interpolation, guided by a predefined number of control points [115].

As per Figure 6.5b, a cylindrical reference system is used, with the axis defined by the straight line joining the start and end of the path. Herein, the longitudinal location of the control points along the axis is pre-defined (e.g. equally spaced along the axis). Accordingly, an optimized path can be generated by selecting optimal set of parameters δ_k and φ_k for each k^{th} control point ($k = 1, \dots, cp$), representing the perpendicular distance and angle of rotation of a particular control point with respect to the axis, respectively. Notably, the risk metric (as per E.q.(6.1)) is calculated along the entire path (using a path-integral formulation) and not only using the described control points.

Further, we introduce two environmental factors that translate into hazards: 1) signal strength (4G signal) and 2) the weather conditions (defined by different wind speeds) as illustrated by Figure 6.6. Given the presence of such hazards, Figures 6.6a and 6.6b analyze the trade-off between path efficiency (distance) and the associated risks of 5 different UAVs. Notice the minimum distance (dotted line) and minimum risk (solid lines) trajectories generated on color maps representing 4G signal strength and wind speed, respectively.

There exist a couple of other constraints such as maintaining a minimum separation distance between UAVs and flying within specified altitude boundaries. For full experimental details, the reader is referred to [114].

6.3.2 Experimental Configuration

The experimental setup is outlined in what follows. The performance of the adaptive MO-MFEA-II is evaluated against its single tasking baseline, i.e., NSGA-II. Note that due to constant parameter tuning hurdles and its susceptibility to negative knowledge transfers (as previously emphasized in Chapter 5), the existing MO-MFEA is omitted from this particular set of expensive optimization studies.

Further, to ensure a fair comparison, the population size N (per task) is kept the same for both algorithms. For all the practical test problems under consideration, the following general settings are applied:

1. Continuous unified search space $[0, 1]^D$;
2. Evolutionary operators used for MO-MFEA-II and NSGA-II are:
 - (a) SBX crossover with probability $(p_c) = 1$ and distribution index $\eta_c = 10$;
 - (b) Polynomial mutation with probability $(p_m) = 1/D$ and distribution index $\eta_m = 10$;
3. Probability model in MO-MFEA-II: Normal distribution;
4. Population size (N): 20;
5. Maximum number of generations: 250;
6. *RMP* is learned online for MO-MFEA-II.

Note that no uniform crossover like variable swap is applied between offspring to ensure parent-centric crossover operations with minimum chromosome disruptions [90], thus allowing us to clearly observe the effects of inter-task implicit genetic transfers.

Table 6.5 shows two different experimental configurations used to demonstrate the efficacy of MO-MFEA-II for high fidelity multi-criteria optimization of UAV trajectories.

TABLE 6.3: Practical study experimental configurations. Different task instances are generated by defining different discretization step sizes (d_{step})

	Tasks	
	Main task	Helper task(s)
Configuration 1	$T_1 : d_{step} = 500m$	$T_2 : d_{step} = 1000m$
Configuration 2	$T_1 : d_{step} = 500m$	$T_2 : d_{step} = 750m$ $T_3 : d_{step} = 1000m$

As shown, different instances of tasks (representing different fidelity levels) are generated using different discretization step sizes (d_{step}) associated with the path-integral risk measure. In both configurations, the high fidelity task T_1 represents the main task of interest, while the low fidelity tasks T_2 and T_3 are considered as helper tasks. Since all the specified tasks essentially solve the same MOOP, but with different fidelity levels, there inherently exists underlying complementarities that can be exploited via multitasking.

While the baseline NSGA-II solves the main tasks in isolation, the MO-MFEA-II pools together the main and helper task(s) for simultaneous optimization. The hypervolume metric [116] is used as an indicator to measure the quality of solutions obtained by the two different algorithms.

Figure 6.7a shows the convergence trends of T_1 mapped against the total run time for experimental Configuration 1. As is seen, the MO-MFEA-II leads to a significant boost in the optimization search during the initial stages of the main task. Specifically, to achieve a hypervolume measure of 0.415 for the high fidelity task T_1 , MO-MFEA-II takes 33% less time compared to NSGA-II. The corresponding learned rmv values are depicted in Figure 6.7b. As shown, T_1 experiences high frequency of transfers from its lower fidelity helper task T_2 , thereby boosting the overall convergence trend.

FIGURE 6.7: **Left:** Convergence trends (averaged over 10 independent runs) achieved by NSGA-II and MO-MFEA-II for experimental Configuration 1. **Right:** Pairwise rmp 's learned between tasks T_1 and the helper task T_2 over total run time. The shaded regions span one half standard deviation on either side of the mean.

FIGURE 6.8: **Left:** Convergence trends achieved by NSGA-II and MO-MFEA-II for experimental Configuration 2. The MO-MFEA-II employs two helper tasks, respectively. **Right:** Pairwise rmp 's learned between tasks T_1 and the helper tasks T_2 and T_3 . The shaded regions span one half standard deviation on either side of the mean.

Next, in experimental Configuration 2, we demonstrate the effects of introducing an additional low fidelity helper task T_3 . As shown in Figure 6.8a, the overall convergence trend of the high fidelity task T_1 (blue curve) is further boosted relative to the former experimental setup with only one helper task (shown by the red curve in Figure 6.8a). The performance enhancement is attributed to the learned *rmf* curves given by Figure 6.8b. Therein, T_1 is observed to be experiencing knowledge transfer from both helper tasks T_2 and T_3 , with the learned rate of transfer from T_2 being much more frequent; this agrees with our intuition from Table 6.5 that T_1 is more similar to T_2 than it is to T_3 . Notably, from Figure 6.8b, it is also seen that T_3 helps T_2 (see $T_2 - T_3$ *rmf* curve), which in turn helps boost the optimization performance of the high fidelity task T_1 .

6.4 Part III: Agile Reconfiguration of Deep Models via Evolutionary Multitasking

Since its inception, Deep Learning has been transforming our lives in various aspects; from autonomous cars, virtual assistants and advanced healthcare to facial and speech recognition, etc. The evolution of new paradigms such as Cloud Computing are now adept at orchestrating deep learning at scale, offering powerful computing capacities to train and access gigantic (overparametrized) deep models remotely on the cloud from almost anywhere. However, these huge (compute intensive) models bring a unique set of challenges, particularly when attempting to access them on a variety of portable smart devices with limited compute powers - for which local embedded processing near the edge sensors is preferred over the cloud due to privacy or latency concerns. Notably, several key metrics/criteria for embedded deep learning have to be considered such as model accuracy, memory/energy consumption, inference time, throughput/latency, carbon footprint, etc. To this end, this section proposes a multi-criteria optimization search capable of facilitating agile reconfiguration of gigantic deep networks into diverse compact master models, customized for unique smart device requirements and user preferences. More importantly, a multitasking framework is adopted as an enabler to process multiple reconfiguration tasks concurrently, much faster than standard approaches handling one task at a time. The efficacy of the proposed work is showcased on a set of common image recognition dataset.

6.4.1 Premise

The emergence of deep learning [117] is largely credited for the latest boom in artificial intelligence (AI) research today. Undoubtedly, the available computing power and unprecedented increase of data that the advent of internet has led to recently, have been the main triggers. Cloud computing [118], the modern day democratization of computing, has revolutionized the field of deep learning - offering computing capacities to small and medium enterprises that previously were only available to research labs owned or funded by wealthy tech companies (e.g. Google, Amazon etc). The scope for seamless access to large data processing centers on the cloud, coupled with edge systems prioritizing network traffic and speed [119], have made deep learning a practical reality.

The convenience of cloud resources has numerous advantages. Not only does it orchestrate deep learning at scale, it easily caters for potentially thousands of unknown or widely varying compute requirements of Internet of Things (IoT) devices hosted at disparate edge computing sources. The number of IoT devices in circulation today is already staggering, with ongoing demand for faster, more efficient services and content delivery. With the evolution of such ubiquitous computing, the cloud caters for a wide variety of IoT devices, ranging from powerful personal computers (PCs) like desktops and laptops to portable smart devices and home appliances with limited compute powers.

FIGURE 6.9: Elastic and agile cloud computing platform simultaneously catering for small, unknown or variable compute requirements at the edge. As per unique device requirements and user-specific preferences, diverse specialized (master) models such as 1-class, 2-class or multi-class output can be queried from the cloud hosting a gigantic, pretrained, generalized (Jack of all trades) deep model. These light-weight master models can then be conveniently accessed on portable devices with lesser compute powers.

Figure 6.9 highlights a cloud-based scenario where many (diverse) compact and resource efficient deep learning models (extracted from a huge pretrained deep model hosted on the cloud) are needed to be accessed simultaneously at disparate locations and deployed on the fly to the edge. While most IoT devices on the edge are akin to PCs, they are not regular computing devices designed to perform multiple computationally intensive functions, but are usually resource constrained with limited memory and compute powers. Depending on the available compute resources, these devices are able to intelligently query and respond to unique user requests in a manner that exploits the appropriate memory limitations such that compact deep models can be conveniently accessed at the edge.

Interestingly, edge IoT devices offer the added flexibility to specialize to diverse model types of variable sizes, each tailored for different user requirements. That is, one is not always limited to an overparameterized (gigantic) generalized deep model, but also has the luxury to access models specializing in particular domains or classes of interest. For instance, consider an image recognition scenario whereby a deep VGG-16 [121] convolutional neural network (CNN) is trained on ImageNet dataset consisting of all available 1000 output classes [122]. Such a pretrained generalized deep model is often referred to as a *jack of all trades, master of none*. It is however contended that end-users today may not always be interested in the jack of all trades (JOAT) model, but rather prefer to seek for the appropriate *masters* according to their respective needs. To elaborate, one may not be interested in all the available 1000 classes, but often in only one or a subset of classes. Since only a subset of class labels are of interest, most of the model parameters of a generalized deep network thus become trivial and can be deactivated, paving way for the extraction of light-weight resource efficient deep models suitable for portable IoT devices. By hosting gigantic JOAT deep models remotely and extracting resource efficient master models whenever requested, the cloud servers efficiently free up the memory and computing power of individual IoT edge computing devices. Such a cloud-based setting ensures a faster, seamless experience for their customers, who expect to have access to their content and applications on demand anywhere at any time.

With the above in mind, the main objective of this section is thus to apply methods capable of facilitating agile reconfiguration of gigantic deep networks into diverse

compact master models, customized for unique IoT device requirements and user preferences. To elaborate, our key contributions are two fold.

1. *Firstly, to accommodate for diverse resources requirements of various different computing devices at the same time. To this end, a multi-criteria optimization search is employed and the trade-off between memory footprint and accuracy of acquired solutions (i.e., reconfigured models) is measured via a Pareto front (popularly known to be a graphical representation of optimal solutions used to evaluate trade-offs between multiple conflicting criteria of interest [81]).*
2. *Secondly, to offer the flexibility for simultaneous extraction of many different "master" models specializing for distinct use cases or problem types such as 1-class, 2-class or multi-class output etc. Accordingly, an evolutionary multi-tasking framework [21] is adopted as an enabler to process multiple reconfiguration tasks concurrently, much faster than standard evolutionary approaches handling one task at a time.*

The practicality of using the aforementioned multi-criteria multitask optimization approach (i.e., via MO-MFEA-II in this case) for deep model reconfiguration is expected to yield substantial cost saving benefits, particularly in modern day cloud computing platforms where speed and accuracy are at premium.

6.4.2 Multi-criteria deep reconfiguration

All in all, efficient deep learning computing requires algorithm and hardware co-design to enable specialization. In what follows, we present formulations describing the concept of deep model reconfiguration for the purpose of specialization, which forms the crux of this particular work.

To begin with our discussion, we first highlight two important criteria that need to be considered during the process of deep reconfiguration for specialization. These are:

1. Model performance - the performance (accuracy) of reconfigured (i.e., master) models need to be preserved during the reconfiguration search.

2. Memory footprint - the efficiency of the reconfigured models needs to be guaranteed to deploy them on resource-constrained computing devices.

With the above, we can thus address the task of reconfiguring a deep model as a multi-criteria optimization problem.

Consider a CNN $\mathcal{L}$ with p convolutional layers $\{\mathcal{L}_1, \dots, \mathcal{L}_p\}$. We define p sets of convolution filters $\mathbb{F} = \{\mathbb{F}_1, \dots, \mathbb{F}_p\}$ for these layers. For the i^{th} convolutional layer, its associated filters are denoted as $\mathbb{F}_i \in \mathbb{R}^{H_i \times W_i \times C_i \times N_i}$, where H_i and W_i are the height and width of filters, respectively. Moreover, C_i is the channel size and N_i is the number of filters in i^{th} convolutional layer.

Further, consider the dataset $\mathcal{D}$ consisting a set of Φ available output classes $\mathbb{Q} = \mathbb{Q}_1, \dots, \mathbb{Q}_\Phi$, on which the deep CNN $\mathcal{L}$ is trained. Recall that during specialization, we are not always interested in all of the available classes, but often only a *subset* of classes $\mathbb{S}$, s.t. $\mathbb{S} \subseteq \mathbb{Q}$. Accordingly, a reconfiguration task is formulated as the following bi-criteria/objective optimization problem.

$$\begin{cases} \min_{B_i, \dots, B_p} \frac{1}{|\mathbb{S}|} \sum_{s=1}^{|\mathbb{S}|} \|E(\mathcal{X}_s, \mathcal{Y}_s, \hat{\mathbb{F}}) - \mathcal{Y}_s\|_F^2, \\ \min \frac{\sum_{i=1}^p \|\mathbb{F}_i \circ B_i\|_1}{\sum_{i=1}^p \|\mathbb{F}_i\|_1} \end{cases} \quad (6.2)$$

with $\mathbb{F}_i \in \mathbb{R}^{H_i \times W_i \times C_i \times N_i}$, $\hat{\mathbb{F}} = \{\mathbb{F}_1 \circ B_1, \dots, \mathbb{F}_p \circ B_p\}$,
 $B_i^{(:, :, :, n)} \in \{0, 1\}, \forall i = 1, \dots, p$,

where $B_i^{(:, :, :, n)}$ denotes the n^{th} filter in the i^{th} convolutional layer.

The first criterion in Eq. (6.2) shows that only the selected classes $s \in \{1, 2, \dots, |\mathbb{S}|\}$, from the superset $\mathbb{Q}$, are considered for calculating the network error E during the reconfiguration search. Note that $\mathcal{X}_s$ and $\mathcal{Y}_s$ represent the training sample and the corresponding ground truth of each selected class $s, \forall s = 1, \dots, |\mathbb{S}|$. Further, the second criterion adjusts the ratio of activated filter weights of the reconfigured model to that of the original model (in which all the filter weights are active). Importantly, in the context of reconfiguration, $B_i^{(:, :, :, n)} = 1$ implies that the weights of n^{th} filter in the i^{th} convolutional layer are active. On the other hand, $B_i^{(:, :, :, n)} = 0$ implies that the filter weights are deactivated, i.e., the weights are zeroed (which

can be discarded). For instance, the tensor B populated with many zeros during the reconfiguration search will result in deactivating (zeroing) most filters of the original deep model. As a result, the specialized model would only have a few active filters; thus, the extracted master model would entail a much smaller memory footprint with respect to the original deep model. It is noteworthy that the second criterion in Eq. (6.2) serves as a means to approximate total memory consumption of desired master models. Other variations such as simply counting the total number of filters activated (by ignoring filter sizes, input-output channels, etc) or even sophisticated strategies that accurately calculate the memory footprints (i.e., exact number of bytes), can also be conceived in the above formulation.

6.4.3 Multi-master multi-criteria optimization

Cloud computing ecosystems have the potential to benefit both providers and users, allowing deep models to be dynamically provisioned and reconfigured to deliver services in a scalable (i.e., anywhere at any time) manner. In light of the above, a matter of increasing importance has been the design of algorithms that are capable of processing incoming streams of data from multiple tasks, identifying relationships between them and solving them simultaneously via fruitful knowledge transfers. In what follows, we propose a multi-tasking approach capable of handling multiple different multi-criteria reconfiguration tasks at once - which is the key contribution of this section.

6.4.3.1 Multitasking approach - a brief recap

We begin with a quick recap of a multitask setting that deals with the co-existence of multiple related search/optimization problems, such that knowledge can be transmitted among them. Consider a series of K (related) optimization tasks that are labelled as $T_1, T_2, \dots, T_K$. Without loss of generality, all tasks (that are treated as self-contained multi-criteria reconfiguration tasks as per Eq. (1)) are assumed to be minimization problems. The k^{th} task, denoted as T_k , has design space $\mathbf{X}_k$ on which the vector-valued M objective (criteria) function is defined as $F_k : \mathbf{X}_k \rightarrow \mathbb{R}^{M_k}$, where M_k is the number of criteria in the objective function vector. In such a setting, a multitasking approach aims to simultaneously navigate the

design space of all tasks, assisted by the potential for fruitful knowledge transfer, so as to efficiently deduce $\operatorname{argmin}\{F_1(x), F_2(x), \dots, F_K(x)\}$.

6.4.3.2 Neuroevolutionary approach to deep reconfiguration

The present work proposes to use a neuroevolutionary-based search for the task of reconfiguring a deep model. Inspired from the evolution of biological nervous systems in nature, neuroevolution is a powerful machine learning technique that applies evolutionary algorithms (EAs) for architecture search of neural networks [123, 124]. In contrast to other neural network learning methods, neuroevolution is considered highly general, facilitating learning without explicit targets, with only sparse feedback, as well as with arbitrary neural models and network structures. Backed by several benefits of EAs, neuroevolution has shown various success stories in the general theme of deep learning [125, 126].

6.4.3.3 Standard deep reconfiguration

The typical steps associated with reconfiguring a deep CNN during specialization is provided by the flowchart in Figure 6.10. During the course of the reconfiguration search, filters are iteratively activated/deactivated (i.e., switched on and off). Notably, the respective filters to be activated/deactivated is dictated by an evolutionary search. That is, a binary vector is employed to represent whether the filters takes effect or not; a value of 1 implies that a filter is active. Else, it is deactivated (i.e., zeroed). Further, to recover the parameters (weights) of a reconfigured (specialized) model, it is further fine-tuned (retrained). The accuracy of the reconfigured model is validated by evaluating on the available dataset. This process repeats until a desired reconfigured model is found or a termination condition is met.

6.4.3.4 Deep reconfiguration via evolutionary multitasking

Being laser-focused on a single task at any given time often casts barrier in scaling to numerous incoming requests at once. The ability to rapidly solve many complex

FIGURE 6.10: A standard neuroevolutionary approach to deep model reconfiguration.

tasks, with the scope of knowledge transfers from related tasks, has been a long-standing goal of artificial intelligence systems targeting improved problem-solving efficacy. Along similar lines, evolutionary multitasking has recently emerged as an efficient method to solve multiple related tasks simultaneously via genetic knowledge exchange among them [11, 21, 29, 61, 62, 78, 80]. With this in mind, we thus propose to exploit the idea of evolutionary multitasking to efficiently handle multiple reconfiguration tasks at once – allowing us to rapidly extract multiple different master models simultaneously.

Figure 6.11 illustrates the proposed idea. As shown, multiple reconfiguration tasks T_1 , T_2 and T_3 are tackled concurrently in contrast to standard evolutionary approaches that seek to solve each task in isolation (i.e., one at a time). Herein, each of the reconfiguration tasks conduct search on a pretrained deep model with the aim of extracting distinct master models unique to a particular user’s requirement; i.e., specializing to different domains of interest. For instance, tasks T_1, T_2 and T_3 could specialize to a 1-class, 2-class, and 3-class problem, respectively – with all of these specialized class labels selected from the superset of all available classes in the original deep model.

Since the respective specialized models are extracted from a common pretrained deep model, there inherently exists underlying complementarities between each of the reconfiguration tasks that can be exploited via multitasking. Notice that for the case of a pretrained CNN model, the convolutional layers capture general, low-level features that are applicable across all types of images, even from different domains — such as edges, patterns, gradients. Thus, these universal low-level features (in

FIGURE 6.11: Multiple reconfiguration tasks (aimed at switching on and off filters of a deep CNN) are handled simultaneously via evolutionary multitasking. The scope for spontaneous knowledge transfers under the simultaneous occurrence of multiple tasks unveils the benefits of multitasking.

the form of convolutional filters) can be shared between multiple different specialized models during the reconfiguration search. In such settings, the notion of evolutionary multitasking is thus deemed invaluable; offering the scope for knowledge (encoded as genetic materials) exchange between multiple reconfiguration tasks to facilitate simultaneous exploitation of the respective search landscapes, thereby efficiently evading any obstacles to converge faster. Notably, the scope for inter-task genetic transfers radically reduce the cost of re-exploring overlapping regions of the search space, thus leading to significant performance speedup, in contrast to standard evolutionary approaches with no knowledge transfers. In particular, the practicality of evolutionary multitasking in resource-constrained environments is indeed of significant value, as shall be shown experimentally in Section 6.4.4.

As an aside, notice (as per Figure 6.11) that any available multi-criteria based multitasking algorithm can be applied for deep reconfiguration. That said, the *MO-MFEA-II* is considered herein for the subsequent pilot studies.

6.4.4 Experimental Study

In this section, experimental results are presented that showcase the efficacy of evolutionary multitasking MO-MFEA-II algorithm in contrast to standard single tasking NSGA-II, that solves one reconfiguration task at a time.

The experimental set-up is outlined in what follows. To ensure consistency, the population size N (per task) is kept the same for all algorithms. For all the test problems under consideration, the following general settings are applied:

1. Continuized unified representation with range $[0, 1]^{D_{unified}}$;
2. Evolutionary operators for MO-MFEA-II, and single tasking NSGA-II
 - (a) SBX crossover with probability $(p_c) = 1$ and distribution index $\eta_c = 20$;
 - (b) Polynomial mutation with probability $(p_m) = 1/d$ and distribution index $\eta_m = 20$;
3. Population size (N): 30
4. Maximum number of generations: 250;

TABLE 6.4: IoT Device Categories

	Device types	Memory allocation
Category₁	Desktop PC's, workstations	High
Category₂	Laptops, tablets	Medium
Category₃	Smart phones, smart watches, hand-held games etc.	Low

6.4.5 Dataset

The pilot set of experiments considers a pretrained LeNet deep network that needs to be reconfigured. The dataset used in the following example is MNIST. Further, the LeNet model consists of four convolutional layers of size $20 \times 1 \times 5 \times 5$, $50 \times 20 \times 5 \times 5$, $500 \times 50 \times 4 \times 4$, and $10 \times 500 \times 1 \times 1$, respectively, and is called a

fully convolutional LeNet. Since filters in all layers except the output layer can be reconfigured (switched on and off), the search space for finding a subset of less important filters is 2^{570} . In addition, the MNIST dataset contains 60000 images as the training set and 10000 images as the test set, which are gray handwritten digits of ten classes in the size of 28×28 .

A reconfiguration task (as per Section 6.4.2) is thus to use multi-objective evolutionary search to extract compact specialized models by simply activating/deactivating the appropriate filters (which shall be eventually pruned prior to being deployed on various different portable devices enlisted in Table 6.4). The choice of specialized models is user dependent. Table 6.5 shows the experimental configuration based on the MNIST dataset comprising of 10 output classes. Herein, each of the reconfiguration tasks is to search for the appropriate one-against-all (OAA) master model based on the specified class label. For instance, a reconfiguration task T_1 seeks to find a master model that specializes in classifying whether an input image is digit 0 or not.

TABLE 6.5: Reconfiguration tasks and the corresponding class labels

Reconfiguration task	Master model {OAA class label}
T_1	{0}
T_2	{1}
T_3	{2}
T_4	{3}
T_5	{4}
T_6	{5}
T_7	{6}
T_8	{7}
T_9	{8}
T_{10}	{9}

The reconfiguration search efficacy of MO-MFEA-II is compared against the baseline single tasking NSGA-II. Experimental results are reported in Figure 6.12 and Table 6.6, respectively. As shown by Figure 6.12, the MO-MFEA-II facilitates faster convergence(reconfiguration) than NSGA-II; thus, allowing much faster extraction of each of the 10 different master model variants. Table 6.6 presents the accuracies of each of the master models (with varying memory allocations (measured as $x\%$ of the total memory of JOAT)) against the original gigantic JOAT model. As is clear, the specialized master models have higher inference accuracies.

FIGURE 6.12: Convergence curves for all tasks $T_1 - T_{10}$. The bottom right plots show the learned inter-task *rmp*'s between distinct task pairs.

6.5 Chapter Summary

In this Chapter, the practicality of the MFEA-II was demonstrated experimentally in the context of complex controller design tasks. The utility of the online knowledge transfer scheme in the face of the present day offline approach, was clearly revealed. It is noteworthy to highlight that the data-driven MFEA-II is automatic, alleviates human intervention (for ascertaining inter-task synergies), and yields consistently superior performance in comparison to some state-of-the-art optimizers. Moreover, the practical study herein not only shows the scalability of the proposed method to many-tasking (i.e., upto six tasks at once), but also underscores the advantages of (regulated) omnidirectional knowledge transfers as well as the boon of intentional-unintentional problem solving capacities.

In addition to the above, a practically useful application was also found to be in the domain of multi-objective multi-fidelity optimization. The simulation results highlight the ability of MO-MFEA-II to exploit relevant knowledge from low fidelity (cheaper) tasks to quickly optimize high fidelity tasks, thereby leading to significant cost saving benefits. Further, the empirical evidence in this chapter also shows

TABLE 6.6: MNIST Example: Master Model accuracies vs Jack of all trade model

Reconfiguration		Master model(s)		Jack of all trades model	
Task	Category	%Memory	Accuracy	%Memory	Accuracy
T_1	1	80	99.26	Full	99.03
	2	60	99.25		
	3	40	99.16		
T_2	1	80	99.25		
	2	60	99.27		
	3	40	99.21		
T_3	1	80	99.28		
	2	60	99.15		
	3	40	99.18		
T_4	1	80	99.20		
	2	60	99.20		
	3	40	99.18		
T_5	1	80	99.25		
	2	60	99.10		
	3	40	99.12		
T_6	1	80	99.28		
	2	60	99.20		
	3	40	99.11		
T_7	1	80	99.25		
	2	60	99.25		
	3	40	99.25		
T_8	1	80	99.12		
	2	60	99.14		
	3	40	99.11		
T_9	1	80	99.25		
	2	60	99.25		
	3	40	99.25		
T_{10}	1	80	99.21		
	2	60	99.23		
	3	40	99.11		

the potential benefits of incorporating multiple low fidelity tasks for performance enhancement of the primary high-fidelity task of interest.

Finally, this chapter highlights the efficacy of evolutionary multitasking in facilitating rapid reconfiguration of generalized (yet overparameterized) deep neural networks into multiple compact specialized models at once. The practicality of knowledge transfers via the MO-MFEA-II algorithm has shown substantial improved problem solving and cost-saving benefits in the otherwise expensive affair of deep learning. In contrast to existing works in the literature, the key contributions and novelties of our proposed work are enumerated as follows.

- Firstly, to accommodate for diverse resources requirements of various different computing devices at the same time. To this end, a multi-criteria optimization search is employed and the trade-off between memory footprint and accuracy of acquired solutions (i.e., reconfigured models) is measured.
- Secondly, an evolutionary multitasking framework is adopted, offering the flexibility for simultaneous extraction of many different *master* models specializing for distinct use cases or problem types such as 1-class, 2-class or multi-class output, etc. Note that in contrast to standard neuroevolutionary approaches handling one task at a time, evolutionary multitasking has shown to dramatically reduce the overall search/optimization run time - *a key facet that makes the framework ideal for cloud/edge ecosystems where speed and accuracy of user demands are at premium.*

While the proposed work currently considers a standard bi-criteria optimization (i.e., model accuracy and memory) for deep reconfiguration, additional criteria such as energy consumption, carbon footprint, complexity of the learning algorithm, data privacy preservation, model inference time, etc., shall also be taken into account in future studies. The extension of the current work to various other domains (beyond image recognition) such as natural learning processing (NLP) remains to be a foreseeable future work as well.

Chapter 7

Conclusions and Future Work

This thesis proposed novel adaptive evolutionary multitasking frameworks (i.e., MFEA-II and MO-MFEA-II) to combat any deleterious effects of negative inter-task knowledge transfers plaguing the existing MFEA variants; thus, ensuring improved convergence characteristics.

Chapter 3 conducted a thorough theoretical analysis of a generic population-based multitask optimization framework, focussing on how *inter-task* knowledge transfers can actuate and subsequently validate its global convergence guarantees.

Chapter 4 unveiled the challenges of the present day MFEA, particularly highlighting its susceptibility to predominantly negative knowledge transfers - a known drawback which often tends to impede the overall convergence characteristics. Accordingly, the MFEA-II, a successor to the present day MFEA was proposed. Notably, the MFEA-II learns inter-task relationships based on overlaps in the probabilistic search distributions derived from data generated during the course of multitasking - and accordingly adapts the extent of genetic transfers online. The efficacy of MFEA-II was shown in a wide range of synthetic discrete and continuous single-objective benchmark functions.

The cognizant MO-MFEA-II, an extension of MFEA-II to the domain of multi-objective optimization problems was put forward in Chapter 5. It is noteworthy that this particular work was among the first to utilize probabilistic modeling to capture inter-task relationships between multi-objective optimization tasks in the context of evolutionary multitasking; i.e., a data-driven approach to explicitly

guard against negative transfers plaguing the present day multi-objective evolutionary multitasking engines. Similar to MFEA-II, the MO-MFEA-II also showcased awareness as to when and how much knowledge to transfer between multiple (related) multi-objective optimization problems.

In Chapter 6, the practicality of the aforementioned frameworks was substantiated via practical case studies from the reinforcement learning, high-fidelity optimization and deep learning domains. Therein, the benefits of adaptive knowledge transfers in evolutionary multitasking were generally quite clear. These include scalability to many-tasking, omnidirectional knowledge exchanges, the boon of intentional-unintentional problem solving capabilities, improved and faster convergence characteristics as well as cost-saving advantages in expensive optimization settings (under resource constrained environments).

7.1 Future Research Directions

While the current thesis may seem to have established a solid contribution within the context of adaptive multitasking, the relative youth of this field has left several challenges and research niches still insufficiently addressed. The following sections enumerate a series of open research directions for future considerations.

7.1.1 Scalability to many-tasking

Although the proposed MFEA-II and MO-MFEA-II frameworks are adept at capturing inter-task relationships across several tasks at once, it is noted that the associated learning procedure becomes computationally demanding as the number of tasks grows increasingly large. This particular issue is further amplified in resource constrained [networked] environments that heavily depend on efficient optimization algorithms for multitasking.

In the literature, there have however been some recent works to efficiently handle *many-task* optimization [97, 127]. While the direct scalability of MFEA-II/MO-MFEA-II to many-tasking can be achieved through simplifications such as reduced

frequency of *RMP* (i.e., the transfer parameter matrix) learning, this study remains a foreseeable future work.

7.1.2 Improved genetic transfers and solution representations

Standard evolutionary multitasking engines have generally invested in two essential modes of information transfers between tasks: explicit and implicit genetic transfers; the notion of genetic transfers (be it direct or indirect) has its own limitations. Over the recent years, the success of multitask optimization is not solely attributed to finding quality solutions to the problems, but also finding *representations* of the solutions for each problem that are more efficient for conveying knowledge transfer among tasks.

Recall that the primary mode of knowledge transmission in the current works (i.e., MFEA-II and MO-MFEA-II) is through implicit genetic transfers, facilitated by inter-task crossovers between solutions belonging to different (related) tasks. Further, these are purely data-driven methods that learn the underlying overlaps/relationships between search distributions (within the naive solution spaces) of independent tasks (preferably from the same domain) such that negative transfers are kept at bay. Essentially, these algorithms simply prohibit the scope for any knowledge transfers while tackling multiple tasks from different domains that may bear little or no apparent inter-task similarities - as direct genetic transfers under such settings are often deemed impractical.

However, recent studies have revealed that correlations between such tasks may still emerge through appropriate search space transformations (i.e., via linear/non-linear domain adaptation), thereby opening doors for fruitful knowledge exchange. These advancements further unleash the possibilities into fusing the ingredients from subspace learning alongside modern day adaptive solvers (like MFEA-II etc) to span a new generation of intelligent evolutionary multitasking frameworks for complex optimization settings - particularly in realistic scenarios faced with thousands (if not millions) of tasks from various different domains at once.

7.1.3 Extensions to constrained optimization

Further, the proposed works in this thesis have demonstrated various success stories in solving multiple optimization problems simultaneously, encompassing discrete, continuous, single-, as well as multi-objective optimization. Herein, it is worthwhile to mention that the tasks of interest have primarily focussed on unconstrained optimization problems.

Over the years, evolutionary multitasking has exhibited equal triumph in handling complex optimization tasks with various [dynamic] constraints as well. Accordingly, extensions of MFEA-II/MO-MFEA-II to constrained optimization settings shall also be considered in the near future.

7.1.4 What lies further ahead in the real world?

Beyond the above algorithmic advancements, it is worthwhile to advocate for other ambitious avenues embracing multitasking in optimization problems close to practical realities. Looking further ahead, emerging platforms such as cloud computing and the internet of things will offer large-scale data storage and seamless communication facilities [118, 122, 128], thereby making it possible for embedded optimization solvers to take advantage of the knowledge (data) made available by related tasks. In such settings, the general principles of multitasking are expected to play a key role in simultaneously solving multiple tasks (both more efficiently and effectively) by exploiting recurring real-world patterns. This notion of multitasking lends itself to a form of *General Optimization Intelligence* (GOI) - characterized by universal machines capable of performing varied tasks and learning with experience.

That said, the proposed multitasking frameworks are naturally expected to have potential applications in the cloud ecosystems for deep learning [117]. Considering the scope for thousands of massively distributed edge computing nodes that can be applied to diverse use cases, real time access to pretrained deep learning models on the cloud have indeed become practically viable today. Recent advances and trends, such as private sensor networks and secure cloud computing, are making it possible to pervasively and efficiently reconfigure gigantic (pretrained) deep models on the cloud; thus, expediting the extraction and deployment of diverse lightweight models on the edge, with guaranteed privacy via private on-device computations.

This alleviates the otherwise impractical need to constantly create many different deep models and retraining them from scratch, particularly when dealing with multiple user-specific requests at once. Needless to say the otherwise enormous amounts of energy consumed and carbon footprints left behind during continuous model (re)creation, retraining and inferencing [120] - plaguing the long term vision of sustainable environment/AI. Accordingly, the proposed adaptive multitasking frameworks are deemed to be a good fit in such a cloud-based scenario where many diverse compact and resource efficient deep learning models (reconfigured from a huge pretrained deep model hosted on the cloud) are needed to be accessed simultaneously at disparate locations and rapidly deployed on the fly to the edge [119]. For a simple proof-of-concept, the interested reader is referred to Section 6.4.

Along similar lines, it is also noted that there is indeed a scarcity of research efforts in the area of evolutionary deep compression [121] that attempt to explore the potential utility of knowledge transfer across several related problems and improve the overall efficiency. In particular, the use of evolutionary computation for deep learning gets challenging when dealing with big datasets. In such a scenario, knowledge transfers from several cheaper helper (evaluated on just a subset of the big dataset) tasks can help quickly optimize the otherwise expensive primary task (evaluated on the big dataset) of interest. The practicality of utilizing smaller datasets to quickly optimize for big datasets, particularly in resource constrained environments, is indeed invaluable. Therefore, the proposed multitasking frameworks are also expected to yield substantial cost improvements in the general theme of [evolutionary] deep learning.

Appendix A

Supplementary Material

A.1 Proof of Lemma 4.2.1

Proof. Let an offspring solution x_a be assigned a skill factor k i.e, $\tau_a = k$. Given the scope of *parent-centric* inter-task crossover, x_a may have been drawn from either of the densities $p^1(x, t), p^2(x, t), \dots, \dots, \text{or}, p^k(x, t), \dots, \text{or}, p^K(x, t)$ corresponding to K distinct tasks in the multitasking environment. By Bayes' theorem, the probability that x_a is drawn from $p^k(x, t)$, denoted as $x_a \sim p^k(x, t)$ is given by the following:

$$P(x_a \sim p^k(x, t) | \tau_a = k) = \frac{P(\tau_a = k | x_a \sim p^k(x, t)) \cdot P(x_a \sim p^k(x, t))}{P(\tau_a = k)}. \quad (\text{A.1})$$

Considering the uniform allocation of resources to each of the K optimization tasks in the MFEA, we have,

$$P(x_a \sim p^k(x, t)) = P(\tau_a = k) = \frac{1}{K}. \quad (\text{A.2})$$

Hence, by further simplification of Eq. A.1, we have,

$$P(x_a \sim p^k(x, t) | \tau_a = k) = P(\tau_a = k | x_a \sim p^k(x, t)). \quad (\text{A.3})$$

The condition that x_a originates from $p^k(x, t)$ implies that at least one of its parents must also belong to $p^k(x, t)$. In other words, at least one of its parent must have a skill factor k . According to Algorithm 1, the probabilistic event $P(\tau_a = k | x_a \sim p^k(x, t))$ occurs under three possible scenarios given as follows.

Scenario-1: *intra-task crossover* occurs where the second randomly chosen parent of x_a also has skill factor k . Accordingly, the probability of the first scenario is given by,

$$P(\text{Scenario} - 1) = \frac{1}{K}. \quad (\text{A.4})$$

Scenario-2: *inter-task crossover* with probability $rm p$, after which x_a is assigned skill factor k with probability 0.5. Thus, the probability of the second case is expressed as,

$$P(\text{Scenario} - 2) = \frac{0.5 \cdot (K - 1) \cdot rm p}{K}. \quad (\text{A.5})$$

Scenario-3: x_a is created by a small (local) variation of the parent with skill factor k . The probability of the third case is given by,

$$P(\text{Scenario} - 3) = \frac{(K - 1) \cdot (1 - rm p)}{K}. \quad (\text{A.6})$$

Consequently we have,

$$P(\tau_a = k | x_a \sim p^k(x, t)) = \sum_{i=1}^3 P(\text{Scenario} - i). \quad (\text{A.7})$$

Thus,

$$P(x_a \sim p^k(x, t) | \tau_a = k) = 1 - \frac{0.5 \cdot (K - 1) \cdot rm p}{K}. \quad (\text{A.8})$$

Similarly, it can be shown that,

$$P(x_a \sim p^j(x, t) \forall j \neq k | \tau_a = k) = \sum_{j \neq k} \frac{0.5 \cdot rm p}{K}. \quad (\text{A.9})$$

Combining Eq. (A.8) and Eq (A.9), the overall mixture distribution $p_c^k(x, t)$ can be expressed as,

$$p_c^k(x, t) = [1 - \frac{0.5 \cdot (K - 1) \cdot rm p}{K}] \cdot p^k(x, t) + \sum_{j \neq k} \frac{0.5 \cdot rm p}{K} \cdot p^j(x, t). \quad (\text{A.10})$$

□

A.2 Visualization of Synthetic Benchmarks proposed in Chapter 4

A.2.1 Visualization of Complete Intersection (CI) Case in 1-D

For elaboration, the notion of complete intersection of global optima is illustrated in Figure A.1. The figure represents a projection in the unified space, and shows that the global optima of the two tasks T_1 and T_2 are located at the same point in the unified search space.

FIGURE A.1: In the complete intersection case, the global optimum of all tasks maps to the same point in the unified search space.

A.2.2 Visualization of No Intersection (NI) Case in 1-D

The notion of no intersection of global optima is illustrated in Figure A.2. From the figure, we see that the global optima of the two tasks are no longer at the same point, but are separated from each other, in the unified search space.

FIGURE A.2: In the no intersection case, the global optima of the two tasks T_1 and T_2 are separated.

List of Author's Publications¹

Journal Articles

- **Bali, K.K.**, Ong, Y.S., Gupta, A. and Tan, P.S., 2019. Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II. *IEEE Transactions on Evolutionary Computation*, 24(1), pp.69-83.
- **Bali, K.K.**, Gupta, A., Ong, Y.S. and Tan, P.S., 2020. Cognizant Multitasking in Multiobjective Multifactorial Evolution: MO-MFEA-II. *IEEE Transactions on Cybernetics*.

Conference Proceedings

- **Bali, K.K.**, Gupta, A., Feng, L., Ong, Y.S. and Siew, T.P., 2017, June. Linearized domain adaptation in evolutionary multitasking. In *IEEE Congress on Evolutionary Computation (CEC)* (pp. 1295-1302).

Bibliography

- [1] Lawrence A Birnbaum. *Machine Learning Proceedings 1993: Proceedings of the Tenth International Conference on Machine Learning, University of Massachusetts, Amherst, June 27-29, 1993*. Morgan Kaufmann, 2014. [xix](#), [9](#)
- [2] Bingshui Da, Yew-Soon Ong, Liang Feng, A.K. Qin, Abhishek Gupta, Zexuan Zhu, Chuan-Kang Ting, Ke Tang, and Xin Yao. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results. Technical report, Nanyang Technological University, Singapore, 2016. <http://www.cil.ntu.edu.sg/mfo/download.html>. [xxiii](#), [3](#), [39](#), [48](#)
- [3] James P Byrnes. *Cognitive development and learning in instructional contexts*. Allyn & Bacon, 2001. [1](#)
- [4] Richard Han. *Math for Machine Learning Open Doors to Data Science and Artificial Intelligence*. <https://www.onlinemathtraining.com>, 2015. [1](#)
- [5] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646, 1996. [2](#)
- [6] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [7] Muhammad Iqbal, Will N Browne, and Mengjie Zhang. Extracting and using building blocks of knowledge in learning classifier systems. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 863–870. ACM, 2012.
- [8] Muhammad Iqbal, Will N Browne, and Mengjie Zhang. Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. *IEEE Transactions on Evolutionary Computation*, 18(4):465–480, 2014.

- [9] Muhammad Iqbal, Bing Xue, Harith Al-Sahaf, and Mengjie Zhang. Cross-domain reuse of extracted knowledge in genetic programming for image classification. *IEEE Transactions on Evolutionary Computation*, 2017. [2](#), [13](#)
- [10] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998. [2](#), [3](#), [8](#)
- [11] Yew-Soon Ong and Abhishek Gupta. Evolutionary multitasking: a computer science view of cognitive multitasking. *Cognitive Computation*, 8(2):125–142, 2016. [2](#), [7](#), [10](#), [18](#), [96](#)
- [12] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Insights on transfer optimization: Because experience is the best teacher. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):51–64, 2017. [2](#), [12](#), [13](#), [14](#), [19](#), [55](#)
- [13] Serkan Cabi, Sergio Gómez Colmenarejo, Matthew W Hoffman, Misha Denil, Ziyu Wang, and Nando de Freitas. The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. *arXiv preprint arXiv:1707.03300*, 2017. [3](#), [75](#), [78](#)
- [14] Pádraig Cunningham and Barry Smyth. Case-based reasoning in scheduling: reusing solution components. *International Journal of Production Research*, 35(11):2947–2962, 1997. [3](#)
- [15] Marjan Kaedi and Nasser Ghasem-Aghaee. Biasing bayesian optimization algorithm using case based reasoning. *Knowledge-Based Systems*, 24(8):1245–1253, 2011.
- [16] Martin Pelikan, Mark W Hauschild, and Pier Luca Lanzi. Transfer learning, soft distance-based bias, and the hierarchical boa. In *International Conference on Parallel Problem Solving from Nature*, pages 173–183. Springer, 2012.
- [17] Liang Feng, Yew-Soon Ong, Ah-Hwee Tan, and Ivor W Tsang. Memes as building blocks: a case study on evolutionary optimization+ transfer learning for routing problems. *Memetic Computing*, 7(3):159–180, 2015.
- [18] Liang Feng, Yew-Soon Ong, Siwei Jiang, and Abhishek Gupta. Autoencoding evolutionary search with learning across heterogeneous problems. *IEEE Transactions on Evolutionary Computation*, 2017.
- [19] Min Jiang, Zhongqiang Huang, Liming Qiu, Wenzhen Huang, and Gary G Yen. Transfer learning based dynamic multiobjective optimization algorithms. *arXiv preprint arXiv:1612.06093*, 2016. [3](#)

- [20] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013. [3](#), [8](#), [10](#), [11](#), [83](#)
- [21] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Multifactorial evolution: towards evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3):343–357, 2016. [3](#), [4](#), [8](#), [10](#), [11](#), [13](#), [14](#), [26](#), [38](#), [76](#), [92](#), [96](#)
- [22] Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. Multiobjective multifactorial optimization in evolutionary multitasking. *IEEE Transactions on cybernetics*, 2017. [3](#), [4](#), [11](#), [13](#), [14](#), [18](#), [54](#), [55](#), [65](#), [75](#)
- [23] Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008. [3](#), [10](#), [11](#), [35](#)
- [24] Martin Pelikan, Mark W Hauschild, and Fernando G Lobo. Estimation of distribution algorithms. In *Springer Handbook of Computational Intelligence*, pages 899–928. Springer, 2015. [4](#), [26](#)
- [25] Martin Pelikan, David E Goldberg, and Fernando Lobo. A survey of optimization by building and using probabilistic models. In *American Control Conference, 2000. Proceedings of the 2000*, volume 5, pages 3289–3293. IEEE, 2000. [4](#)
- [26] Linda S Gottfredson. Mainstream science on intelligence: An editorial with 52 signatories, history, and bibliography, 1997. [7](#)
- [27] Christopher D Wickens. Situation awareness: Review of mica endsley’s 1995 articles on situation awareness theory and measurement. *Human factors*, 50(3):397–403, 2008. [7](#)
- [28] Kirsi Tirri and Petri Nokelainen. *Measuring multiple intelligences and moral sensitivities in education*, volume 5. Springer Science & Business Media, 2012. [7](#)
- [29] Liang Feng, Lei Zhou, Jinghui Zhong, Abhishek Gupta, Yew-Soon Ong, Kay-Chen Tan, and AK Qin. Evolutionary multitasking via explicit autoencoding. *IEEE transactions on cybernetics*, (99):1–14, 2018. [8](#), [14](#), [64](#), [96](#)
- [30] Abhishek Gupta and Yew-Soon Ong. Back to the roots: Multi-x evolutionary computation. *Cognitive Computation*, pages 1–17, 2019. [8](#), [13](#), [55](#)
- [31] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. [9](#)

- [32] Yu-Chi Ho and David L Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570, 2002. [9](#)
- [33] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. [10](#)
- [34] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019. [10](#)
- [35] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1871–1880, 2019.
- [36] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020. [10](#)
- [37] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. [11](#)
- [38] Martin Zaefferer and Thomas Bartz-Beielstein. Efficient global optimization with indefinite kernels. In *International Conference on Parallel Problem Solving from Nature*, pages 69–79. Springer, 2016. [11](#)
- [39] Heinz Mühlenbein, Martina Gorges-Schleuter, and Ottmar Krämer. Evolution algorithms in combinatorial optimization. *Parallel computing*, 7(1):65–85, 1988. [11](#)
- [40] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer, 1994. [11](#)
- [41] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, and Xin Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):378–393, 2014. [11](#)
- [42] Qunjian Chen, Xiaoliang Ma, Yiwen Sun, and Zexuan Zhu. Adaptive memetic algorithm based evolutionary multi-tasking single-objective optimization. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 462–472. Springer, 2017. [11](#)

- [43] Rung-Tzuo Liaw and Chuan-Kang Ting. Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 2266–2273. IEEE, 2017.
- [44] Yu-Wei Wen and Chuan-Kang Ting. Parting ways and reallocating resources in evolutionary multitasking. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 2404–2411. IEEE, 2017. [11](#)
- [45] Chellapilla Patvardhan, Sulabh Bansal, and Anand Srivastav. Quantum-inspired evolutionary algorithm for difficult knapsack problems. *Memetic Computing*, 7(2): 135–155, 2015. [11](#)
- [46] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987. [11](#)
- [47] Mohammad-H Tayarani-N and Adam Prügel-Bennett. On the landscape of combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(3):420–434, 2013. [11](#)
- [48] Yu-Wei Wen and Chuan-Kang Ting. Learning ensemble of decision trees through multifactorial genetic programming. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 5293–5300. IEEE, 2016. [12](#)
- [49] Rohitash Chandra, Abhishek Gupta, Yew-Soon Ong, and Chi-Keong Goh. Evolutionary multi-task learning for modular knowledge representation in neural networks. *Neural Processing Letters*, pages 1–17, 2017. [12](#)
- [50] Eneko Osaba, Aritz D Martinez, and Javier Del Ser. Evolutionary multitask optimization: a methodological overview, challenges and future research directions. *arXiv preprint arXiv:2102.02558*, 2021. [13](#)
- [51] Jinliang Ding, Cuie Yang, Yaochu Jin, and Tianyou Chai. Generalized multitasking for evolutionary optimization of expensive problems. *IEEE Transactions on Evolutionary Computation*, 2017. [13](#), [83](#)
- [52] Lei Zhou, Liang Feng, Jinghui Zhong, Yew-Soon Ong, Zexuan Zhu, and Edwin Sha. Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

- [53] Rung-Tzuo Liaw and Chuan-Kang Ting. Evolutionary manytasking optimization based on symbiosis in biocoenosis. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.
- [54] Fang Shen, Jing Liu, and Kai Wu. Evolutionary multitasking fuzzy cognitive map learning. *Knowledge-Based Systems*, 192:105294, 2020.
- [55] Liang Bao, Yutao Qi, Mengqing Shen, Xiaoxuan Bu, Jusheng Yu, Qian Li, and Ping Chen. An evolutionary multitasking algorithm for cloud computing service composition. In *World Congress on Services*, pages 130–144. Springer, 2018.
- [56] Pham Dinh Thanh, Huynh Thi Thanh Binh, and Tran Ba Trung. An efficient strategy for using multifactorial optimization to solve the clustered shortest path tree problem. *Applied Intelligence*, 50(4):1233–1258, 2020.
- [57] Xiaolong Zheng, A Kai Qin, Maoguo Gong, and Deyun Zhou. Self-regulated evolutionary multitask optimization. *IEEE Transactions on Evolutionary Computation*, 24(1):16–28, 2019. [13](#)
- [58] Shuangshuang Yao, Zhiming Dong, Xianpeng Wang, and Lei Ren. A multiobjective multifactorial optimization algorithm based on decomposition and dynamic resource allocation strategy. *Information Sciences*, 511:18–35, 2020. [14](#)
- [59] Huynh Thi Thanh Binh, Nguyen Quoc Tuan, and Doan Cao Thanh Long. A multi-objective multi-factorial evolutionary algorithm with reference-point-based approach. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2824–2831. IEEE, 2019. [15](#)
- [60] J. Lin, H. Liu, B. Xue, M. Zhang, and F. Gu. Multi-objective multi-tasking optimization based on incremental learning. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2019. ISSN 1941-0026. doi: 10.1109/TEVC.2019.2962747. [15](#)
- [61] Kavitesh Kumar Bali, Abhishek Gupta, Liang Feng, Yew Soon Ong, and Tan Puay Siew. Linearized domain adaptation in evolutionary multitasking. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 1295–1302. IEEE, 2017. [15](#), [96](#)
- [62] Kavitesh Kumar Bali, Yew-Soon Ong, Abhishek Gupta, and Puay Siew Tan. Multifactorial evolutionary algorithm with online transfer parameter estimation: Mfea-ii. *IEEE Transactions on Evolutionary Computation*, 2019. [17](#), [25](#), [56](#), [59](#), [60](#), [75](#), [96](#)
- [63] Abhishek Gupta, Jacek Mańdziuk, and Yew-Soon Ong. Evolutionary multitasking in bi-level optimization. *Complex & Intelligent Systems*, 1(1-4):83–95, 2015. [18](#)

- [64] Padhraic Smyth and David Wolpert. Stacked density estimation. In *Advances in neural information processing systems*, pages 668–674, 1998. [19](#)
- [65] Pedro Larrañaga and Jose A Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2001. [19](#)
- [66] L Devroye L Györfi, Gabor Lugosi, and L Devroye. A probabilistic theory of pattern recognition, 1996. [20](#), [34](#)
- [67] Hans-Georg Beyer. *The theory of evolution strategies*. Springer Science & Business Media, 2013. [20](#)
- [68] Qingfu Zhang and Heinz Muhlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on evolutionary computation*, 8(2): 127–136, 2004. [21](#), [59](#)
- [69] YS Chow and H Yeicher. Probability theory, 3rd ed., 1997. [22](#)
- [70] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary computation*, 10(4):371–395, 2002. [27](#)
- [71] Ram Bhushan Agrawal, K Deb, and RB Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995. [27](#)
- [72] Kalyanmoy Deb and Mayank Goyal. A combined genetic adaptive search (geneas) for engineering design. *Computer Science and informatics*, 26:30–45, 1996. [27](#)
- [73] Robert Hinterding. Gaussian mutation and self-adaption for numeric genetic algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, page 384. IEEE, 1995. [27](#)
- [74] Kalyanmoy Deb, Dhiraj Joshi, and Ashish Anand. Real-coded evolutionary algorithms with parent-centric recombination. In *Evolutionary Computation, 2002. CEC’02. Proceedings of the 2002 Congress on*, volume 1, pages 61–66. IEEE, 2002. [27](#)
- [75] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. [34](#)
- [76] David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 863–870. Omnipress, 2010. [35](#)

- [77] Kumara Sastry, David E Goldberg, and Martin Pelikan. Limits of scalability of multiobjective estimation of distribution algorithms. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2217–2224. IEEE, 2005. [39](#)
- [78] Kavitesh Kumar Bali, Abhishek Gupta, Yew-Soon Ong, and Puay Siew Tan. Cognizant multitasking in multiobjective multifactorial evolution: Mo-mfea-ii. *IEEE Transactions on Cybernetics*, 2020. [53](#), [75](#), [96](#)
- [79] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007. [54](#)
- [80] Yuan Yuan, Yew-Soon Ong, Liang Feng, A Kai Qin, Abhishek Gupta, Bingshui Da, Qingfu Zhang, Kay Chen Tan, Yaochu Jin, and Hisao Ishibuchi. Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics and baseline results. *arXiv preprint arXiv:1706.02766*, 2017. [55](#), [65](#), [68](#), [96](#)
- [81] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. [55](#), [62](#), [92](#)
- [82] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013. [55](#)
- [83] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2013. [55](#)
- [84] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001. [55](#)
- [85] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007. [55](#)
- [86] Ke Li, Sam Kwong, Qingfu Zhang, and Kalyanmoy Deb. Interrelationship-based selection for decomposition multiobjective optimization. *IEEE transactions on cybernetics*, 45(10):2076–2088, 2014.

- [87] Shouyong Jiang and Shengxiang Yang. An improved multiobjective optimization evolutionary algorithm based on decomposition for complex pareto fronts. *IEEE transactions on cybernetics*, 46(2):421–437, 2015. [55](#)
- [88] James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160, 1994. [55](#)
- [89] Abhishek Gupta and Yew-Soon Ong. *Memetic Computation: The Mainspring of Knowledge Transfer in a Data-Driven Optimization Era*, volume 21. Springer, 2018. [61](#)
- [90] Edwin A Williams and William A Crossley. Empirically-derived population size and mutation rate guidelines for a genetic algorithm with uniform crossover. In *Soft computing in engineering design and manufacturing*, pages 163–172. Springer, 1998. [65](#), [87](#)
- [91] Marco Locatelli. A note on the griewank test function. *Journal of global optimization*, 25(2):169–174, 2003. [66](#)
- [92] David Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer Science & Business Media, 2012. [67](#)
- [93] LA Rastrigin. Extremal control systems. *Theoretical foundations of engineering cybernetics series*, 3, 1974. [67](#)
- [94] Heinz Mühlenbein, M Schomisch, and Joachim Born. The parallel genetic algorithm as function optimizer. *Parallel computing*, 17(6-7):619–632, 1991. [67](#)
- [95] Siwei Jiang, Yew-Soon Ong, Jie Zhang, and Liang Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE transactions on cybernetics*, 44(12):2391–2404, 2014. [67](#)
- [96] Jing Tang, Yingke Chen, Zixuan Deng, Yanping Xiang, and Colin Paul Joy. A group-based approach to improve multifactorial evolutionary algorithm. In *IJCAI*, pages 3870–3876, 2018. [71](#)
- [97] Rung-Tzuo Liaw and Chuan-Kang Ting. Evolutionary manytasking optimization based on symbiosis in biocoenosis. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. [71](#), [104](#)
- [98] Yuan Yuan, Yew-Soon Ong, Abhishek Gupta, Puay Siew Tan, and Hua Xu. Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP. In *Region 10 Conference (TENCON), 2016 IEEE*, pages 3157–3164. IEEE, 2016. [75](#)

-
- [99] Rohitash Chandra, Abhishek Gupta, Yew-Soon Ong, and Chi-Keong Goh. Evolutionary multi-task learning for modular training of feedforward neural networks. In *International Conference on Neural Information Processing*, pages 37–46. Springer, 2016. 75
- [100] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9(May):937–965, 2008. 76, 78
- [101] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014. 76, 77
- [102] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 845–852. ACM, 2011. 76, 80, 81
- [103] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. Exponential natural evolution strategies. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 393–400. ACM, 2010. 76, 80, 81
- [104] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017. 76
- [105] Tom Schaul. Natural evolution strategies, November 2017. URL <http://people.idsia.ch/~tom/nas.html>. 77
- [106] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical report, Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011. 77
- [107] Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984. 78
- [108] Faustino J Gomez and Risto Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *IJCAI*, volume 99, pages 1356–1361, 1999.
- [109] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002. 79

- [110] Christian Igel. Neuroevolution for reinforcement learning using evolution strategies. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 4, pages 2588–2595. IEEE, 2003. [78](#)
- [111] Alexis P Wieland. Evolving neural network controllers for unstable systems. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 667–673. IEEE, 1991. [79](#)
- [112] Myungsoo Jun and Raffaello D’Andrea. Path planning for unmanned aerial vehicles in uncertain and adversarial environments. In *Cooperative control: models, applications and algorithms*, pages 95–110. Springer, 2003. [83](#)
- [113] Momena Monwar, Omid Semiari, and Walid Saad. Optimized path planning for inspection by unmanned aerial vehicles swarm with energy constraints. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018. [83](#)
- [114] Jaime Rubio-Hervas, Abhishek Gupta, and Yew-Soon Ong. Data-driven risk assessment and multicriteria optimization of uav operations. *Aerospace Science and Technology*, 77:510–523, 2018. [83](#), [84](#), [86](#)
- [115] Christian H Reinsch. Smoothing by spline functions. *Numerische mathematik*, 10(3):177–183, 1967. [85](#)
- [116] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 862–876. Springer, 2007. [87](#)
- [117] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. [90](#), [106](#)
- [118] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—the business perspective. *Decision support systems*, 51(1):176–189, 2011. [90](#), [106](#)
- [119] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016. [90](#), [107](#)
- [120] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019. [107](#)

- [121] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. [91](#), [107](#)
- [122] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. [91](#), [106](#)
- [123] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019. [95](#)
- [124] Dario Floreano, Peter Dürri, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1(1):47–62, 2008. [95](#)
- [125] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017. [95](#)
- [126] Aritz D Martinez, Javier Del Ser, Esther Villar-Rodriguez, Eneko Osaba, Javier Poyatos, Siham Tabik, Daniel Molina, and Francisco Herrera. Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. *arXiv preprint arXiv:2008.03620*, 2020. [95](#)
- [127] Yongliang Chen, Jinghui Zhong, Liang Feng, and Jun Zhang. An adaptive archive-based evolutionary framework for many-task optimization. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019. [104](#)
- [128] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90, 2017. [106](#)