Multi-scale search for black-box optimization : theory & algorithms

Abdullah Shamil Hashim Al-Dujaili

2017

http://hdl.handle.net/10356/70300

https://doi.org/10.32657/10356/70300

Multi-Scale Search for Black-Box Optimization: Theory & Algorithms

ABDULLAH SHAMIL HASHIM AL-DUJAILI
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
2017

Multi-Scale Search for Black-Box Optimization: Theory & Algorithms

ABDULLAH SHAMIL HASHIM AL-DUJAILI

School of Computer Science and Engineering

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

2017

Acknowledgements

First and foremost, I would like to express my praises to God for blessing me in all my endeavors. It is my faith in Him that kept me firm throughout my PhD.

My deepest appreciation and gratitude is extended to my supervisor, Assoc. Prof. Suresh Sundaram for his endless support, diligence, and encouragement. He patiently guided and believed in me through thick and thin. I have learnt a lot from his ways of approaching a hurdle and breaking it down. I am lucky to have such a great mentor and am very proud to be his student.

I want to greatly thank Assoc. Prof. Suhaib Fahmy for admitting me as a PhD student at Nanyang Technological University. He had selflessly sought to provide me with the best possible platform and exposure in the early course of my PhD.

Apart from that, I am very thankful to Dr. Narasimhan Sundararajan and Assoc. Prof. Deepu Rajan for their numerous helpful comments and enlightening discussions throughout my research course.

I was indeed blessed to know many great people at Nanyang Technological University (NTU). Special thanks to my colleagues at the Computational Intelligence Laboratory (CIL) and the Centre for High Performance Embedded Systems (CHIPES), who have given me invaluable suggestions and support. It is an honor for me to thank our laboratory executive, Mr. Kesavan Asaithambi for being very dedicated in providing technical support. Thanks extended to Bhavarth Pandya, Chaitanya Prasad, Khyati Mahajan, and Shaleen Gupta for their contribution to the Black-box Multi-objective Optimization Benchmarking (BMOBench) platform.

I am heartily indebted to my family. My father and my hero, Shamil, has been a great role model for my entire life. From you, I have learned perseverance and patience. My mother, Hadeel, constantly encouraged and showed me the joy of learning and kindness. You are my angel of mercy. My siblings, Yasmeen and Ahmed, have given their deepest love to me. I miss all of you a lot and I am thankful for your love and support. Apart from that, my source of happiness, support, and encouragement was my wife, Sana. She was the only person with whom I shared it all. I can not believe how lucky I am to be with you, my princess. I am thankful to NTU for accepting both of us in the Summer of 2012.

Finally, I am thankful to Singapore for providing me this unique opportunity to pursue my PhD. The Red Dot will always be my second home and I love to see it, forever, prosperous.

To Baghdad, my hometown.

To my family.

Contents

Acknowledgements ii

Abstract viii

List of Figures xi

List of Tables xviii

List of Abbreviations xxi

List of Notation xxii

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 5

1.3 Thesis Contributions 6

1.3.1 Theoretical Analysis Contributions 6

1.3.2 Algorithmic Design Contributions 7

1.3.3 Benchmarking Contributions 8

1.4 Thesis Organization 9

2 Literature Review **11**

2.1 Continuous Black-Box Optimization 11

2.2 Historical Overview of Black-Box Mathematical Programming . . . 12

2.3 Partitioning Algorithms for Continuous Black-Box Optimization . . 21

2.3.1 Lipschitzian Optimization (`LO`) 22

2.3.2 DIviding RECTangles (`DIRECT`) 24

2.3.3 Multilevel Coordinate Search (`MCS`) 27

2.3.4 Optimistic Optimization 29

2.4 Discussion 32

2.4.1 Rules of Partitioning 32

2.4.2 Multi-Objective Optimization 33

2.5 Summary 37

3 Single-Objective Multi-Scale Search 39

3.1 Some Preliminaries 40

3.2 Multi-scale Search Optimization (MSO) 41

3.2.1 A Generic Procedure of MSO Algorithms 42

3.2.2 MSO Algorithms in the Literature 46

3.3 Convergence Analysis of MSO algorithms 52

3.3.1 Theoretical Framework for Convergence 52

3.3.2 Analysis of MSO Algorithms 63

3.4 Discussion 71

3.5 Conclusions 73

4 Naive Multi-Scale Optimization 75

4.1 Naive Multi-scale Search Optimization (`NMSO`) 76

4.1.1 The `NMSO` Algorithm 77

4.1.2 Convergence Analysis 80

4.2 Numerical Assessment 83

4.2.1 Experimental Setup 84

4.2.2 Performance Evaluation Procedure 86

4.2.3 Performance Evaluation Presentation 86

4.2.4 Performance Evaluation Discussion 87

4.2.5 Empirical Computational Complexity 96

4.3 Black Box Optimization Competition (BBComp) 97

4.4 Conclusion 99

5 Multi-Objective Multi-Scale Search 100

5.1 Multi-Scale Search for Multi-Objective Optimization 102

5.1.1 Multi-Objective DIviding RECTangles 102

5.1.2 Multi-Objective Simultaneous Optimistic Optimization . . . 104

5.2 A Worked Example 106

5.3 Convergence Analysis 110

5.3.1 Convergence Analysis of `MO-DIRECT` 110

5.3.2 Convergence Analysis of `MO-SOO` 111

5.4 Conclusion 129

6 Benchmarking Multi-Objective Multi-Scale Search 130
6.1 Bi-Objective Black-Box Optimization Benchmarking (BO-BBOB) . 131
6.1.1 Compared Algorithms 131
6.1.2 Test Problems 131
6.1.3 Benchmark Procedure 134
6.1.4 Performance Discussion 135
6.2 Black-box Multi-objective Optimization Benchmarking (BMOBench) 140
6.2.1 Test Problems 140
6.2.2 Evaluation Budget 140
6.2.3 Benchmark Procedure 142
6.2.4 Compared Algorithms 142
6.2.5 Results & Discussion 143
6.3 Conclusion 143

7 Conclusions and Future Works 147
7.1 Summary of Contributions 147
7.2 Future Directions 151

Publications List 153

References 155

Abstract

This thesis focuses on a special class of Mathematical Programming (MP) algorithms for continuous black-box optimization. Black-box optimization has been a recurrent subject of interest for decades, as many real-world applications can be modeled as black-box optimization problems. In particular, this research work studies algorithms that partition the problem's decision space over multiple scales in search for the optimal solution and investigates three central topics.

- Plenty of such algorithms have been proposed and *analyzed independently.* Furthermore, the theoretical analysis has been dominantly concerned with the asymptotic (limit) behavior of the algorithms and their convergence to optimal points, whereas *finite-time* behavior is more crucial in practice.
- Due to their systematic sampling of the decision space, the aforementioned algorithms are inherently exploratory. This prevents from using them effectively on *computationally expensive* optimization problems.
- The bulk of these algorithms has been tailored towards single-objective optimization, whilst in practice, most real-world problems involve the optimization of *multiple* objectives.

The present thesis refers to these algorithms as *Multi-scale Search Optimization* (MSO) algorithms and addresses the above issues as follows. First, a theoretical methodology is presented to *consolidate* the analysis of MSO algorithms and study their *finite-time* convergence based on three basic assumptions: i). local Hölder continuity of the objective function f; ii). partitions boundedness; and iii). partitions sphericity. Moreover, the worst-case finite-time performance and convergence rate of several leading MSO algorithms, viz. Lipschitzian optimization methods, Multilevel Coordinate Search (`MCS`), DIviding RECTangles (`DIRECT`), and optimistic optimization methods have been shown.

Second, in order to deal with expensive optimization, an algorithm—referred to as the Naive Multi-scale Search Optimization (`NMSO`) algorithm—within the MSO framework is developed. When compared with other MSO algorithms, `NMSO`'s exploitative search component is more dominant than its exploratory one. Besides preserving its asymptotically-consistent property, `NMSO` enjoys a finite-time convergence, which is generally difficult to establish for exploitation-biased algorithms. Along with its theoretical study, a numerical assessment of `NMSO`—comparing it with established multi-scale search algorithms such as `MCS` and `DIRECT`—has been conducted. Based on the results, `NMSO` demonstrates a better performance than the compared algorithms under limited (expensive) evaluation budget, particularly in problems with separability, multi-modality, and low dimensionality. This is in line with the Black-box Optimization Competition (BBComp) held at GECCO'15, where `NMSO` finished third out of twenty-eight competitors in solving 1000 black-box problems.

Third, in order to expand the frontiers of multi-scale search algorithms, two MSO algorithmic instances are extended for multi-objective optimization: i). Multi-Objective DIviding RECTangles (`MO-DIRECT`); ii) Multi-Objective Simultaneous Optimistic Optimization (`MO-SOO`). Both of these algorithms are asymptotically convergent towards the Pareto front. Furthermore, based on the presented theoretical methodology to analyze MSO algorithms, an upper bound on the Pareto-compliant unary additive epsilon indicator is established as a function of the number of iterations. The bound is characterized by the objectives smoothness as well as the structure of the Pareto front with respect to its extrema. First time in the literature, a deterministic upper bound on a Pareto-compliant indicator has been presented for a solver of continuous multi-objective problems.

Finally, to validate the efficacy of the proposed multi-objective MSO algorithms, a Black-box Multi-objective Optimization Benchmarking (BMOBench) platform is built around 100 multi-objective optimization problems from the literature, where the performance assessment is reported in terms of Pareto- and Non-Pareto-compliant data profiles. BMOBench can be employed as a comprehensive platform for benchmarking any multi-objective optimization algorithm. Besides the built platform, 300 bi-objective were used in comparing the proposed multi-objective MSO algorithms with the state-of-the-art algorithms. Based on the results, `MO-SOO` shows a performance on a par with the top performing algorithm, viz. `SMS-EMOA` and `DMS`, especially with limited number of function evaluations. Future work includes: i). breaking away from MSO's systematic sampling towards more adaptive/learning scheme; ii). handling large-scale problems and general constraints; and iii). employing indicator-based techniques for multi-objective MSO.

List of Figures

1.1 A binary search tree built with 11 function evaluations over the decision space $\mathcal{X}$. The tree constructed, in a hierarchical fashion, four $\{0, 1, 2, 3\}$ partitions of the search space of different scale/granularity. 4

2.1 3-dimensional maps for 2-dimensional functions exhibiting typical challenges in numerical black-box optimization. Figures (b), (c), and (d) are adapted from Chen et al. [1]. 14

2.2 Working principle of `Lipschitzian Optimization` in one-dimensional search space. 23

2.3 Identification of *potentially-optimal* hyperrectangles by `DIRECT`. . . 25

3.1 The process of computing exploration and exploitation scores for the set of leaf nodes $\mathcal{L} \subseteq \mathcal{T}$ (a) of an MSO algorithm $\mathcal{A}$ can be regarded as projecting them onto a 2-D Euclidean space (b) $\mathbb{R}^2$ via the mapping function $s : \mathcal{L} \to \mathbb{R}^2$. Y represents $\mathcal{L}$'s image under s and $P(Y) \subseteq Y$ is the potentially optimal set. Here $P(Y)$ lies on the level set of $b(x) = l_x + \lambda \cdot g_x$ that corresponds to the value 4 (the greatest among Y's). $\mathcal{P}$ and $\mathcal{Q}$ are chosen from the set of leaf nodes whose image under s is $P(Y)$. 45

3.2 Selecting expandable leaf node(s) $\mathcal{Q}$ (represented by black dots) for an iteration in `LO` (a), for a batch of iterations in `DIRECT` (b) , a batch of iterations in `MCS` (c), an iteration in `DOO` (d), and a batch of iterations in `SOO` (e). The set Y, whose elements are represented by black and gray dots, is the set of projected evaluated leaves into the exploration-exploitation space. 53

3.3 $\mathcal{X}_\epsilon$ in a 2-dimensional space is an $\acute{\ell}$-circle (Here, $\acute{\ell}$ and ℓ are the l_2 norms, with α and β set to 1) centered at $\mathbf{x}^*$ with a radius of ϵ. . . 56

3.4 The empirical convergence rate and its theoretical bound with respect to the number of function evaluations `#f-evals` for the algorithms `DIRECT` (a), `SOO` (b), and `MCS` (c) in minimizing $||\mathbf{x} - \mathbf{x}^*||_\infty^\alpha$, with $n = 1$. 73

4.1 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 2-D. 88
4.2 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 3-D. 89
4.3 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 5-D. 90
4.4 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 10-D. 91
4.5 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 20-D. 92
4.6 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 40-D. 93
4.7 Timing Performance of `NMSO` compared with the state-of-the-art MSO optimization algorithms. The y-axis represents the time (in millisecond) per a function evaluation (FEval). All the algorithms were run on a PC with: 64-bit Ubuntu, i7-4702MQ @ 2.20GHz CPU, 8GB of memory. 97

5.1 An illustration of MO-DIRECT (Algorithm 8). The technique identifies and expands potentially-optimal hyperrectangles to look for Pareto optimal solutions by partitioning their spaces along the decision space dimensions according to Algorithm 2, using Eq. (5.3) instead of Eq. (2.6). In each iteration t, MO-DIRECT selects a set of hyperrectangles whose sizes are greater than a certain value σ_t and expands those that are non-dominated in the $(m+1)$-dimensional space of the m-function space appended by the hyperrectangle size range where hyperrectangles of minimum m-function values and bigger sizes are preferred. Subsequently, one or more hyperrectangles can be expanded in one iteration; at the second iteration, for instance, only one hyperrectangle is partitioned (whose center is point 3) into smaller hyperrectangles (whose centers are the points 3, 6, and 7) as it is the only non-dominated point in $(m+1)$-dimensional space. With regard to its partitioning, it has only one dimension (x_1) of maximum side length. Hence, division takes along that dimension, generating three new hyperrectangles. On the other hand, at the third iteration two hyperrectangles are partitioned (resp., centers are the points 3 and 5) into smaller hyperrectangles (resp., centers are the points 3, 8, 9, 10, and 11; and 5, 12, and 13). Here, the division of hyperrectangle 3 takes place along the two dimensions as both sides have equal lengths; the procedure first divides along x_1 as the points 8 and 10 are farther from point 3 in the function space than the points 9 and 11. After 250 function evaluations, MO-DIRECT's approximation set A closely coincides on a sampled set Y_* of the Pareto front of Problem (5.4). 107

5.2 An illustration of MO-SOO. The algorithm expands its leaf nodes to look for Pareto optimal solutions by partitioning their cells along the decision space coordinates one at a time in a round-robin manner, with a partition factor of $K = 3$. Sweeping its tree from the root node depth till the maximal depth specified by $h_{max}(t)$. MO-SOO expands a set of leaf nodes per depth if they are non-dominated with respect to other leaf nodes in the same level and with respect to those expanded at lower depths in the current sweep. Subsequently, none or more nodes can be expanded in one iteration; at the third iteration, for instance, only one node is expanded (whose representative state is point 4) into its children (whose representative states are the points 4, 6, and 7). On the other hand, at the fourth iteration three nodes are expanded (resp., representative states are the points 4, 6, and 7) into their children nodes (resp., representative states are the points 4, 6, 7, 8, 9, 10, 11, 12, and 13). After 20 iterations, MO-SOO's approximation set A closely coincides on a sampled set of the Pareto front $\mathcal{Y}^*$ of Problem (5.4). 108

5.3 Hierarchical partitioning of the decision space $\mathcal{X}$ with a partition factor of $K = 3$ at iteration t represented by a K-ary tree. Consider a multi-objective problem where $m = 2$ and the global optimizers of the first and second objective functions ($\mathbf{x}_1^*$ and $\mathbf{x}_2^*$, respectively) are shown. Thus, the nodes $\{(3,4),(2,1),(1,0),(0,0)\}$ and $\{(2,6),(1,2),(0,0)\}$ are 1- and 2-optimal nodes, respectively. Furthermore, $h_{1,t}^* = 3$ and $h_{2,t}^* = 2$. 114

5.4 The feasible decision space and the corresponding jth objective space ($\mathcal{Y}_j \subseteq \mathbb{R}$). The global optimizer $\mathbf{x}_j^*$ and any solution $\mathbf{x}$ whose image under the jth objective lies within $\{f_j(\mathbf{x}) \leq f_j(\mathbf{x}_j^*) + \delta_j(h)\}$ are denoted by $\mathcal{X}_j^{\delta_j(h)}$. 115

5.5 The objective space $\mathcal{Y}$ for a multi-objective optimization problem ($m = 2$). The solid curve marks the Pareto front $\mathcal{Y}^*$. At depth h, assuming the $\{j\}_{j=1,2}$-optimal nodes are not expanded yet, one can use the $\texttt{ND}_{min}(\cdot)$ operator, which causes nodes whose representative states lie in the decision space portion $\cup_{j=1,2}\mathcal{X}_j^{\delta_j(h)}$ to be expanded before others. However, this may hold up discovering other parts of the Pareto front (the circled region). This is not the case with the $\texttt{ND}(\cdot)$ operator, where another region $\mathcal{X}_{ND}^h$—whose image in the objective space is bounded by the Pareto front and the points $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3 = \mathbf{y}^{nadir}(\mathbf{f}(\cup_{j=1,2}\mathcal{X}_j^{\delta_j(h)}))$—is as well considered. Let the considered depth at iteration t be h and the depth of the deepest $\{j\}_{j=1,2}$-optimal nodes be $h-1$. Then, prior to expanding the $\{j\}_{j=1,2}$-optimal nodes at depth h, the set $\mathcal{Q}_t$ (of Algorithm 9) comprises of at most 3 types of nodes whose representative states lie in $\mathcal{X}_1^{\delta_1(h)}$, $\mathcal{X}_2^{\delta_2(h)}$, and $\mathcal{X}_{ND}^h$, respectively. Furthermore, one can note from Eq. (5.5) as well as Assumptions A5.1 and A5.2 that the point $\mathbf{y}^1$ is greater than or equal $\mathbf{y}^4$, and hence $\mathbf{f}(\mathbf{x}_1^*)$, along f_2 by at most $C_1\delta_1(h)^{-d_{s_1}} \cdot 2\delta_2(h)$, that is to say $y_2^1 - f_2(\mathbf{x}_1^*) \leq C_1\delta_1(h)^{-d_{s_1}} \cdot 2\delta_2(h)$; similar argument can be made between the points $\mathbf{y}^2$ and $\mathbf{y}^5$ along f_1. This observation is the main ingredient in the proof of Theorem 11. 116

5.6 Illustration of the vectorial loss $\mathbf{r}(t)$ of Eq. (2.15) and its relation to the unary additive epsilon indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$ (Definition 7) for a multi-objective problem ($m = 2$) whose objective space $\mathcal{Y}$ is shown in two scenarios: (i) non-conflicting objectives and (ii) conflicting objectives. The faded square (resp., curve) in the first (resp., second) scenario represents the least-translated Pareto front so as every translated element is weakly dominated by at least one element in the approximation set $\mathcal{Y}^t_*$, i.e., $\{(y_1 + I^1_{\epsilon+}(\mathcal{Y}^t_*), \ldots, y_m + I^1_{\epsilon+}(\mathcal{Y}^t_*))\}_{\mathbf{y}\in\mathcal{Y}^*}$. Mathematically, $I^1_{\epsilon+}(\mathcal{Y}^t_*) \geq \max_{1\leq j\leq m} r_j(t)$ where equality sufficiently holds when $|\mathcal{Y}^t_*| = 1$ (see Lemma 3). 119

5.7 The objective space $\mathcal{Y}$ for a multi-objective optimization problem ($m = 2$) with conflict dimension $\Psi \geq 0$ (Definition 22). The solid curve marks the Pareto front $\mathcal{Y}^*$. The faded curve represents the Ψ-translated Pareto front, i.e., $\{(y_1 + \Psi, \ldots, y_m + \Psi)\}_{\mathbf{y} \in \mathcal{Y}^*}$. Every element of the Ψ-translated Pareto front is *weakly* dominated by at least one element in the set $\{\mathbf{f}(\mathbf{x}_j^*)\}_{1 \leq j \leq m} \subseteq \mathcal{Y}^*$. 120

5.8 Bi-objective problem ($m = 2, n = 1$) over $\mathcal{X} = [0, 1]$ with $f_1(\mathbf{x}) = ||\mathbf{x} - 0.25||_\infty^{\alpha_1}$, $f_2 = ||\mathbf{x} - 0.75||_\infty^{\alpha_2}$, $\ell_1(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_\infty^{\beta_1}$, $\ell_2(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_\infty^{\beta_2}$ where $\beta_1 \leq \alpha_1$, $\beta_2 \leq \alpha_2$. The region $\mathcal{X}_1^\epsilon$ (resp., $\mathcal{X}_2^\epsilon$) is the interval centered around x_1^* (resp., x_*^2) of length $2 \cdot \epsilon^{1/\alpha_1}$ (resp., $2 \cdot \epsilon^{1/\alpha_2}$). They can be packed with $\epsilon^{1/\alpha_1 - 1/\beta_1}$ (resp., $\epsilon^{1/\alpha_2 - 1/\beta_2}$) intervals of length $2 \cdot \epsilon^{1/\beta_1}$ (resp., $2 \cdot \epsilon^{1/\beta_2}$). 124

5.9 Empirical validation of `MO-SOO`'s finite-time analysis for eight instances $\{a, \ldots, h\}$ of the bi-objective problem of Figure 5.8. Each plot shows the problem measures, namely the loss measures $r_1(t)$, $r_2(t)$ and the indicator $I_{\epsilon+}^1(\mathcal{Y}_*^t)$, as well as their upper bounds (denoted by $\bar{r}_1$, $\bar{r}_2$, and $\bar{I}_{\epsilon+}^1$, respectively) as a function of the number of iterations t with a computational budget of $v = 10^4$ function evaluations. The upper bounds are obtained via symbolic computation of the (5.9) and (5.11) equations using MATLAB's Symbolic Math Toolbox. The header of each instance's plot reports the decision space dimension n and the conflict dimension Ψ. The j-optimal solutions (x_1^*, x_2^*) are fixed as follows: $(\mathbf{0}, \mathbf{1})$ for (a) and (e), $(\mathbf{0.21}, \mathbf{0.81})$ for (b) and (f), $(\mathbf{0.47}, \mathbf{0.61})$ for (c) and (g), and $(\mathbf{0.57}, \mathbf{0.57})$ for (d) and (h). 128

6.1 I_H^--based data profiles aggregating over all 300 problems for a decision space dimensionality $n \in \{2, 3, 5, 10, 20\}$. The symbol $\times$ indicates the maximum number of function evaluations. The bottom right plot shows the aggregated data profiles over all the decision space dimensions tested. 136

6.2 $I^1_{\epsilon+}$-based data profiles aggregating over all 300 problems for a decision space dimensionality $n \in \{2, 3, 5, 10, 20\}$. The symbol $\times$ indicates the maximum number of function evaluations. The bottom right plot shows the aggregated data profiles over all the decision space dimensions tested, where the term *lcm* denotes the least common multiple operator. 137

6.3 I^-_H-based data profiles aggregated over problem categories across all the dimensions tested. The symbol $\times$ indicates the maximum number of function evaluations, whereas the term *lcm* denotes the least common multiple operator. 138

6.4 $I^1_{\epsilon+}$-indicator-based data profiles aggregated over problem categories across all the dimensions tested. The symbol $\times$ indicates the maximum number of function evaluations, whereas the term *lcm* denotes the least common multiple operator. 139

6.5 Data profiles aggregated over problem categories for the hypervolume and additive epsilon quality indicators computed. The symbol $\times$ indicates the maximum number of function evaluations. 144

6.6 Data profiles aggregated over problem categories for the generational and inverted generational distance quality indicators computed. The symbol $\times$ indicates the maximum number of function evaluations. . 145

6.7 Data profiles aggregated over all the problems across all the quality indicators computed for each of the compared algorithms. The symbol $\times$ indicates the maximum number of function evaluations. . 145

List of Tables

2.1 Common challenges in black-box optimization. The present thesis does not address problems that are noisy (stochastic) and/or dynamic. 13

2.2 Commonly-used quality indicators [for more details, see 2, 3, 4]. . . 36

3.1 Convergence rate of different MSO algorithms. These rates hold provided that Assumptions 1, 2, & 3 are satisfied. The *Condition* column provides the relation between the number of expansions p and the depth $h(p)/s(p)$ besides other algorithm-specific conditions. h_{max} and s_{max} define the maximum depth for the trees of `SOO` and `MCS`, respectively. 72

4.1 List of MSO algorithms considered for empirical analysis and their implementation. We have used our own implementation of `BaMSOO` [5]. 85

4.2 Parameters setting of `NMSO`. 85

4.3 Algorithms Ranking in BBComp within GECCO'2015 98

6.1 The compared algorithms and their implementation sources. 132

6.2 Parameters setting of `MO-SOO`. In accordance with many single-objective MSO algorithms, we set the partition factor K to be 3 and the cells partitioning to be along one dimension per depth in a sequential fashion. Given a finite evaluation budget $v(t)$ at iteration t, we have simulated the effect of exploration-vs.-exploitation trade-off by offsetting the maximal depth $h_{max}(t)$ from $h_o(t)$, the depth of the shallowest leaf node (i.e., the minimal depth at iteration t), by the depth of the uniform tree that would have been built using double the evaluation budget. Furthermore, as the algorithm splits one dimension per depth, we considered an additive factor proportional to the decision space n. We found a factor of $n^{1.5}$ generalize well over the range of dimensions considered $n \in \{2, 3, 5, 10, 20\}$. 133

6.3 The modified parameters of `MOEA/D`, `MO-CMA-ES`, and `SMS-EMOA` according to Brockhoff et al. [6]. 133

6.4 Test problems definition and properties. Symbols: **D** : dimensionality $\in$ {L : low-dimensionality, H : high-dimensionality}; **S** : separability $\in$ {S : separable, NS : non-separable}; **M** : modality $\in$ {U : uni-modal, M : multi-modal}; $\times$: uncategorized/mixed. 141

6.5 Employed Quality Indicators. Adapted from Hadka [2] (for more details, see [3, 4]). 142

List of Abbreviations

ACO	Ant Colony Optimization.
AFSA	Artificial Fish School Algorithm.
BaMSOO	Bayesian Multi-Scale Optimization.
BB-LS	Branch-and-Bound and Local Search.
BBComp	Black-box Optimization Competition.
BBOB	Black-Box Optimization Benchmarking.
BFO	Bacterial Forging Optimization.
BMOBench	Black-box Multi-objective Optimization Benchmarking.
BO-BBOB	Bi-Objective Black-Box Optimization Benchmarking.
Bsrr	Brent-STEP with Round-Robin.
COCO	Comparing Continuous Optimizers.
DFO	Derivative-Free Optimization.
DIRECT	DIviding RECTangles.
DMeSR-PSO	Dynamic Mentoring and Self-Regulation based Particle Swarm Optimization.
DMS	Direct MultiSearch.
DOE	Design of Experiments.
DOO	Deterministic Optimistic Optimization.
EA	Evolutionary Algorithm.
ECDF	Empirical Cumulative Distribution Function.
ERT	Expected Running Time.

GP	Gaussian Process.
GPS	Generalized Pattern Search.
HOO	Hierarchical Optimistic Optimization.
KNITRO	Nonlinear Interior point Trust Region Optimization.
LO	Lipschitzian Optimization.
MADS	Mesh Adaptive Direct Search.
MCS	Multilevel Coordinate Search.
MO-DIRECT	Multi-Objective DIviding RECTangles.
MO-SOO	Multi-Objective Simultaneous Optimistic Optimization.
MOEA/D	Multi-objective Evolutionary Algorithm Based on Decomposition.
MOP	Multi-objective Optimization Problem.
MP	Mathematical Programming.
MSO	Multi-scale Search Optimization.
MULTIMADS	Multi-objective Mesh Adaptive Direct Search.
MVMO	Mean-Variance Mapping Optimization.
NMSO	Naive Multi-scale Search Optimization.
PSO	Particle Swarm Optimization.
RF5-CMA-ES	Covariance Matrix Adaptation Evolution Strategy with Random Forests.
RL-SHADE	Success-History based Adaptive Differential Evolution with Restart and Linear population size reduction strategy.
SMS-EMOA	$\mathcal{S}$ Metric Selection Evolutioanry Multi-objective Optimization Algorithm.
SOO	Simultaneous Optimistic Optimization.
SOP	Single-objective Optimization Problem.
SRPSO	Self-Regulating Particle Swarm Optimization.
StoSOO	Stochastic Simultaneous Optimistic Optimization.

List of Notation

$\mathcal{A}_s$	sequential algorithm
$\mathcal{A}_p$	passive algorithm
n	number of decision variables
n_s	number of expansion sequences
m	number of objective fuctions
$\mathcal{X}$	feasible decision space
$\mathcal{Y}$	objective space
$2^{\mathcal{X}}$	power set of $\mathcal{X}$
f	objective function
$\mathbf{f}$	multi-objective function
f_j	jth objective function for a multi-objective problem
$\mathcal{Y}_j$	jth objective space for a multi-objective problem
$\mathcal{X}^*$	Pareto optimal set
$\mathcal{Y}^*$	Pareto front
$\mathcal{Y}_*^t$	approximation set at iteration t
$\mathbf{x}$	decision vector (solution/point)
$\mathbf{x}^*$	optimal solution/point
f^*	optimal objective value
$\mathbf{y}^*$	ideal objective vector (point)
v	number of function evaluations (evaluation budget)
p	number of expansions
t	number of iterations
r	loss (regret)

J	number of function evaluations per node
K	partition factor
L	Lipschitz constant
P	number of evaluated nodes per iteration
Q	number of expanded nodes per iteration
V	number of visits for a node
$\mathcal{B}$	set of *good enough* nodes
R	reference set
A	set of objective vectors
$\texttt{ND}$	non-dominated operator
I_{IGD}	inverted generational distance indicator
I_{GD}	generational distance indicator
$I^1_{\epsilon+}$	additive epsilon indicator (with respect to a reference set R)
I^-_H	binary hypervolume indicator (with respect to a reference set R)
Ψ	conflict dimension
d_v	v-near-optimality dimension
$\mathcal{P}$	set of to-be-evaluated nodes
$\mathcal{Q}$	set of expandable nodes
h^*	depth of deepest optimal node
$\mathcal{I}^\epsilon_h$	set of ϵ-optimal nodes at depth h
(h, i)	node of depth h and index i
$\delta(h)$	decreasing sequence in h
$\mathbf{y}^{nadir}(\hat{\mathcal{Y}})$	nadir point of the $\hat{\mathcal{Y}}$ region
$\mathcal{T}$	space-partitioning tree
$\mathcal{L}$	leaf nodes of space-partitioning tree $\mathcal{T}$
ℓ	semi-metric
ℓ^2	Euclidean norm

Chapter 1

Introduction

“Begin at the beginning,“ the King said gravely, “and go on till you come to the end: then stop.“

- Lewis Carroll, *Alice in Wonderland*

1.1 Motivation

Numerical optimization has been a recurring subject of interest for centuries, as many decision-making problems for real-world systems can be described by objective functions of decision variables that model and capture such systems. Examples range from transmembrane protein structure determination [7] and molecular docking [8] to designing hybrid fuel cell vehicles [9] and controlling wet clutches [10].

Nevertheless, the complexity and difficulty of such problems are growing steadily. On the one hand, the objective functions may often be multi-variate, non-differentiable, ill-conditioned, and/or multi-modal [11]. On the other hand, it is not uncommon that derivatives of the objective functions are neither symbolically nor numerically available [12], and therefore the objective functions are only accessible through a *black box* (e.g., a computer code or a laboratory experiment) where one can provide a solution (a point in the decision space) to the black box and gets in return the objective value at that solution. In other words, the only source of information about the objective is point-wise evaluations. Optimizing such objectives is often referred to as *black-box* optimization. In various real-world black-box problems (see, e.g., [13, 14, 15, 16, 17]), evaluating a solution is typically expensive, requiring some computational resources (e.g., time, power, money), and hence one seeks the fastest possible search to achieve the best possible solution.

The aforementioned challenges of black-box optimization have been studied to a great extent by the Mathematical Programming (MP) community. MP methods, compared with others, are generally devised to show a systematic convergence towards the optimal solution from the beginning of the search—by making use of all the points sampled to guide the search and/or construct models of the objective function. Moreover, the asymptotic convergence is often provable due to their systematic sampling of the decision space. Examples of MP algorithms solving real-world black-box problems can be found in applications ranging from aircraft routing to maximum power tracking for a photovoltaic system [18, 19, 20, 21, 22].

Many of MP algorithms for continuous black-box optimization have been developed and *analyzed independently*, for which the theoretical studies have been dominantly concerned with the *limit* behavior of the algorithms and their *asymptotic* convergence to optimal/stationary points (e.g., [23, 24, 25, 26, 27, 28, 29, 30, 31, 32]). Few papers, such as those by Hansen et al. [33] and Munos [34], have studied their *finite-time* behavior, which is more crucial in practice [35], given the finite constraints on the computational budget (the number of function evaluations). Moreover, finite-time analysis lays the foundations towards possible directions of research for efficient algorithms.

On the one hand, several black-box MP algorithms are—attributable to their structured sampling—inherently exploratory: behaving, sometimes, as an exhaustive grid search [36]. This, among other factors, results in a slow convergence [37], which is less of a desired attribute with the growing computational complexity of real-world problems, as mentioned earlier. On the other hand, exploitative (greedy) algorithms may not adapt well to the complexity of the problem at hand (e.g., multi-modality) converging to a local stationary point.

Additionally, a substantial number of problems encountered in practice are of multiple objectives: Multi-objective Optimization Problems (MOPs)—in contrast to Single-objective Optimization Problems (SOPs)—involve a set of (often) conflicting objectives that are to be optimized simultaneously (see, e.g., [38, 39, 40, 41, 42]). With conflicting objectives, there does not exist a single optimal solution, but a set of incomparable optimal solutions: each is inferior to the other in some objectives and superior in other objectives. This induces a partial order on the set of objective vectors of an MOP. The set of optimal objective vectors according to this partial order is commonly named as the *Pareto front* of the problem. The task of MOP solvers therefore becomes finding the Pareto front or producing a good approximation of the same. Although Evolutionary Algorithms (EAs) have been applied with empirical success to multi-objective problems, there has been a growing gap between EAs' theory and practice [43]. Besides, among the several lessons learned from developed MOP solvers over the past decades is that, in order to generate a dense and good approximation set of the Pareto front, one must maintain the set diversity. Furthermore, one must not discard inferior solutions too easily, as some of them may pave the way towards rarely-visited regions of the Pareto front [44]. In other words, the exploration-vs.-exploitation trade-off in search for the Pareto optimal set should be thought carefully about, at the *algorithmic design level*. To this end, one can tailor MP techniques towards MOPs leveraging their exploratory nature and, at the same time, seek conceptual insights making use of their theoretical guarantees.

In the light of this discussion, three key points arise with regard to black-box MP algorithms:

- With the growing bulk of computationally expensive black-box problems, there is a need to be able to gauge the performance of black-box MP algorithms against the number of function evaluations used (or any time-related quantity). In other words, a principled methodology is required to study the *finite-time* performance of black-box MP algorithms.

- To deal with computationally expensive optimization, there is a need for techniques that prioritize exploitation over exploration given a very expensive computational budget, yet preserve sound theoretical properties as those of within the MP framework.

- As plenty of real-world problems involve multiple objectives, the theoretical and exploratory properties of MP algorithms make them an attractive candidate to solve MOPs. In other words, there is a need to extend and investigate the suitability of black-box MP algorithms for MOPs.

In this thesis, the above points are addressed with regard to a special family/class of MP algorithms that employ a divide-and-conquer tree search by partitioning the continuous decision space in a hierarchical fashion given a finite number

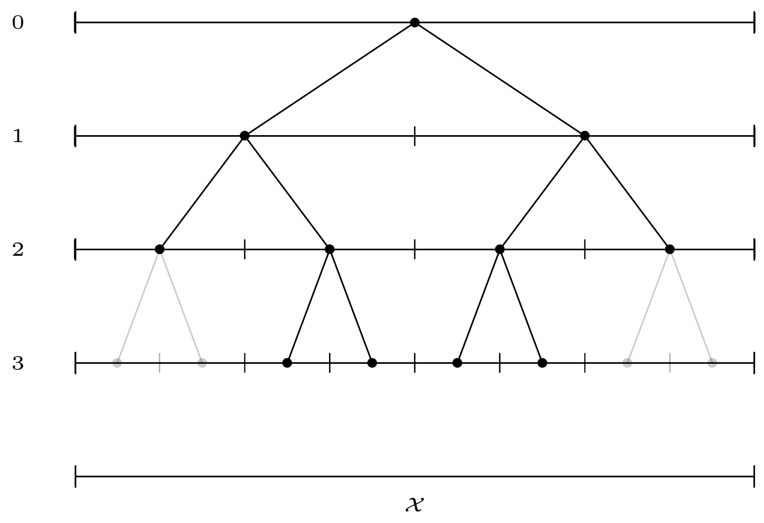


Figure 1.1: A binary search tree built with 11 function evaluations over the decision space $\mathcal{X}$. The tree constructed, in a hierarchical fashion, four $\{0, 1, 2, 3\}$ partitions of the search space of different scale/granularity.

of function evaluations (see, Fig. 1.1). These algorithms, iteratively, carry out a dynamic expansion of a tree whose nodes represent subspaces of the decision space, with the root corresponding to the entire search space. The space-partitioning tree grows seeking a trade-off between exploration and exploitation of the decision space: a dilemma whose initial investigations date back to Thompson [45] and Robbins et al. [46], where it was known as the *multi-armed bandit* problem. Tree-based frameworks have a big body of literature in such various fields as artificial agents, control, and planning [47]. They have recently been gaining popularity [e.g., game of GO, 48] and witnessed a sensational success [49].

With this regard, global continuous black-box optimization can be modeled as a structured bandit problem where the objective value is a function of some arm parameters (see, for instance, [50, 51]). Based on the observations (sampled points) and assumptions about the objective smoothness, an optimistic strategy would compute a bound on the objective (reward) value at each solution (arm) $\mathbf{x} \in \mathcal{X}$ and choose the arm with the best bound. Examples of global continuous optimization algorithms with a closely related approach are Lipschitzian optimization techniques [52]. However, this approach poses two problems: i) the computational complexity of computing the bounds over $\mathcal{X}$ at each step; ii) the restriction that the smoothness assumption puts on the objective functions that can be optimized. While the second issue can be addressed with weak, yet effective assumptions on the function smoothness, e.g., local (rather than global) smoothness; the first issue can be alleviated by transforming the problem from a many-arm bandit to a hierarchy of multi-armed bandits (often referred to as *hierarchical bandits* [53]). Hence, such an algorithm can be regarded as a tree-search divide-and-conquer technique that iteratively constructs finer and finer partitions of the search space $\mathcal{X}$ at multiple scales $h \in \mathbb{N}_0$. Given a scale $h \geq 0$ and a partition factor $K \geq 2$, $\mathcal{X}$ can be partitioned into a set of K^h cells/hyperrectangles/subspaces $\mathcal{X}_{h,i}$ where $0 \leq i \leq K^h - 1$ such that $\cup_{i \in \{0,\ldots,K^h-1\}} \mathcal{X}_{h,i} = \mathcal{X}$. These cells are represented by nodes of a K-ary tree $\mathcal{T}$ (as shown in Figure 1.1), where a node (h, i) represents the cell $\mathcal{X}_{h,i}$—the root node $(0, 0)$ represents the entire search space $\mathcal{X}_{0,0} = \mathcal{X}$. A parent node possesses K child nodes $\{(h+1, i_k)\}_{1 \leq k \leq K}$, whose cells $\{\mathcal{X}_{h+1,i_k}\}_{1 \leq k \leq K}$ form a partition of the parent's cell $\mathcal{X}_{h,i}$.

1.2 Objectives

This research is concerned with Mathematical Programming (MP) algorithms for continuous black-box optimization given a finite budget of function evaluations. In particular, the present thesis focuses on MP algorithms, which search for the (or one) optimal solution over multiple scales of the decision space of problems that are of one (or more) objective function(s). The objectives of the present thesis can be summarized as follows.

- **Finite-Time Analysis**. In literature, the bulk of convergence analysis studies on the aforementioned algorithms has been primarily concerned with their infinite-time behavior, whereas finite-time analysis is more relevant in a finite-budget setting. Hence, providing a principled, generic methodology to study the finite-time behavior is a major objective of this research.

- **Expensive Optimization**. Fueled by the increasing number of computationally expensive real-world optimization problems, this research seeks to design a black-box MP algorithm, aimed at expensive optimization, prioritizing exploitation over exploration whilst providing solid theoretical guarantees on its convergence.

- **Multi-Objective Optimization**. In practice, several problems encountered are of multi-objective nature. However, when compared to Single-objective Optimization Problems (SOPs), a few MP algorithms have been tailored towards Multi-objective Optimization Problems (MOPs). This opens the door to a promising evaluation of black-box MP techniques for MOPs, leveraging their theoretical and exploratory properties. Therefore, another major objective of this research is to investigate the suitability of MP algorithms for black-box multi-objective optimization.

1.3 Thesis Contributions

This thesis brings several contributions to the theory and algorithms of the Mathematical Programming (MP) field for continuous black-box optimization given a finite budget of function evaluations. These contributions revolve around a special family of algorithms that employ a space-partitioning tree search. The present thesis refers to such algorithms as *Multi-scale Search Optimization* (MSO), as they search for the (or one) optimal solution over multiple scales of the decision space. In essence, the contributions presented in this thesis can be categorized into three parts: i). theory; ii). algorithms; and iii). benchmarking of MSO algorithms. These contributions are highlighted next, respectively.

1.3.1 Theoretical Analysis Contributions

This thesis provides a generic framework for black-box Multi-scale Search Optimization (MSO) algorithms and proposes a theoretical methodology to analyze their finite-time convergence based on three basic assumptions: a) Hölder continuity of the objective function; b) partitions boundedness; and c) partitions sphericity. The analysis is built on quantifying how much exploration is required to achieve near-optimal solutions. Based on this methodology, the obtained theoretical results are two-fold:

- *Single-Objective Optimization.* The worst-case finite-time performance and convergence rate of several established MSO algorithms, namely, Lipschitzian Optimization (`LO`) [54, 55], Multilevel Coordinate Search (`MCS`) [27], DIviding RECTangles (`DIRECT`) [56], and optimistic optimization methods [34] are presented.

- *Multi-Objective Optimization.* The finite-time analysis presented in the single-objective setting is extended towards MOPs, where a hypothetical measure of objectives conflict is proposed and is referred to as the conflict dimension. The conflict dimension captures the proximity of Pareto front extrema to the rest of its elements. In the light of this measure, the present research establishes an upper bound on the Pareto-compliant unary additive epsilon indicator characterized by the objectives smoothness as well as the conflict dimension. First time in the literature, a deterministic upper bound on a Pareto-compliant indicator has been presented for a solver of continuous MOPs.

Additionally, the concluded theoretical results are empirically validated on a set of synthetic problems using *symbolic maths*.

1.3.2 Algorithmic Design Contributions

This thesis proposes and analyzes three algorithms to deal with expensive as well as multi-objective optimization:

- *Expensive Single-Objective Optimization.* The bulk of MSO algorithms are exploratory in nature due to their systematic sampling. The present research proposes an algorithm within the framework of MSO and refers to it as the Naive Multi-scale Search Optimization (`NMSO`) algorithm. Similar to other MSO algorithms, `NMSO` builds a partitioning tree over the decision space. The tree is expanded using the following rule: *exploit until no further improvement is observed.* Compared to other MSO algorithms, `NMSO` prioritizes exploitation over exploration, which makes it suitable for computationally expensive optimization problems. Despite being exploitative (greedy), `NMSO` enjoys a theoretical finite-time and asymptotic convergence within the established theoretical methodology. At the Black-box Optimization Competition (BBComp) [57], `NMSO` emerged third out of twenty-eight competitors in solving 1000 black-box problems.

- *Multi-Objective Optimization.* To expand the frontier and investigate the suitability of MSO algorithms in solving MOPs, two algorithmic instances are proposed: i). Multi-Objective DIviding RECTangles (`MO-DIRECT`) based on DIRECT [56]; and ii). Multi-Objective Simultaneous Optimistic Optimization (MO-SOO) based on SOO [34]. While both of these algorithms enjoy an asymptotic convergence towards the Pareto front, the iterative sweep over multiple scales of the search by MO-SOO makes its finite-time performance provable within the proposed theoretical framework. Moreover, the exhaustive exploratory nature of MO-DIRECT makes it an attractive candidate to initialize local search multi-objective solvers. On the other hand, MO-SOO serves as a suitable solver for problems with a limited number of function evaluations.

1.3.3 Benchmarking Contributions

Empirical analysis employs experimental simulations of algorithms on complex problems, gaining an insight on the their practicality/applicability on real-world problems. In order to complement the theoretical perspective and the algorithmic development in the present thesis, an empirical analysis has been conducted to validate and compare the effectiveness of MSO algorithms. To this end, the experimental evaluation must be able to distinguish the good features of the algorithms and show their differences over different stages of the search for various goals and situations. With these concerns, the contributions are of two folds:

- *Single-Objective Optimization.* A thorough empirical analysis and comparison of several MSO algorithms—viz. classical (DIRECT, MCS), commercial (BB-LS) and recent (SOO, BaMSOO) algorithms—against NMSO have been conducted on the noiseless Black-Box Optimization Benchmarking (BBOB) testbed under both expensive and cheap-budget settings. The Comparing Continuous Optimizers (COCO) methodology [58] has been adopted as it meets the above concerns in benchmarking. Furthermore, recently developed stochastic algorithms, namely SRPSO [59], DMeSR-PSO [60], RL-SHADE [61], RF5-CMA-ES [62], and Bsrr [63] have as well been compared with NMSO. Overall, NMSO is suitable for problems with a small number of function evaluations, low-dimensionality search space, and objective functions that are separable or multi-modal. Otherwise, it is comparable with the top performing algorithms.

- *Multi-Objective Optimization.* Several test problems for validating multi-objective algorithms have been proposed (see, e.g., [64]). However, most of the time, methods proposed to solve MOPs are benchmarked on a different set of problems under arbitrary budgets of function evaluations. Hence and in line with ongoing efforts of the community in consolidating testbeds for black-box optimization (e.g., BBOB and BBComp), the Black-box Multi-objective Optimization Benchmarking (BMOBench) platform is built around 100 multi-objective optimization problems from the literature collected by Custódio et al. [65]. In BMOBench, the performance assessment is reported in terms of Pareto- and Non-Pareto-compliant data profiles. BMOBench can be employed as a comprehensive platform for benchmarking any multi-objective optimization algorithm. Besides the built platform, 300 bi-objective from Brockhoff et al. [6] were used in comparing the proposed multi-objective MSO algorithms with several stochastic and deterministic algorithms. The results substantiate the efficacy of `MO-SOO` especially for expensive MOPs.

1.4 Thesis Organization

The thesis is organized as follows:

- In **Chapter 2**, a historical overview of Mathematical Programming (MP) techniques in the context of continuous black-box optimization is presented, followed by a literature review on the family of space-partitioning tree-search MP algorithms. Here these algorithms are grouped into two categories: one category requires the knowledge about the objective function smoothness, while the other category makes an assumption about the smoothness whose knowledge may not be available.

- **Chapter 3** addresses the class of space-partitioning tree-search MP algorithms for continuous black-box optimization and provides a generic procedure referred to as Multi-scale Search Optimization (MSO). Moreover, a principled theoretical methodology is presented to study the finite-time behavior of MSO algorithms. The approach is based on three assumptions: i). local Hölder continuity of the objective function; ii). partitions boundedness; and iii). partitions sphericity. In the light of this analysis, the worst-case finite-time performance of several leading MSO algorithms is presented and validated empirically using symbolic maths.

- **Chapter 4** tackles expensive optimization and proposes `NMSO` (short for Naive Multi-scale Search Optimization): an MSO algorithm with a finite-time and asymptotic provable performance, sitting at the exploitation end of exploration-vs.-exploit- ation dilemma. `NMSO`'s analysis provides the basis for analyzing exploitation-biased MSO algorithms. Besides validating the empirical performance of `NMSO`, a thorough numerical validation and comparison of the classical, commercial, as well as recent MSO algorithms is conducted on the noiseless Black-Box Optimization Benchmarking (BBOB) testbed for both expensive- and cheap-budget settings.

- In **Chapter 5**, the frontier of MSO algorithms is extended towards multi-objective optimization in terms of algorithmic development and theoretical analysis. In particular, the present research provides multi-objective algorithmic instances of `DIRECT` and `SOO`, which we refer to as `MO-DIRECT` and `MO-SOO`, respectively. This chapter shows that both algorithms enjoy an optimal limit behavior. Moreover, it analyzes the finite-time performance of `MO-SOO` and validates its theoretical guarantees on a set of synthetic problems. The finite-time analysis establishes an upper bound on the Pareto-compliant unary additive epsilon indicator characterized by the objectives smoothness as well as the structure of the Pareto front with respect to its extrema.

- **Chapter 6** complements Chapter 5 by validating the empirical performance of the two developed multi-objective solvers, viz. `MO-DIRECT` and `MO-SOO` on a set of multi-objective problems. To this end, the algorithms are tested on two testbeds: i). 300 bi-objective problems generated from the BBOB testbed and ii). the Black-box Multi-objective Optimization Benchmarking (BMOBench), which is built to consolidate 100 MOPs from the literature. The algorithms are also compared with state-of-the-art deterministic and stochastic multi-objective solvers in terms of different quality indicators (e.g., the additive epsilon indicator) in the form of data profiles.

- **Chapter 7** concludes the thesis and provides directions for future research work.

Chapter 2

Literature Review

"The greatest part of a writer's time is spent in reading, in order to write: a man will turn over half a library to make one book."

- Samuel Johnson, *The Life of Samuel Johnson LL.D. Vol 2*

This chapter is intended as a general overview of Mathematical Programming (MP) techniques for continuous black-box optimization. After briefly motivating black-box problems in Section 2.1, a historical summary of black-box MP methods is presented in Section 2.2. This is followed by a literature review on the family of space-partitioning tree-search MP algorithms in Section 2.3. These algorithms are grouped into two categories: one category requires the knowledge about the objective function smoothness. On the other hand, the other category just makes an assumption about its smoothness whose knowledge may not be available. Towards the end of this chapter, a formal background on multi-objective optimization is presented. Lastly, some comments are made with regard to the objectives of this thesis.

2.1 Continuous Black-Box Optimization

In various real-world applications and industries, products are designed using computer/ math models that alleviate the need for physical prototypes and ease the exploration of alternative designs [66]. To achieve an optimal design, optimization algorithms are tailored to query these models for a specific design to get the corresponding objective value. In other words, the objective function is evaluated through a computer code with no *analytical expression* nor knowledge about its *structure* nor *derivatives*. Besides computer-based models, there are chemical, mechanical, and aeronautical problems based on laboratory experiments where information other than objective function values are impossible or difficult to collect. Such scenarios are commonly referred to as *black-box* optimization. Without loss of generality, the numerical (continuous) deterministic black-box minimization problem with n decision variables has the form:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{X} \ , \end{aligned} \tag{2.1}$$

given v function evaluations, where $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}$ is the black-box objective function defined on decision (search) space $\mathcal{X}$ subset of $\mathbb{R}^n$. The sole source of information about f is *point-wise evaluations*: one can query f at any point $\mathbf{x} \in \mathcal{X}$ and the black box will return the corresponding objective value $f(\mathbf{x})$. Therefore, the black-box optimization problem can be looked at as a search for the (or one) optimal solution $\mathbf{x}^*$ within the decision space $\mathcal{X}$ given a finite number of function evaluations v, since only zero-order information is available. While this may seem a simple task, the objective function f often comes with a number of challenges that make solving (2.1) or finding a near-optimal solution tedious. Table 2.1 briefly discusses typical challenges encountered in real-world problems.

Black-box optimization is also referred to as Derivative-Free Optimization (DFO), a term coined within the MP community, in contrast to optimization techniques with derivatives (e.g., Newtonian methods [67]). The next section presents a historical summary of mathematical programming methods for continuous black-box optimization.

2.2 Historical Overview of Black-Box Mathematical Programming

Let $\mathbf{x}^*$ be the (or one) solution to (2.1) such that $f^* = f(\mathbf{x}^*)$. A simple, yet not so efficient, approach to find any solution $\mathbf{x}^*$ would be to use a *passive* algorithm $\mathcal{A}_p$, which would use its v-evaluation budget at once to evaluate a uniform grid of v points in the objective function's domain $\mathcal{X}$. $\mathcal{A}_p$ then returns $\mathbf{x}(v)$, the point in the grid with the best function value among the grid points, as a guess on $\mathbf{x}^*$ (see, e.g., [72]). Alternatively, a sequential algorithm $\mathcal{A}_s$ would start by arbitrarily

Table 2.1: Common challenges in black-box optimization. The present thesis does not address problems that are noisy (stochastic) and/or dynamic.

Challenge	**Description**
Computational Cost	A problem is said to be *expensive* if the cost (e.g., time, power, money) of optimizing it per an (function) evaluation is less than the cost of one function evaluation [44] (e.g., vehicle crash simulation could take 20 hours [56]).
Non-Separability	A problem is *separable* if the optimum can be found by performing n one-dimensional independent searches along each of the n coordinates. Thus, the difficulty of separable problems increases linearly with the decision space dimension n, whereas non-separable problems—e.g, see Fig. 2.1 (a)—face the curse of dimensionality [68] with higher dimensions [69] as discussed next.
High-Dimensionality	Besides the proportional computational complexity, the volume of search space scales exponentially with n. In order to get a coverage in 10-dimensional unit hypercube similar to the coverage of 10 points in a unit interval, 10^{10} points are required.
Multi-Modality	The presence of more than one optimum. This could lead to premature convergence to a local optimum—e.g, see Fig. 2.1 (b).
Ill-Conditioning	A problem is *ill-conditioned* if different decision variables contribute differently to the objective function—e.g, see Fig. 2.1 (c). In other words, different directions in the decision space exhibit a considerably-different contribution (orders of magnitude difference) to the objective function value [69].
Multi-Objective	Optimizing m objective functions $\{f_j\}_{1\leq j\leq m}$ is referred to as a Multi-objective Optimization Problem (MOP). The solution is generally a set of incomparable optimal points. More will be discussed in Section 2.4.2.
Dynamic	A problem is *dynamic* if the objective function value of a feasible solution depends on time [70].
Accuracy/Noise	Sometimes, the function evaluation accuracy depends on procedures that involve random errors (e.g., chemical reactions) or the computational time allocated to it (e.g., the finer is the grid size in solving a partial differential equation, the more accurate will be the result) [71].
Ruggedness	Similar to noise, non-differentiability / non-continuity can be a source of difficulty in black-box optimization [35]—e.g, see Fig. 2.1 (d).

guessing/selecting a point in the search space and evaluating it on f. $\mathcal{A}_s$ then iteratively refines its next guess based on the previously selected points and their corresponding function f values. After v evaluations, $\mathcal{A}_s$ returns $\mathbf{x}(v)$: its best guess on $\mathbf{x}^*$ (see, e.g., [54]). Since there is no prior knowledge on f, there is

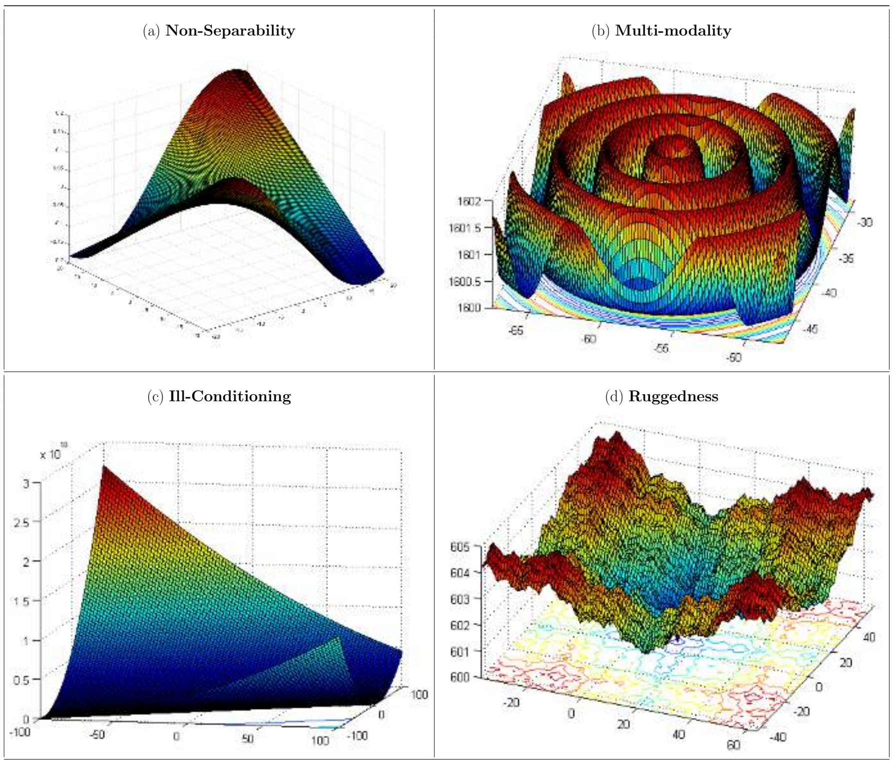


Figure 2.1: 3-dimensional maps for 2-dimensional functions exhibiting typical challenges in numerical black-box optimization. Figures (b), (c), and (d) are adapted from Chen et al. [1].

no guarantee that $\mathbf{x}(v)$ of either $\mathcal{A}_p$ nor $\mathcal{A}_s$ is the solution to the Problem (2.1). While passive algorithms are simple and easy, sequential algorithms can achieve solutions of better quality given the same computational budget. It was shown independently by Ivanov [73] and Sukharev [74] that the evaluation budget needed to find a solution with a desired degree of quality is the same for the best passive and sequential algorithms in the worst case for some functions, whereas for most other functions, the required evaluation budget for a passive algorithm is much greater than that of a sequential algorithm.

One can look at sequential algorithms as *v-round sequential decision-making processes under uncertainty* where a future *action* (selecting a candidate solution) depends on past actions (previously selected points) and their *rewards* (observed function values). A principal question is: how can $\mathcal{A}_s$ identify, as quickly as possible, the most rewarding action to select next? Intuitively, $\mathcal{A}_s$ would *explore* the set of possible actions it can take to know more about f and as its knowledge improves, $\mathcal{A}_s$ should increasingly *exploit* its current knowledge by selecting what it believes to be the best action. Clearly, the trade-off between *exploration* and *exploitation* affects $\mathcal{A}_s$'s returned solution $\mathbf{x}(v)$. The quality of the returned solution, as a function of the number of function evaluations v, is evaluated by the following *regret* (or *loss*) measure:

$$r(v) = f(\mathbf{x}(v)) - f^* \tag{2.2}$$

Such measure also serves as an indicator of $\mathcal{A}_s$'s convergence rate with respect to v, as it measures how efficient $\mathcal{A}_s$ is, in capturing enough information about f to produce a solution with a desired degree of accuracy.

The literature on sequential decision-making problems under uncertainty is rich and large with applicability in various scientific fields (e.g., machine learning, planning, and control). Initial investigations on such problems date back to Thompson [45] in 1933 and Robbins et al. [46] in 1952, where they were known as *multi-armed bandit* problems. Since then, much progress has been made to address the exploration vs. exploitation dilemma. Many strategies were proposed, studied, and analyzed, such as ϵ-greedy [75], softmax [76], Bayesian [77], upper-confidence-bound [75], and optimistic [78] strategies. With respect to *numerical black-box optimization*, (sequential) techniques can be grouped into one of two groups:

- Heuristic algorithms that provide only probabilistic guarantees on their convergence.
- Algorithms, with roots in the field of Mathematical Programming (MP), that often guarantee asymptotic convergence to an optimal solution (or a good approximation with a specified accuracy).

Heuristic Algorithms. Stochastic algorithms (particularly, evolutionary algorithms) are one established class of the former group. These evaluate the objective function at a *suitably chosen random collection* of points and subsequently attempt to approach a local (or hopefully global) optimal solution. They try to mimic the nature, viz. concepts from physics and biology, in solving optimization challenges. Examples include genetic algorithms [79], Ant Colony Optimization (ACO) [80], Bacterial Forging Optimization (BFO) [81], Particle Swarm Optimization (PSO) [82], Artificial Fish School Algorithm (AFSA) [83], and a lot more. In essence, these algorithms keep a population of entities (ants in ACO, bacteria in BFO, birds in PSO, etc.), each of which represents a possible solution for the problem at hand, and iteratively construct a new population of solutions based on the previously constructed population. Although there are several reasons that make the search for $\mathbf{x}^*$ difficult as discussed in Table 2.1, the majority of these algorithms have been designed with a strong emphasis on multi-modality problems [35] and they require some time (more function evaluations) to move towards encouraging regions of the search space [44]. Furthermore, the algorithmic development and validation of such methods have been mainly empirical; there has been a growing gap between their theory and practice [43], where convergence to the global optimum is only guaranteed with a positive probability. Nevertheless, they have been applied with success to several real-world problems [84, 85, 86] and several recent studies have been addressing their convergence speed [35].

MP Algorithms. On the other hand and with origins in the MP community, the latter group is generally tailored to show a systematic convergence towards the optimum from the initial phase of the search. MP methods make use of all the points sampled in an iterative manner to direct the search and/or construct models of the objective function. Moreover, the asymptotic convergence is often provable due to their structured sampling of the search space. Dating back to the fifties of the twentieth century, the field of black-box MP optimization—or Derivative-Free Optimization (DFO)[1]—has a long history and a huge body of literature, which is motivated by the growing number of black-box applications (e.g., [88]) and advances in computational paradigms. Initial investigations and empirical algorithmic developments in this field were pioneered by Piyavskii [54], Shubert [55], Box and Wilson [89], Fermi and Metropolis [90], Box [91], Hooke and Jeeves [92], Spendley et al. [93], Nelder and Mead [94], and Strongin [95, 96], as described next.

In 1951, Box and Wilson [89] built on the work of Fisher [97] in the field of Design of Experiments (DOE) to identify optimal conditions in chemical investigations

[1]The use of this term in the literature has been often confusing and inconsistent. While there exist some works that reserve the term *derivative-free algorithms* for a special class of black-box MP algorithms, others generalize it to include both heuristic and MP methods [87]. In this thesis, this term is used to refer to black-box algorithms from the MP community in line with the classical mathematical optimization algorithms with derivatives.

and after 6 years, Box [91] proposed a method to increase industrial productivity and assess the effects of changes in variables controlling manufacturing processes. The method, referred to as Evolutionary Operation, resembles coordinate search where a small hyperrectangle is constructed around the current solution, with its corners representing slightly-perturbed solutions of the manufacturing process compared to the current solution. The quality of these solutions (their corresponding objective function values) is then estimated and the best-quality solution is chosen to be next center/mean of the hyperrectangle. On the other hand, Fermi and Metropolis [90] used one of the first digital computers to vary one variable at a time via steps of the same size that get halved if no improvement is observed. The procedure is repeated iteratively until a stopping criterion with regard to the step size is met.

Direct-Search Methods. Likewise, Hooke and Jeeves [92] adopted the iterative framework of function evaluations at a sample of points/solutions. It is generally agreed that they have been the first to use the term *direct search* in the DFO community to *“describe sequential examination of trial solutions involving comparison of each trial solution with the ”best” obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be”* [92]. Their work recognized the notion of *pattern search* as a specific kind of strategy, where *exploratory* moves to identify the behavior of the objective function f are employed, in addition to so-called *pattern* moves along probable directions in the search space $\mathcal{X}$ towards a better solution. Audet and Dennis Jr [98] extended Hooke and Jeeves’ method and introduced Generalized Pattern Search (GPS) methods for unconstrained optimization unifying similar algorithmic variants using the notion of translated, scaled integer lattices within the search space $\mathcal{X}$.

Simplex-based Methods. Spendley et al. [93] put the foundations to *simplex-based* algorithms in 1962. In search for the optimal solution, they suggested using the vertices of an $n+1$-simplex within $\mathcal{X}$. Based on the vertices’ objective function values, the method attempts to improve the worst vertex of the simplex by isometrically reflecting it with respect to the mean of the other n vertices. Otherwise, the same process is repeated to the second worst vertex. Benefited by the notion of simplex designs, Nelder and Mead [94] extended Spendley et al. [93]'s proposition to non-isometric reflections, viz. contractions and expansions, thereby allowing arbitrary simplex shapes. The Nelder-Mead algorithm has ever since become a *Science Citation Classic.*

Trust-Region Methods. The aforementioned approaches compute the objective function f values directly to identify promising search directions. In 1969, Winfield [99] suggested the use of a surrogate model of f, namely a quadratic interpolation model, to direct the search process within a region of validity. Along the same line, multiple approaches were proposed, to be later known as *trust-region* methods, denoting the region of validity: the neighborhood about the current solution, where the surrogate model is assumed to be valid (see, e.g., [100, 101, 102]).

Line-Search Methods. Furthermore, similar techniques were suggested to accelerate the search by constructing surrogate models to approximate the gradient. For instance, the implicit-filtering algorithm of Kelley et al. [26, 103] approximated the gradient at the current solution based on forward (or centered) differences. It has been shown that the approximated gradient is nothing but the *simplex gradient*: the gradient of the linear model fitting $n + 1$ sample points [104]. Some of these methods were also viewed as *line-search* algorithms, since they seek a better solution along a specific direction in the search space [105, 106, 107].

Partitioning Methods. In some optimization problems, decision variables take values constrained in a given range. Such problems, often referred to as *bound-constrained* optimization problems, play a key role in the design of general optimization algorithms, because many of which reduce their solutions to the solution of a sequence of bound-constrained problems. Moreover, bound-constrained optimization problems are present in several practical applications as the decision variables of many real-world systems are often bounded by physical limits [108, 109]. In the seventies of the twentieth century, a growing interest towards these problems was taking place. Piyavskii [54], Shubert [55], and Strongin [95, 96] were the pioneers in applying sequential partitioning for derivative-free optimization problems over a closed (compact) decision space.

Piyavskii [54] and Shubert [55] proposed, in 1972, to solve the problem of finding the global minimum of a function f defined on a closed interval $\subset \mathbb{R}$. Independently, they devised a sequential method—to be known later as Lipschitzian Optimization (`LO`)—of partitioning the space by iteratively refining a piecewise linear lower bound (of a sawtooth shape) on f based on the assumption that f's rate of change is upper-bounded by the Lipschitz constant $L > 0$. The next split (sample) takes place at the minimum of f's lower bound.

While Piyavskii and Shubert used a priori given constant L, Strongin [95, 96] proposed to adaptively estimate the Lipschitz constant during the search based on a statistical model, which also computes the likelihood of optimal solutions within each of the subintervals. It should be noted that pattern search methods have as well been extended to bound-constrained problems by Lewis and Torczon [110], where they proposed to include axis-wise directions in the set of poll directions.

From all of these early contributions, various algorithmic frameworks have emerged (see, e.g., [30, 37, 111, 112, 113, 114]) and there have been several classifications of these algorithms in the literature (see, e.g., [31, 87, 107]). Though with regard to the discussion herein, one can classify them into five groups: i). direct-search; ii). simplex-based; iii). trust-region; iv). line-search / implicit filtering; and v). partitioning methods.

Theoretical Analysis. With regard to convergence analysis, initial direct search methods lacked coherent mathematical analysis. Although key ingredients to prove convergence had already been established [115, 116], they fell out of favor with the mathematical community in the beginning of 1970s. According to Swann [117], *"... [direct search] methods have been developed heuristically, ... no proofs of convergence have been derived for them,... [and] sometimes the rate of convergence can be very slow."*

However, as gradient calculation had been the biggest cause of error in optimization software, engineering and scientific communities remained using direct search methods as a way of avoiding gradient calculation [37]. It was only in the 1990s that concrete mathematical analysis started to appear. The first convergence theory for a class of direct methods was provided by the PhD work of Torczon [118] and later works [119, 120] under the assumption of continuous derivatives. In 2003, Audet and Dennis Jr [98] relaxed this assumption by only requiring the objective function to be Lipschitz continuous and proved the optimality of a limit point for GPS methods using Clarke's calculus [121]. On the other hand, Tseng [122] proved that a modified Nelder-Mead algorithm converges to a stationary point provided that—among other conditions—the function is continuously differentiable. Convergence properties for several forms of convex objective functions were proved for simplex-search variants in [123]. Bortz and Kelley [124] built on ideas presented by Kelley [125] to analyze the global convergence of the implicit-filtering algorithm to a critical point.

The bulk of the convergence studies has sought to prove optimality of the limit (asymptotic) behavior of black-box MP methods. Few papers addressed the convergence rate (speed) of the same, which is more relevant in practical scenarios. Based on a theoretical study by Danilin [126] in 1971 on the number of iterations of Piyavskii [54]'s algorithm for univariate optimization problems, Hansen et al. [33] compared the number of iterations needed to have the minimum of the refined piecewise linear bound not more than $\varepsilon > 0$ lower than the minimum of the function f with the smallest possible number of iterations required by an artificial sequential algorithm whose bound achieves the same. Elster and Neumaier [127] proposed a low-dimensional trust-region algorithm that employs a quadratic regression model built by sampling successively refined grids and provided a hypothetical finite-time upper bound on the gradient as the algorithm refines its search. Theoretical analysis for trust-regions methods based on polynomial interpolation or regression have been presented in [31, 101]. Furthermore, the rate of local convergence for the implicit-filtering algorithm has been studied by Choi and Kelley [128]. In the recent years, the finite-time convergence has increasingly become the focus for several theoretical analyses [5, 34, 35].

The established convergence properties of black-box MP algorithms made them an attractive and popular choice for practitioners in scientific as well as engineering fields. Up to 2016 and according to Audet and Kokkolaras [129],

> *the design engineering community is increasingly becoming aware that rigorous black-box and derivative-free algorithms have made significant advances in the past 20 years and can be much better than heuristics.*

The latter are popular because they are easy to understand and implement, but their solutions can rarely be characterized in terms of optimality or quality. With only a modest investment in understanding the theoretical background and appropriate use, design engineers may reap large benefits by using black-box and derivative-free algorithms with convergence properties.

Multi-Objective Optimization. With regard to black-box multi-objective optimization,[2] very little/limited, yet slowly growing research has been reported on derivative-free approaches. Such algorithmic instances are mainly based on direct search methods, with two established extensions, viz. Multi-objective Mesh Adaptive Direct Search (`MULTIMADS`) [130] and Direct MultiSearch (`DMS`) [65]. In `MULTIMADS`, the problem is formulated as a series of single-objective problems that are solved using the Mesh Adaptive Direct Search (`MADS`) algorithm [30]. On the other hand, `DMS` carries out the search in the original problem formulation based on the partial order relationship among feasible solutions. In the limit, both of the algorithms converge to Pareto-Clarke critical points [65]. Recently, some papers started addressing the multi-objective problem from a partitioning, yet stochastic, scheme [131, 132, 133]. However, these algorithmic developments have been dominantly empirical.

As the focus of this thesis is on space-partitioning black-box MP (derivative-free) algorithms, in the next section, a detailed literature review is presented: categorizing key methods and briefly describing their variants.

2.3 Partitioning Algorithms for Continuous Black-Box Optimization

Partitioning algorithms, as discussed briefly in the past section, is one of the most important classes of black-box MP methods. The basic concept is that the decision (search) space $\mathcal{X}$ is broken recursively by splitting it into smaller and smaller parts. In essence, such algorithms implicitly build a tree over the decision space (as illustrated in Fig. 1.1) where each node corresponds to a subspace that can be further partitioned into smaller subspaces by expanding the corresponding node

[2]Multi-objective optimization will be discussed in detail in Section 2.4.2.

to two or more child nodes. The space-partitioning tree is used to look for the (or one) optimal solution over multiple scales (partitions) of the decision space.

As mentioned in Section 2.2, Piyavskii [54], Shubert [55], and Strongin [95, 96] are considered to be the pioneers in applying the framework of sequential partitioning to mathematical black-box optimization. Their work has become the basis, on which a large existing body of research relies (see, e.g., [32, 52, 134, 135, 136]). In general, partition algorithms make certain assumptions about the objective function smoothness (e.g., continuity, differentiability). In this thesis, these methods are grouped into two categories: one category requires the knowledge about the objective function smoothness, while the other category makes an assumption about the smoothness whose knowledge may not be available. In line with this classification, the next section presents and discusses several established algorithms from the two categories, namely Lipschitzian Optimization (`LO`) [54, 55] and Deterministic Optimistic Optimization (`DOO`) [34] from the first category; DIviding RECTangles (`DIRECT`) [56], Multilevel Coordinate Search (`MCS`) [27], and Simultaneous Optimistic Optimization (`SOO`) [34] from the second category. It should be noted that partitioning methods have been typically designed for single-objective problems. Towards the end of this section, a discussion on aspects of partition algorithms in terms of partitioning schemes is presented. This is followed by a formal background on multi-objective optimization and recent attempts to extend the space-partitioning algorithmic framework towards these problems.

2.3.1 Lipschitzian Optimization (`LO`)

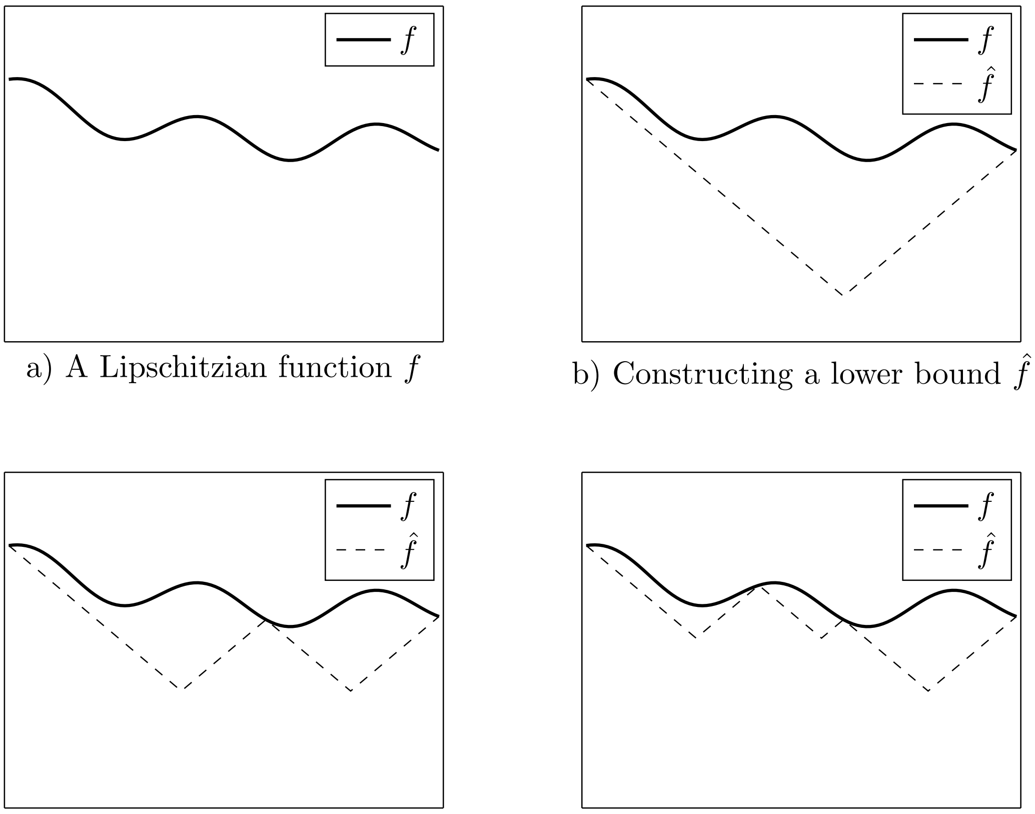
As discussed earlier, Piyavskii [54] and Shubert [55] independently proposed what was known later as Lipschitzian optimization (`LO`) for univariate problems. It has been extended to multi-dimensional problems in [137]. `LO` assumes that f satisfies the Lipschitz condition:

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L||\mathbf{x} - \mathbf{y}|| \ , \ \forall \mathbf{x}, \mathbf{y} \in \mathcal{X} \ , \tag{2.3}$$

where L, a positive real number, is the Lipschitz constant.[3] For a minimization problem, `LO` begins by evaluating the extreme points of the search space (e.g.,

[3]In this paper, we refer to the work of Piyavskii [54] and Shubert [55] by `LO`. One should note that Strongin [96] independently employed the Lipschitz condition for optimization problems. While Piyavskii and Shubert used a priori given constant L, Strongin proposed to adaptively estimate the Lipschitz constant during the search.

vertices of $\mathcal{X} = [\mathbf{l}, \mathbf{u}] \subset \mathbb{R}^n$ for (2.1)) and constructs a piecewise linear estimator $\hat{f}$ lower bounding f, by employing the Lipschitz condition (2.3). The minimum point of $\hat{f}$ represents the current estimator's best guess on where $\mathbf{x}^*$ lies. `LO` then uses this point as a partitioning point of the search space into multiple subspaces, on which the process of constructing a lower bound is applied iteratively (demonstrated in Fig. 2.2). In essence, `LO` gradually refines a piecewise linear estimator $\hat{f}$ to guide its hierarchical partitioning of the search space towards $\mathbf{x}^*$, where the next partitioning (sample) point is the optimum according to $\hat{f}$.



a) A Lipschitzian function f

b) Constructing a lower bound $\hat{f}$

c) $\hat{f}$ after one sample

d) $\hat{f}$ after two samples

Figure 2.2: Working principle of `Lipschitzian Optimization` in one-dimensional search space.

Limitations of `LO`. `LO` techniques come with several desired features such as the fact that L places a bound on how far the search is from the optimum and hence a more useful termination criterion can be employed rather than iteration/evaluation count. Second, a few parameters are required to be tuned when compared with other optimization algorithms. However, this approach has some drawbacks. First, the knowledge of L: although an estimate for the Lipschitz constant can be found (e.g. L may be set to $||\acute{f}||_\infty$ if f is differentiable [138, 139]), this is often infeasible for many applications as the objective function can have no closed-form expression [37, 140]. Since then, several methods have been proposed within the `LO` framework to approximate the Lipschitz constant [52]. The computational complexity of the method in higher dimensions is another drawback: with an n-dimensional search space, `LO` partitions the space into a set of hyperrectangles whose vertices are the sampled points. As a result, the number of function evaluations grows exponentially with n. Apart from the computational complexity of 2^n function evaluations, finding the next point to sample can be hard and involves solving several systems of linear equations [136, 141].

Variants of `LO`. Techniques from combinatorial optimization [142] can be employed in `LO` constructing lower and upper bounds for f. Such bounds are used to eliminate a portion of the search space $\mathcal{X}$. In other words, some leaf nodes of the space-partitioning tree are *never* considered for expansion and only a small part of the tree has to be generated and processed. This results in branch-and-bound search algorithms [52, 136, 143, 144]. `LO` techniques are only reliable if analytical knowledge about the function f is available, or an approximation of L is used (e.g., see [145]). For performance gains, `LO` can be parallelized as demonstrated in [146, 147, 148]. A general discussion on parallel algorithms for global optimization has been presented in [147].

2.3.2 DIviding RECTangles (`DIRECT`)

In 1993, Jones et al. [56], motivated by `LO`'s limitations, proposed a search technique, `DIRECT` (stands for DIviding RECTangles), which does not need the knowledge of L. Instead, it carries out the search by using all possible values of L from 0 to ∞ in a simultaneous framework. Furthermore, rather than splitting one subspace (or so-called hyperrectangle as the search space is normalized to be the unit box) at a time, `DIRECT` selects a set of subspaces/hyperrectangles—described as potentially-optimal—that are *convex-Pareto-optimal* in the 2-dimensional space of objective function values and hyperrectangle sizes, as illustrated in Fig. 2.3. The next definition puts formally the condition for hyperrectangles to be potentially-optimal.

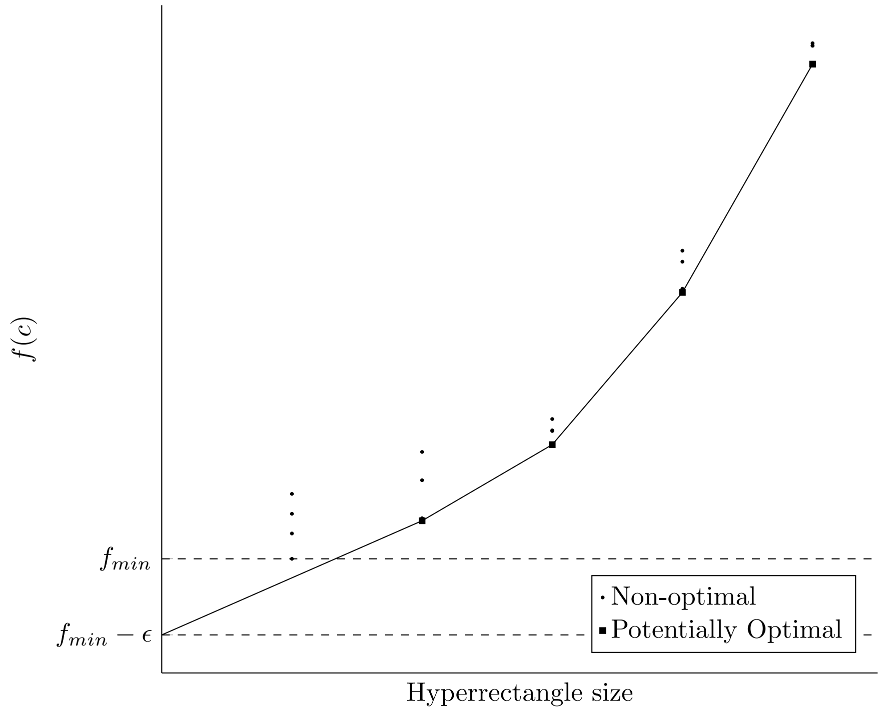


Figure 2.3: Identification of *potentially-optimal* hyperrectangles by `DIRECT`.

Definition 1. *(Potentially-optimal hyperrectangles) Denote the set of hyperrectangles created by* `DIRECT` *after k iterations by $\mathcal{H}$. Let $\epsilon > 0$ be a positive number, and let f_{min} be the best value of the objective function found so far. A hyperrectangle $i \in \mathcal{H}$, with center (base point) $\boldsymbol{c}_i$ and size σ_i, is said to be potentially optimal if there exists $\hat{L}$ such that,*

$$f(\boldsymbol{c}_i) - \hat{L}\sigma_i \leq f(\boldsymbol{c}_j) - \hat{L}\sigma_j \;, \forall j \in \mathcal{H} \tag{2.4}$$

$$f(\boldsymbol{c}_i) - \hat{L}\sigma_i \leq f_{min} - \epsilon |f_{min}| \;. \tag{2.5}$$

The procedure of `DIRECT` is outlined in Algorithm 1. It iteratively identifies and splits potentially-optimal hyperrectangles until the evaluation budget is exhausted. Initially, $\mathcal{H}_1$—the set of hyperrectangles at step 1—is initialized with a single subspace/ hyperrectangle that is the entire search space $\mathcal{X}$. At step t, the algorithm evaluates all the new hyperrectangles $i \in \mathcal{H}_t$ at their base points—i.e., centers $\mathbf{c}_i \in i$. It then identifies the set of potentially-optimal hyperrectangles $\mathcal{I}_t$ (Definition 1). Prior to step $t+1$, `DIRECT` partitions all the hyperrectangles in $\mathcal{I}_t$ according to a heuristic procedure (listed in Algorithm 2), which ensures well-shaped hyperrectangles and retains better solutions with bigger hyperrectangles to speed up local convergence, whilst maintaining an exploration-vs.-exploitation trade-off. After exhausting the evaluation budget v, `DIRECT` returns $\mathbf{c}(v)$, the hyperrectangle center with the best function value $f(\mathbf{c}(v))$ obtained.

`DIRECT` is far more efficient than `LO` as it performs sampling at the hyperrectangles' centers rather than their vertices. Moreover, `DIRECT` is consistent, i.e., it asymptotically converges to the global optimum ($\lim_{v\to\infty} r(v) = 0$) [56]. Originally, the algorithm was designed for decision spaces of simple bounds, i.e., $\mathcal{X}$ of Problem (2.1) is a hyperrectangle. Nevertheless, it can be easily extended to general-constraint settings (e.g., using artificial assignment heuristics) as shown in [139].

Algorithm 1 The DIviding RECTangles (`DIRECT`) algorithm by Jones et al. [56]

```
   Input: function to be minimized as a black-box $f$, search space $\mathcal{X}$, evaluation budget $v$
   Initialization: $\mathcal{H}_1 = \{\mathcal{X}\}$
   Output: Approximation of $f$'s minimizer $\mathbf{x}(v)$
1: while evaluation budget $v$ is not exhausted do
2:     Evaluate all the new hyperrectangles $\in \mathcal{H}_t$.
3:     $\mathcal{I}_t \leftarrow$ the set of hyperrectangles $\in \mathcal{H}_t$ that are potentially optimal (Definition 1).
4:     Partition the hyperrectangles in $\mathcal{I}_t$ according to the procedure outlined in Algorithm 2.
5:     $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \setminus \mathcal{I}_t \cup \{\mathcal{I}_t$'s newly generated hyperrectangles$\}$.
6:     $t \leftarrow t + 1$.
7: end while
8: return $\arg\min_{\mathbf{c}_i : i \in \mathcal{H}_t} f(\mathbf{c}_i)$
```

Algorithm 2 Partitioning Procedure of `DIRECT` [56]

```
   Input: function $f$, potentially-optimal hyperrectangle $i \in \mathcal{I}_t$,
1: Identify the set $J$ of the $n$ dimensions with the maximum side length of hyperrectangle $i$. Let $\delta$ equal one-third of $i$'s maximum side length.
2: Sample the function at the points $\mathbf{c}_i \pm \delta \cdot \mathbf{e}_j$ for all $j \in J$, where $\mathbf{c}_i$ is the center of $i$ and $\mathbf{e}_j$ is the $j$th unit vector.
3: Divide the hyperrectangle $i$ containing $\mathbf{c}$ into thirds along the dimensions in $J$, starting with the dimension with the lowest value of
       $$w_j = \min(f(\mathbf{c}_i + \delta \cdot \mathbf{e}_j), f(\mathbf{c}_i - \delta \cdot \mathbf{e}_j)) , \qquad (2.6)$$
   and continuing to the dimension with the highest $w_j$.
```

Variants of `DIRECT`. Over the past years, several studies and modifications have been conducted on `DIRECT` (see, e.g., [36, 139, 149, 150, 151, 152, 153, 154, 155, 156]). While `DIRECT` represented the size of a hyperrectangle $i \in \mathcal{H}$ (denoted by σ_i) by the l^2-norm of the distance from the center to the vertices (diagonal), Gablonsky and Kelley [150] used l^∞-norm, thereby reducing the variations in the hyperrectangle sizes; and making the algorithm locally biased. Numerical experiments showed that it is good for problems with a few global minima as compared to the original `DIRECT` algorithm. On the other hand, it has been shown in [36] that the ϵ-constraint (2.5) is sensitive to additive scaling and leads to a slow asymptotic convergence. Hence, in [36], a modification has been proposed by using the threshold $\epsilon|f_{median} - f_{min}|$ instead of $\epsilon|f_{min}|$ in Eq. (2.5), where f_{median} is the median of the observed function values. Sergeyev and Kvasov [151] coupled `DIRECT`'s idea of using several values of the Lipschitz constant L instead of a unique value with estimating a bound on f and proposed a two-phased algorithm that is suitable for multi-modal functions. The first phase focuses on moving closer towards discovered local optima by splitting hyperrectangles close to the current best solution. On the other hand, the second phase is oriented towards discovering new local optima by expanding hyperrectangles with high global scores and far from the current best solution.

2.3.3 Multilevel Coordinate Search (`MCS`)

Inspired by `DIRECT`, Huyer and Neumaier [27] proposed what can be regarded as a *branch without bound* algorithm to look for the optimal solution over a box $\mathcal{X} = [\mathbf{l}, \mathbf{u}]$ with infinite or finite bounds. Similar to `DIRECT`, the so-called Multilevel Coordinate Search (`MCS`) algorithm partitions $\mathcal{X}$ in hyperrectangles/boxes. However, `MCS` allows these boxes to be of uneven sizes. The base points can be anywhere in the corresponding boxes, which rectifies the slow-convergence shortcoming of `DIRECT` in optimizing functions whose optimizers lie at the boundary of the decision space. Furthermore, the boxes are split one coordinate at a time, in contrast to `DIRECT`, which splits a box along several coordinates. In addition to its branching framework, `MCS` contains a local search enhancement using a quadratic interpolation model.

Algorithm 3 The Multilevel Coordinate Search (`MCS`) algorithm by Huyer and Neumaier [27]. Adapted from [157]

```
    Input: search space $\mathcal{X} = [\mathbf{l},\mathbf{u}]$, function to be minimized as a black-box $f$, $\mathbf{x}^0 \in [\mathbf{l},\mathbf{u}]$, initialization list $x_i^j (j = 1,\dots,L_i, i = 1,\dots,n)$, $s_{max}$, evaluation budget $v$
 1: $\mathbf{x} \leftarrow \mathbf{x}^0$; b $\leftarrow [\mathbf{l},\mathbf{u}]$; $level(b) \leftarrow 1$
 2: for $i \leftarrow 1$ to $n$ do                                              ▷ initialization
 3:     $\mathbf{x} \leftarrow$ the best of $\{\mathbf{x}^j\}_{j=1}^{L_i}$, where $\mathbf{x}^j$ is $\mathbf{x}$ with $x_i$ changed to $x_i^j$.
 4:     Split $b$ along the $i$th coordinate at $x_i^j (j = 1,\dots,L_i)$ and between.
 5:     $b \leftarrow$ the largest box containing $\mathbf{x}$.
 6: end for
 7: while there are no boxes of level $s < s_{max}$ and
 8:         evaluation budget $v$ is not exhausted do
 9:     for all non-empty levels $s \leftarrow 2$ to $s_{max} - 1$ do                  ▷ branching
10:         Choose the box $b$ at the level $s$ with the lowest function value.
11:         $i \leftarrow$ the coordinate used least often when producing $b$.
12:         if $s > 2n(n_i + 1)$ then                                     ▷ split by rank
13:             Split the box $b$ along the $i$th coordinate.
14:         else if box not tagged as not promising then               ▷ split by exp. gain
15:             Determine the most promising splitting coordinate $i$.
16:             Compute the minimal expected function value $f_{exp}$ at new point(s).
17:             if $f_{exp} < f_{best}$ then
18:                 Split $b$ along the $i$th coordinate.
19:             else
20:                 Tag $b$ as not promising, increase its level by 1.
21:             end if
22:         end if
23:     end for
24:     for base points $\mathbf{x}$ of all the new boxes at level $s_{max}$ do            ▷ local search
25:         Start a local search from $\mathbf{x}$ if improvement is expected.
26:     end for
27: end while
28: return the(or one) best of all the $v$ sampled points $\mathbf{x}(v)$, i.e., $f_{best} = f(\mathbf{x}(v))$.
```

From Algorithm 3, one can observe that the procedure of `MCS` has three main phases: initialization, branching, and local search phase. First, along each dimension (coordinate) $i = 1,\dots,n$, three or more user-defined values $x_i^1 < x_i^2 < \cdots < x_i^{L_i}$ in $[l_i, u_i]$ are specified, where $L_i \geq 3$. Based on these values and an initial point $\mathbf{x}^0$, the algorithm then performs n splits, where a new best point $\mathbf{x}$ is computed and the partitioned box at step i is the box that contains $\mathbf{x}$. The number of boxes formed at each step is at least $2(L_i - 1) \geq 4$ boxes since the splits take place at and between the user-defined values.

After initialization is complete, MCS starts the core of its search: *branching* through partitions (boxes) over the decision space. Each box is assigned a level s, which indicates the number of times the corresponding box has been processed. As outlined in Algorithm 3 (lines 9–23), MCS sweeps through the assigned levels and nominates the box with the least objective value at each level value for splitting. Let n_i be the number of splits along the least often used coordinate i in the history of the candidate box. The algorithm employs two heuristic rules to decide if a nominated box is to be partitioned, namely i). split by rank: split the box if $s > 2n(n_i + 1)$; else ii). split by expected gain: split the box along a direction that minimizes a local separable quadratic model if the expected best function value improves by some additive factor ($\min_{1\leq i\leq n} \hat{e}_i$, where $\hat{e}_i$ is the computed expected gain along coordinate i) over the best achieved function value; otherwise, the box's level s is increased by one and the procedure continues. Base points at level s_{max} are put into a *shopping basket*, assuming them to be useful points. One can accelerate the convergence by starting local searches from these points and checking that they do not converge to points already in the shopping basket before putting them there.

With local search, MCS seeks to accelerate the convergence to optimal points from the base points of boxes of level s_{max}. This is achieved by constructing a local quadratic interpolation surrogate model by triple searches, followed by a line search along the direction that minimizes the surrogate model until a stopping criterion is met, i.e., no progress is observed or the approximated gradient is arbitrarily small.

With regard to its theoretical properties, MCS is consistent, that is asymptotically converges if and only if s_{max} goes to infinity [27].

2.3.4 Optimistic Optimization

Although the class of Lipschitz functions is very general [52], it is still a strong restrictive assumption on a wide class of functions. In an attempt to handle more general assumptions, Munos [34] designed two algorithms, viz. DOO (stands for Deterministic Optimistic Optimization) and SOO (stands for Simultaneous Optimistic Optimization) that follow the *optimism in the face of uncertainty* principle, whose foundation is in the field of machine learning (e.g., [46]). Munos proposed DOO by assuming f to be locally smooth (around one of its global optima $\mathbf{x}^*$) with respect to a semi-metric ℓ. That is to say,

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \ell(\mathbf{x}, \mathbf{x}^*) \ , \ \forall \mathbf{x} \in \mathcal{X} \ . \tag{2.7}$$

This relaxes the restrictive assumption of LO. Based on (2.7), DOO estimates the minimum lower bound of f within a subspace/hyperrectangle, which is referred to as the hyperrectangle's b-value. Similar to LO, this lower bound is used to guide the hierarchical partitioning of the decision space.

As outlined in Algorithm 4, DOO can be regarded as a divide-and-conquer search algorithm that iteratively constructs finer and finer partitions of the search space $\mathcal{X}$ at multiple scales $h \geq 0$ in looking for the optimum solution. Given a scale $h \geq 0$ and a partition factor $K \geq 2$, $\mathcal{X}$ can be partitioned into a set of K^h cells/hyperrectangle $\mathcal{X}_{h,i}$ such that $\cup_{i\in\{0,\ldots,k^h-1\}}\mathcal{X}_{h,i} = \mathcal{X}$. These cells can be represented by nodes of a K-ary tree $\mathcal{T}$. A node (h, i) represents the cell $\mathcal{X}_{h,i}$. The set of leaves in $\mathcal{T}$ is denoted as $\mathcal{L} \subseteq \mathcal{T}$. Similar to DIRECT and MCS, each node is associated with a base point $\mathbf{x}_{h,i} \in \mathcal{X}_{h,i}$ at which f is evaluated. In each iteration, one leaf node (h, i) is expanded if its b-value is the (or one of the) smallest value(s) among $\mathcal{L}$. The b-value for (h, i) is computed as $f(\mathbf{x}_{h,i}) - \min_{\mathbf{x}\in\mathcal{X}_{h,i}} \ell(\mathbf{x}, \mathbf{x}_{h,i})$ based on (2.7).

Algorithm 4 The Deterministic Optimistic Optimization (DOO) by Munos [34]

```
   Input: function to be minimized as a black-box $f$, search space $\mathcal{X}$, budget
   evaluation $v$, partition factor $K$
   Initialization: $t \leftarrow 1,\ \mathcal{T}_1 = \{(0,0)\}$
   Output: approximation of $f$'s minimizer
1: while the evaluation budget is not exhausted do
2:     Evaluate all the nodes $(h, i) \in \mathcal{L}_t$.
3:     Expand the node $(h_*, i_*) = \arg\min_{h,i:(h,i)\in\mathcal{L}_t} f(\mathbf{x}_{h,i}) - \min_{\mathbf{x}\in\mathcal{X}_{h,i}} \ell(\mathbf{x}, \mathbf{x}_{h,i})$.
4:     $\mathcal{T}_{t+1} \leftarrow \mathcal{T}_t \cup \{(h_*, i_*)$'s $K$ children$\}$.
5:     $t \leftarrow t + 1$.
6: end while
7: return $\mathbf{x}(v) = \arg\min_{\mathbf{x}_{h,i}:(h,i)\in\mathcal{T}_t} f(\mathbf{x}_{h,i})$
```

Similar to LO, DOO requires the knowledge of the function smoothness. This makes DOO practically inapplicable, which motivated the design of the Simultaneous Optimistic Optimization (SOO) algorithm. Munos [34] sought to approximate DOO's behavior when ℓ is unknown. It expands simultaneously all the nodes (h, i) of its

Algorithm 5 The Simultaneous Optimistic Optimization (SOO) by Munos [34]

```
    Input: function to be minimized as a black-box $f$, search space $\mathcal{X}$, budget evaluation $v$, partition factor $K$
    Initialization: $t \leftarrow 1$, $\mathcal{T}_1 = \{(0,0)\}$ , Evaluate $f(\mathbf{x}_{0,0})$
    Output: approximation of $f$'s minimizer
 1: while evaluation budget is not exhausted do
 2:     $\nu_{\min} \leftarrow \infty$
 3:     for $h = 0 : \min\{\text{depth}(\mathcal{T}_t), h_{\max}(t)\}$ do
 4:         Select $(h, j) = \arg\min_{j \in \{j|(h,j) \in L_t\}} f(\mathbf{x}_{h,j})$
 5:         if $f(\mathbf{x}_{h,j}) < \nu_{\min}$ then
 6:             Evaluate the children of $(h, j)$
 7:             Add the children of $(h, j)$ to $\mathcal{T}_t$
 8:             $\nu_{\min} \leftarrow f(\mathbf{x}_{h,j})$
 9:             $t \leftarrow t + 1$
10:         end if
11:     end for
12: end while
13: return $\mathbf{x}(v) = \arg\min_{x_{h,i}:(h,i) \in \mathcal{T}_t} f(\mathbf{x}_{h,i})$
```

tree $\mathcal{T}$ for which there exists a semi-metric ℓ such that the corresponding lower bound would be the minimum. This is simulated by expanding at most a leaf node per depth if such node has the least $f(\mathbf{x}_{h,i})$ with respect to leaf nodes of the same or lower depths as illustrated in Algorithm 5. In addition to that, the algorithm takes a function $h_{max}(t)$, as a parameter, such that after t node expansions only nodes at depth $h \leq h_{max}(t)$ can be expanded.

Both the algorithms enjoy sound finite-time properties as the regret (2.2) decreases as a function of the number of function evaluations v in $O(v^{-1/d})$, where d is the near-optimality dimension of f defined similar to optimality measures in [158, 159]. An exponential decreasing loss can as well be achieved when $d = 0$—i.e., $r(v) = O(e^{-cv})$, where $c > 0$.

Variants of SOO. Two main variants of SOO have been proposed in the literature, namely, the Stochastic Simultaneous Optimistic Optimization (StoSOO) [160] and Bayesian Multi-Scale Optimization (BaMSOO) [5] algorithms. StoSOO addresses the situation where function evaluations are perturbed independently by noise. The b-values of StoSOO's nodes are lower confidence bounds of f at their representative/base points. As a stochastic extension of SOO, it expands at most one node per depth after having it evaluated at its base point several times. Multiple evaluations per node are to ensure that the expanded nodes are with high probability close to optimal. Similar to StoSOO's stochastic settings is the Hierarchical Optimistic Optimization (HOO) by Bubeck et al. [161], which builds a binary tree to estimate the mean of the stochastic f over $\mathcal{X}$. On the other hand, BaMSOO was designed with the goal of cutting down the number of function evaluations incurred by SOO. BaMSOO eliminates the need for evaluating representative states $\mathbf{x}_{h,i}$ deemed unfit by Gaussian Process (GP) posterior bounds. Prior to evaluating a node (h, i), upper and lower confidence bounds on $f(\mathbf{x}_{h,i})$ are computed using a GP posterior fitted on the previous evaluations. These bounds are employed to guide the search efficiently and possibly serve as the b-value of the corresponding node instead of $f(\mathbf{x}_{h,i})$.

For more details on black-box MP algorithms, one can refer to some survey papers on the topic, viz. the ones by Kolda et al. [37], and Lewis et al. [162]. The first textbooks dedicated to this topic were written by Conn et al. [104], and Kelley [163].

2.4 Discussion

In this section, different aspects of partition methods for black-box optimization are discussed, followed by a formal background on multi-objective optimization problems and methods tailored for such problems.

2.4.1 Rules of Partitioning

Partition algorithms use different procedures for splitting a region of the decision space. On one end, DOO and SOO partition a cell/hyperrectangle by a single coordinate creating K cells of equal length along that coordinate. On the other end, LO partitions with respect to all the n coordinates, which results in 2^n (not necessarily equal in size) cells. For MCS, knowledge built from sampled points is used to determine the partitioning coordinate as well as the position of the partitioning through the rules of golden section ratio and expected gain [27]. DIRECT stands out in two aspects: i). its partitioning may go by a single coordinate up to n coordinates in a way that retains better solutions with bigger regions; ii). these partitions are applied recursively to the hyperrectangle itself and some of its descendants. In other words, if DIRECT is expanding a region (h, i) by n coordinates, the first split takes place at that region, and the kth split ($k \leq n$) is applied on a subregion created from the past $m-1$ splits as described in Algorithm 2.

A number of rules for splitting the search space has been investigated and numerically validated [164, 165, 166]. For instance, numerical experiments in [166] showed that bisecting a subspace into two subcells works better than partitioning it into 2^n subcells using n intersecting hyperplanes.

2.4.2 Multi-Objective Optimization

Many real-world application and decision problems involve optimizing two or more objectives at the same time [see, for instance, 10, 42]. In the general case, Multi-objective Optimization Problems (MOPs) are hard because the objective functions are often conflictual, and it is difficult to design strategies that are optimal for all objectives simultaneously. Furthermore, with conflicting objectives, there does not exist a single optimal solution but a set of incomparable optimal solutions: each is inferior to the other in some objectives and superior in other objectives. This induces a partial order on the set of feasible solutions to an MOP. The set of optimal feasible solutions according to this partial order is referred to as the *Pareto optimal set* and its corresponding image in the objective space is commonly named as the *Pareto front* of the problem. The task of multi-objective algorithms therefore becomes finding the Pareto front or producing a good approximation of it (referred to as an *approximation set* of the problem).

Formal Background. Without loss of generality, the multi-objective minimization problem with n decision variables and m objectives, has the form:

$$\begin{aligned} \text{minimize} \quad & \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ \text{where} \quad & \mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X} \ , \\ & \mathbf{y} = (y_1, \ldots, y_m) \in \mathcal{Y} \ , \end{aligned} \tag{2.8}$$

given v function evaluations. The vector $\mathbf{x}$ is called the *decision vector (solution)*, $\mathbf{y}$ is called the *objective vector*,[4] $\mathcal{X}$ is the *feasible decision space*, and $\mathcal{Y} = \prod_{1\leq j\leq m} \mathcal{Y}_j$ is the corresponding *objective space*, where $\mathcal{Y}_j$ is the jth-objective space and we write the corresponding image in the objective space for any region $\hat{\mathcal{X}} \subseteq \mathcal{X}$ as $\mathbf{f}(\hat{\mathcal{X}}) \subseteq \mathcal{Y}$. It is assumed that the derivatives of the functions involved are neither

[4]For brevity, we sometimes omit the word *objective* when referring to an objective vector.

symbolically nor numerically available. Nevertheless, $\mathbf{f}$ can be evaluated point-wise, which is typically an expensive operation, requiring some computational resources (e.g., time, power, money). More specifically, the task is to best approximately solve Problem (2.8) using a computational budget of v function evaluations.

A vector $\mathbf{y}^1$ is more preferable than another vector $\mathbf{y}^2$, if $\mathbf{y}^1$ is at least as good as $\mathbf{y}^2$ in all objectives *and* better with respect to at least one objective. $\mathbf{y}^1$ is then said to be *dominating* $\mathbf{y}^2$. This notion of dominance is commonly known as *Pareto dominance*, which leads to a *partial order* on the objective space, where we can define a Pareto optimal vector to be one that is non-dominated by any other vector in $\mathcal{Y}$. Nevertheless, $\mathbf{y}^1$ and $\mathbf{y}^2$ may be incomparable to each other, because each is inferior to the other in some objectives and superior in other objectives. Hence, there can be several Pareto optimal vectors. The following definitions put these concepts formally (see, for more details, [44, 167]).

Definition 2 (Pareto dominance)**.** *The vector* $\mathbf{y}^1$ *dominates the vector* $\mathbf{y}^2$*, that is to say,* $\mathbf{y}^1 \prec \mathbf{y}^2 \iff y^1_j \leq y^2_j$ *for all* $j \in \{1, \dots, m\}$ *and* $y^1_k < y^2_k$ *for at least one* $k \in \{1, \dots, m\}$.

Definition 3 (Strict Pareto dominance)**.** *The vector* $\mathbf{y}^1$ *strictly dominates the vector* $\mathbf{y}^2$ *if* $\mathbf{y}^1$ *is better than* $\mathbf{y}^2$ *in all the objectives, that is to say,* $\mathbf{y}^1 \prec\prec \mathbf{y}^2 \iff y^1_j < y^2_j$ *for all* $j \in \{1, \dots, m\}$.

Definition 4 (Weak Pareto dominance)**.** *The vector* $\mathbf{y}^1$ *weakly dominates the vector* $\mathbf{y}^2$ *if* $\mathbf{y}^1$ *is not worse than* $\mathbf{y}^2$ *in all the objectives, that is to say,* $\mathbf{y}^1 \preceq \mathbf{y}^2 \iff y^1_j \leq y^2_j$ *for all* $j \in \{1, \dots, m\}$.

Definition 5 (Pareto optimality of vectors)**.** *Let* $\hat{\mathbf{y}} \in \mathcal{Y}$ *be a vector.* $\hat{\mathbf{y}}$ *is Pareto optimal* $\iff \nexists \mathbf{y} \in \mathcal{Y}$ *such that* $\mathbf{y} \prec \hat{\mathbf{y}}$*. The set of all Pareto optimal vectors is referred to as the Pareto front and denoted as* $\mathcal{Y}^*$*. The corresponding decision vectors (solutions) are referred to as the Pareto optimal solutions or the Pareto set and denoted by* $\mathcal{X}^*$*.*

In other words, the solution to the MOP (2.8) is its Pareto optimal solutions (Pareto front in the objective space). Practically, multi-objective solvers aim to identify a set of objective vectors that represent the Pareto front (or a good approximation of it). We refer to this set as the approximation set.

Definition 6 (Approximation set)**.** *Let $A \subseteq \mathcal{Y}$ be a set of objective vectors. A is called an approximation set if any element of A does not dominate or is not equal to any other objective vector in A. The set of all approximation sets is denoted as Ω. Note that $\mathcal{Y}^* \in \Omega$.*

Furthermore, denote the *ideal point (vector)*—not necessarily reachable—by

$$\mathbf{y}^* \overset{\text{def}}{=} (\min_{\mathbf{y}\in\mathcal{Y}^*} y_1, \ldots, \min_{\mathbf{y}\in\mathcal{Y}^*} y_m) \,. \tag{2.9}$$

Likewise, let us denote the (or one of the) global optimizer(s) of the jth objective function by $\mathbf{x}_j^*$, i.e., $y_j^* = f_j(\mathbf{x}_j^*)$. Note that $\mathbf{x}_j^* \in \mathcal{X}^*$. On the other hand, we define the *nadir point* of a region in the objective space $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$ as

$$\mathbf{y}^{nadir}(\hat{\mathcal{Y}}) \overset{\text{def}}{=} (\max_{\mathbf{y}\in\hat{\mathcal{Y}}} y_1, \ldots, \max_{\mathbf{y}\in\hat{\mathcal{Y}}} y_m) \,. \tag{2.10}$$

Performance Assessment for Multi-Objective Optimization. Given two approximation sets $A, B \in \Omega$, it is not that easy to tell which set is better, particularly if their elements are incomparable [168]. In general, two aspects are considered in an approximation set: i). its distance (the closer the better) to the optimal Pareto front and ii). its diversity (the higher the better) within the optimal Pareto front. To this end, several *quality indicators* have been proposed [169]. The quality of an approximation set is measured by a so-called (unary) quality indicator $I : \Omega \to \mathbb{R}$, assessing a specific property of the approximation set. Likewise, an l-ary quality indicator $I : \Omega^l \to \mathbb{R}$ quantifies quality differences between l approximation sets [65, 168]. A quality indicator is not Pareto-compliant if it contradicts the order induced by the Pareto-dominance relations. Four commonly-used quality indicators, whose properties are listed in Table 2.2, are formally defined next.

Definition 7. *(Additive ϵ-indicator [168]) For any two approximation sets $A, B \in \Omega$, the additive ϵ-indicator $I_{\epsilon+}$ is defined as:*

$$I_{\epsilon+}(A, B) = \inf_{\epsilon\in\mathbb{R}}\{\forall \boldsymbol{y}^2 \in B,\ \exists \boldsymbol{y}^1 \in A : \boldsymbol{y}^1 \preceq_{\epsilon+} \boldsymbol{y}^2\} \tag{2.11}$$

where $\boldsymbol{y}^1 \preceq_{\epsilon+} \boldsymbol{y}^2 \iff y_j^1 \leq \epsilon + y_j^2$ for all $j \in \{1, \ldots, m\}$. If B is the Pareto front $\mathcal{Y}^*$ (or a good approximation reference set $R \in \Omega$ if $\mathcal{Y}^*$ is unknown) then $I_{\epsilon+}(A, B)$ is referred to as the unary additive epsilon indicator and is denoted by $I^1_{\epsilon+}(A)$, i.e., $I^1_{\epsilon+}(A) \overset{\text{def}}{=} I_{\epsilon+}(A, \mathcal{Y}^*)$.

Quality Indicator (I)	Pareto-Compliant	Reference Set Required	Target
Additive ϵ-Indicator ($I^1_{\epsilon+}$)	Yes	Yes	Minimize
Hypervolume Indicator (I^-_H)	Yes	Yes	Minimize
Generational Distance (I_{GD})	No	Yes	Minimize
Inverted Generational Distance (I_{IGD})	No	Yes	Minimize

Table 2.2: Commonly-used quality indicators [for more details, see 2, 3, 4].

Definition 8. *(Hypervolume Difference [169]) For any two approximation sets $A, B \in \Omega$, the hypervolume difference I_H is defined as:*

$$I_H(A, B) = I_H(A) - I_H(B) \tag{2.12}$$

where $I_H(A)$ measures the hypervolume of the objective space portion that is weakly dominated by A [170] with respect to a reference point. If A is the Pareto front $\mathcal{Y}^*$ (or a good approximation reference set $R \in \Omega$ if $\mathcal{Y}^*$ is unknown) then $I_H(A, B)$ is referred to as the hypervolume indicator and is denoted by $I^-_H(B)$, i.e., $I^-_H(B) \overset{\text{def}}{=} I_H(\mathcal{Y}^*, B)$.

Definition 9. *(Generational Distance [171]) For any approximation sets $A \in \Omega$, the generational distance metric I_{GD} is defined as:*

$$I_{GD}(A) = \frac{1}{|A|} \sum_{\boldsymbol{a} \in A} \min_{\boldsymbol{b} \in \mathcal{Y}^*} ||\boldsymbol{a} - \boldsymbol{b}|| \ . \tag{2.13}$$

If the Pareto front $\mathcal{Y}^*$ is unknown, a good approximation reference set $R \in \Omega$ can be used.

Definition 10. *(Inverted Generational Distance [172]) For any approximation sets $A \in \Omega$, the inverted generational distance metric I_{IGD} is defined as:*

$$I_{IGD}(A) = \frac{1}{|\mathcal{Y}^*|} \sum_{\boldsymbol{b} \in \mathcal{Y}^*} \min_{\boldsymbol{a} \in A} ||\boldsymbol{a} - \boldsymbol{b}|| \ . \tag{2.14}$$

If the Pareto front $\mathcal{Y}^*$ is unknown, a good approximation reference set $R \in \Omega$ can be used.

In accordance with the single-objective loss (2.2) measure for optimization methods, we introduce a vectorial loss measure for MOPs. Let $\mathcal{Y}^v_* \in \Omega$ be the approximation set returned by an algorithm after v function evaluations, we have:

$$\mathbf{r}(v) = \mathbf{y}^v_* - \mathbf{y}^* \tag{2.15}$$

where $\mathbf{y}^v_*$ is the empirical ideal point found so far, defined as $(\min_{\mathbf{y} \in \mathcal{Y}^v_*} y_1, \ldots, \min_{\mathbf{y} \in \mathcal{Y}^v_*} y_m)$.

Methods for Multi-Objective Optimization. Conventionally, solving an MOP follows one of two principles, namely *preference-based* and *ideal* principles [173]. Following the preference-based principle, the MOP is transformed into a single-objective optimization problem (through an aggregation/scalarization function that exploits a priori information), which then can be solved using one of many available single-objective optimizers [see, for instance, 174, 175, 176]. While preference-based algorithms converge to a single solution in each run, ideal-based algorithms search for a set of solutions at once. One example in this approach is evolutionary multi-objective algorithms [177, 178] in which a population of solutions evolves, following a crude analogy with Darwinian evolution, towards better solutions. Recently, there has been a growing interest in formulating multi-objective problems within the framework of reinforcement learning [see, for instance, 179, 180, 181, 182, 183].

Partitioning Algorithms for Multi-Objective Optimization. As discussed earlier, two main algorithmic approaches have been designed by the Mathematical Programming (MP) community to deal with multi-objective optimization represented by `MULTIMADS` [130] and `DMS` [65], respectively. These approaches belong to the class of direct-search methods. With regard to decision space partitioning methods, little work has been done on multiple objectives. In 2008, Wang et al. [131] adapted the `DIRECT` [56] algorithm to identify potentially optimal solutions based on their ranks and crowding distance. The identified solutions are then used as candidates for a multi-objective genetic algorithm. On the other hand, Van Moffaert et al. [133] empirically validated an extension of the stochastic Hierarchical Optimistic Optimization (`HOO`) [161] on the problem of filling phase for wet clutches.

2.5 Summary

In this section, a summary of the research challenges/issues with regard to partitioning algorithms is discussed based on the literature review presented in this chapter.

Finite-Time Analysis. On the one hand, the work of Torczon [118, 120] and Audet and Dennis Jr [98] has consolidated a theoretical framework to analyze the asymptotic convergence of a class of black-box MP methods, i.e., direct-search algorithms (particularly, GPS methods). On the other hand, the theoretical analysis of space-partitioning MP algorithms such as `LO` and `DIRECT` has been sparse and independent (see, e.g., [29, 33]). One is therefore motivated to address this theoretical gap, by providing a generic theoretical framework. Furthermore, as optimization algorithms are employed as anytime solvers in practice, finite-time analysis should be the focus of such a generic framework rather than asymptotic analysis, as will be seen in Chapter 3.

Expensive Optimization. Although black-box MP algorithms are designed such that a systematic and consistent behavior is achieved from the beginning of the search, one can observe from the literature that plenty of these algorithms exhibit slow convergence [37]. Among the contributing factors is that global search (exploration) is prioritized over local search (exploitation). For instance, Finkel and Kelley [36] had showed that `DIRECT` may act as an exhaustive grid search. On the other hand, Huyer and Neumaier [27] complemented `MCS`'s branching procedure with an independent local search. In other words, global search is performed before local search. With the widespread emergence of expensive black-box optimization, one is motivated to integrate the local (exploitative) search into the partitioning/branching procedure itself rather than having it as a separate component. This issue is addressed in Chapter 4.

Multi-Objective Optimization. While the *divide-and-conquer* paradigm has inspired several successful evolutionary multi-objective techniques, such as the Multi-objective Evolutionary Algorithm Based on Decomposition (`MOEA/D`) framework [184] that decomposes an MOP into a number of different single objective optimization subproblems (or simple multi-objective optimization subproblems). It would be interesting to take the same inspiration from a different angle, viz. *mathematical programming* whose theoretical foundations and structured sampling may help towards a better understanding of MOPs. Chapter 5 examines this topic theoretically, followed by an empirical validation in Chapter 6.

Chapter 3

Single-Objective Multi-Scale Search

"If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice"

- Donald Knuth

In this chapter, a family of Mathematical Programming (MP) algorithms for continuous black-box optimization is addressed. These algorithms employ a tree search, which partitions the objective function domain over multiple scales in search for the (or one) global optimum. For such algorithms, we provide a generic framework and refer to it as the Multi-scale Search Optimization (MSO) framework. Furthermore, a theoretical methodology is presented to analyze the finite-time performance of MSO algorithms based on three basic assumptions: i). local Hölder continuity of the objective function f; ii). partitions boundedness; and iii). partitions sphericity. Such methodology quantifies the amount of exploration carried by an MSO algorithm over a partition of a specific scale. As a result, a bound on the regret (2.2) can be established reflecting the finite-time behavior of the algorithm. Using these theoretical findings, a finite-time analysis is established for Lipschitzian optimization methods [55], DIviding RECTangles (`DIRECT`) [56], and Multilevel Coordinate Search (`MCS`) [27]. Moreover, we build on and integrate the analysis of Deterministic Optimistic Optimization (`DOO`) and Simultaneous Optimistic Optimization (`SOO`) in [34] under the MSO framework. Without loss of generality, the work presented in this chapter is formulated with respect to a ***maximization*** version of Problem (2.1).

This chapter is organized as follows. Basic notations and terminology are introduced in Section 3.1. In Section 3.2, a formal introduction of the MSO framework to solve continuous black-box optimization problems is provided. Towards the end of the section, some well-known black-box MP algorithms are discussed from the presented framework's viewpoint. In Section 3.3, a principled approach for analyzing MSO algorithms theoretically is presented and demonstrated on the algorithms discussed in Section 3.2.2. Section 3.4 outlines the theoretical contribution and complements it with empirical validation. Towards the end, Section 3.5 summarizes this chapter.

3.1 Some Preliminaries

In this section, using some of the concepts from [185], we formally define the hierarchy of partitions and the data structure (tree), on the search space $\mathcal{X}$ employed by an MSO algorithm.

We denote by $2^{\mathcal{X}}$ the set of all subsets (subspaces, or cells) of $\mathcal{X}$. The size/volume of a subset in $2^{\mathcal{X}}$ is approximated by the function $\sigma : 2^{\mathcal{X}} \rightarrow \mathbb{R}^+$. Two elements $\mathcal{X}_i$, $\mathcal{X}_j$ of $2^{\mathcal{X}}$ are said to be *disjoint* if and only if

$$\mathcal{X}_i \cap \mathcal{X}_j = \beta(\mathcal{X}_i) \cap \beta(\mathcal{X}_j) \tag{3.1}$$

Here, $\beta(\mathcal{X}_i)$ denotes the boundary of $\mathcal{X}_i$. A subset of $2^{\mathcal{X}}$ is called a *partial partition* of $\mathcal{X}$ if its elements are disjoint and nonempty. A union of a partial partition is called its *support*. A *partition* of $\mathcal{X}$ is a partial partition whose support is $\mathcal{X}$. A set $\mathcal{G} \subseteq 2^{\mathcal{X}}$ is a *hierarchy on* $\mathcal{X}$ if any two elements of $\mathcal{G}$ are either disjoint or nested, i.e.:

$$\mathcal{X}_i \cap \mathcal{X}_j \in \{\beta(\mathcal{X}_i) \cap \beta(\mathcal{X}_j), \mathcal{X}_i, \mathcal{X}_j\} \text{ for any } \mathcal{X}_i, \mathcal{X}_j \in \mathcal{G} \tag{3.2}$$

Let X and Y be two distinctive elements of a hierarchy $\mathcal{G}$. We say that Y is a child of X and X is the parent of Y if $Y \subseteq X$ and for any $Z \in \mathcal{G}$, such that $Y \subseteq Z \subseteq X$, we have $Z = X$ or $Z = Y$. In other words, X is the smallest superset of Y among $\mathcal{G}$ elements. A partition factor $K \in \mathbb{Z}^+$ of a hierarchy $\mathcal{G}$ is the maximum number of children of a parent in $\mathcal{G}$. An element $L \in \mathcal{G}$ is called a leaf of $\mathcal{G}$ if it has no child. A hierarchy of partitions on $\mathcal{X}$ is formally defined as:

Definition 11. *A hierarchy $\mathcal{G}$ on $\mathcal{X}$ is a* ***hierarchy of partitions on $\mathcal{X}$*** *if the union of its leaves is a partition of $\mathcal{X}$ and $\mathcal{X} \in \mathcal{G}$.*

The term *multi-scale* in MSO is derived from the fact that a hierarchy of partitions has a set of partitions at *multiple scales* $h \in \mathbb{Z}_0^+$. Let $\mathcal{G}$ be a hierarchy of partitions on $\mathcal{X}$ created by an MSO algorithm. A hierarchy of partitions $\mathcal{G}$ may be represented by a tree structure $\mathcal{T}(\mathcal{G})$ whose nodes correspond to the elements of $\mathcal{G}$ with the root representing the whole space $\mathcal{X}$, while its edges link every node corresponding to a child in $\mathcal{G}$ to its corresponding parent's node. $\mathcal{T}$ is referred to as a *tree on $\mathcal{X}$* if it represents a hierarchy of partitions on $\mathcal{X}$. It is possible to index a node in $\mathcal{T}$ (and subsequently a cell of $\mathcal{X}$) by one or more integer attribute(s). For example, with a partition factor of K, a node can be indexed by its depth/scale h and an index i where $0 \leq i \leq K^h$ as (h, i) which corresponds to a cell/subspace $\mathcal{X}_{h,i} \subset \mathcal{X}$ and possesses up to K children nodes $\{(h+1, i_k)\}_{1 \leq k \leq K}$ such that:

$$\mathcal{X} = \cup_{i \in \{0,\ldots,K^h-1\}} \mathcal{X}_{h,i} \tag{3.3}$$

$$\mathcal{X}_{h,i} \cap \mathcal{X}_{h,j} = \beta(\mathcal{X}_{h,i}) \cap \beta(\mathcal{X}_{h,j}) \quad , \; i \neq j \tag{3.4}$$

$$\mathcal{X}_{h,i} = \cup_{1 \leq k \leq K} \mathcal{X}_{h+1,i_k} \quad , \; \forall h \in \mathbb{Z}_0^+ \tag{3.5}$$

Nodes can be indexed and grouped by any of their attributes such as depth, and size. For example, $S = \{i \in \mathcal{T} : \sigma(i) = s\}$ is the set of all nodes in the tree $\mathcal{T}$ of size s. Note that for any two elements $i, j \in S$, $i \cap j = \beta(i) \cap \beta(j)$. The set of leaves in $\mathcal{T}$ is denoted as $\mathcal{L}_{\mathcal{T}} \subseteq \mathcal{T}$ and its depth is denoted by $\texttt{depth}(\mathcal{T})$.

3.2 Multi-scale Search Optimization (MSO)

In this section, a general framework for sequential decision-making algorithms that dynamically expand a tree on the search space to find the optimal solution, is introduced. We refer to this framework as the Multi-Scale Optimization MSO, as these algorithms partition the search space over multiple scales. We formally define MSO algorithm as follows.

Definition 12. *An algorithm that constructs a hierarchy of partitions on the search space $\mathcal{X}$ whilst looking for f's global optimizer, is an MSO algorithm.*

MSO algorithms differ only in their strategies of growing and using the tree further to provide a good approximation of f's global optimizer point. Hence, we introduce a generic procedure of MSO algorithms.

3.2.1 A Generic Procedure of MSO Algorithms

An MSO algorithm can be regarded as a divide-and-conquer tree search algorithm. In an iterative manner: it evaluates and assesses a set of leaf nodes of its *tree on* $\mathcal{X}$; and selectively expands a subset of them.

Each node provides its approximate solution $\mathbf{x}_{h,i} \in \mathcal{X}_{h,i}$ which is referred to as (h, i)'s representative state (or sometimes *base point*). Let $\mathcal{A}$ be an MSO algorithm with up to J function evaluations per node, P evaluated nodes per iteration, Q subdivided/expanded nodes per iteration, and a partition factor K. Furthermore, let us denote $\mathcal{A}$'s tree $\mathcal{T}$ after iteration $t \geq 1$ by $\mathcal{T}_t$.

Remark 1. *Node expansions make the tree $\mathcal{T}$ grow with time. We may therefore index different forms of $\mathcal{T}$ by an index t to denote $\mathcal{T}$'s form $\mathcal{T}_{t+1}$ after a specific event with respect to its previous form $\mathcal{T}_t$. An event could be a single/several node(s) expansion, or a function evaluation (for which $\mathcal{T}_{t+1} = \mathcal{T}_t$). Expanding Q nodes $\{(h^q, i^q)\}_{1\leq q\leq Q} \in \mathcal{L}_t$ at time t, where (h^q, i^q) is the qth expanded node, results in:*

1. $\mathcal{T}_{t+1} = \mathcal{T}_t \cup \{(h^q + 1, i^q_k)\}_{1\leq q\leq Q, 1\leq k\leq K}$
2. $\mathcal{L}_{t+1} = \mathcal{L}_t \setminus \{(h^q, i^q)\}_{1\leq q\leq Q} \cup \{(h^q + 1, i^q_k)\}_{1\leq q\leq Q, 1\leq k\leq K}$
3. $|\mathcal{T}_{t+1}| = |\mathcal{T}_t| + QK$
4. $|\mathcal{L}_{t+1}| = |\mathcal{L}_t| + Q(K - 1)$
5. $H(\mathcal{T}_{t+1}) = \max(\boldsymbol{\mathtt{depth}}(\mathcal{T}_t), max_{1\leq q\leq Q}\; h^q + 1)$

At iteration $t+1$, two steps take place, namely *evaluation* and *expansion* illustrated as follows.[1]

1. *Leaf Node(s) Evaluation*: In order to find $\mathbf{x}_{h,i}$, $J \geq 1$ function evaluations are performed within $\mathcal{X}_{h,i}$ as a part of the sequential decision making process and out of the v-evaluation budget. These J function evaluations of a leaf node may be independent of its ancestors' and may not happen at the same iteration. We refer to the process of evaluating f within $\mathcal{X}_{h,i}$ J times as *evaluating the node* (h, i). Leaf nodes with function evaluations less than J

[1]At $t = 1$, the root node gets evaluated J times and partitioned into K nodes; $P = Q = 1$ for all MSO algorithms, irrespective of their values at $t > 1$.

are referred to as *under-evaluated* nodes. Otherwise, they are called *evaluated* nodes. The set of evaluated nodes up to iteration t (inclusive) are denoted as $\mathcal{E}_t \subseteq \mathcal{L}_{\mathcal{T}_{t-1}}$. At each iteration, $\mathcal{A}$ **selects** $P \geq 1$ under-evaluated nodes (if any) to be evaluated. Denote the set of selected-to-be-evaluated nodes as $\mathcal{P}_{t+1} \overset{\text{def}}{=} \cup_{h\in\{0,\ldots,\texttt{depth}(\mathcal{T}_t)\}}\mathcal{P}_{t+1,h} \subseteq \mathcal{L}_{\mathcal{T}_t} \setminus \mathcal{E}_t$ where:

$$\begin{aligned}\mathcal{P}_{t+1,h} \overset{\text{def}}{=} \{&(h,i) : 0 \leq i \leq K^h - 1,\\ &(h,i) \text{ is evaluable at } t+1 \text{ according to } \mathcal{A}\}\end{aligned} \tag{3.6}$$

and $|\mathcal{P}_{t+1}| \leq P$.

2. *Leaf Node(s) Expansion*: $\mathcal{A}$ inspects the *evaluated* leaf nodes (if any), and **selects** $Q \geq 1$ among them to be split/partitioned. These selected nodes represent the sub-domain in which $\mathcal{A}$ thinks $\mathbf{x}^*$ potentially lies and hence a finer search is favored. We refer to the process of splitting/ partitioning a leaf node (h,i) into its K children as *expanding the node* (h,i). Denote the set of selected-to-be-expanded nodes as $\mathcal{Q}_{t+1} \overset{\text{def}}{=} \cup_{h\in\{0,\ldots,\texttt{depth}(\mathcal{T}_t)\}}\mathcal{Q}_{t+1,h} \subseteq \mathcal{E}_{t+1} \subseteq \mathcal{L}_{\mathcal{T}_t}$ where:

$$\begin{aligned}\mathcal{Q}_{t+1,h} \overset{\text{def}}{=} \{&(h,i) : 0 \leq i \leq K^h - 1,\\ &(h,i) \text{ is expandable at } t+1 \text{ according to } \mathcal{A}\}\end{aligned} \tag{3.7}$$

and $|\mathcal{Q}_{t+1}| \leq Q$.

Algorithm 6 Pseudo-code for Multi-scale Search Optimization (MSO)

```
   Input: function to be optimized as a black-box $f$, search space $\mathcal{X}$, budget evaluation $v$
   Initialization: $\mathcal{T} \leftarrow$ initial tree with one node $(0,0)$ with $\mathcal{X}_{0,0} = \mathcal{X}$
   Output: approximation of $f$'s minimizer
1: while the evaluation budget is not exhausted do
2:     Evaluate the nodes $\in \mathcal{P}$.
3:     Expand the nodes $\in \mathcal{Q}$ and add their child nodes in $\mathcal{T}$.
4: end while
5: return $\mathbf{x}(v) \in \arg\max_{\mathbf{x}_{h,i}:(h,i)\in\mathcal{T}} f(\mathbf{x}_{h,i})$
```

After v function evaluations, $\mathcal{A}$ returns $\mathbf{x}(v)$:

$$\mathbf{x}(v) \in \arg\max_{\mathbf{x}_{h,i}:(h,i)\in\mathcal{T}} f(\mathbf{x}_{h,i}) \tag{3.8}$$

as an approximate of f's global maximizer point.[2] This procedure is summarized in Algorithm 6.

One can have different MSO algorithms based on the defining policies of the evaluable set $\mathcal{P}$ and the expandable set $\mathcal{Q}$. In fact, an MSO algorithm $\mathcal{A}$ seeks a balance between two components of search [186]: *exploration*, and *exploitation*. Exploration (or global search) refers to the process of learning more about the search space. On the other hand, exploitation (or local search) is the process of acting optimally according to current knowledge. Given a finite computational budget, excessive exploration (e.g., an algorithm with a broad-search tree) leads to slow convergence, whereas excessive exploitation (e.g., an algorithm with a deep-search tree) leads to premature convergence to a local maximum and hence a trade-off must be made. Accordingly, $\mathcal{A}$ chooses $\mathcal{P}$ and $\mathcal{Q}$ to be of leaf nodes of its tree $\mathcal{T}$ that preferably possess jointly good exploration and exploitation $\mathcal{A}$-defined scores.

Let $l : \mathcal{L} \to \mathbb{R}$ be the exploitation score function and $g : \mathcal{L} \to \mathbb{R}$ be the exploration score function. The projection of a node (h, i) onto the exploitation axis is then denoted as $l_{h,i} = l((h, i))$ whereas its projection onto the exploration axis is denoted as $g_{h,i} = g((h, i))$.

Exploitation (Local) Score As the function value at (h, i)'s base point, $f(\mathbf{x}_{h,i})$ is the best value of f within $\mathcal{X}_{h,i}$ according to $\mathcal{A}$'s current belief. It is a direct indicator of (h, i)'s exploitation (local) score. Therefore, $l_{h,i}$ is taken as $f(\mathbf{x}_{h,i})$ or its approximate (if it is unavailable) with an absolute error less than or equal to $\eta \geq 0$:[3]

$$|l_{h,i} - f(\mathbf{x}_{h,i})| \leq \eta \tag{3.9}$$

Exploration (Global) Score While $l_{h,i}$ reflects $\mathcal{A}$'s guess on the best value of f within $\mathcal{X}_{h,i}$, (h, i)'s exploration (global) score $g_{h,i}$ is regarded as the likelihood of finding a better value than $f(\mathbf{x}_{h,i})$ within $\mathcal{X}_{h,i}$. This is correlated with the bulk of unexplored space in $\mathcal{X}_{h,i}$ and often quantified by a rough measure of its size. For example, $g_{h,i} = \texttt{depth}(\mathcal{T}) - h$ or $g_{h,i} = \sigma(\mathcal{X}_{h,i})$. Hence, one can argue that nodes

[2]This is the same $\mathbf{x}(v)$ of the maximization version of Problem (2.2).

[3]Bounding the approximation error could be valid with a probability of $\gamma \geq 0$. In such case, any related analysis holds with a probability of γ.

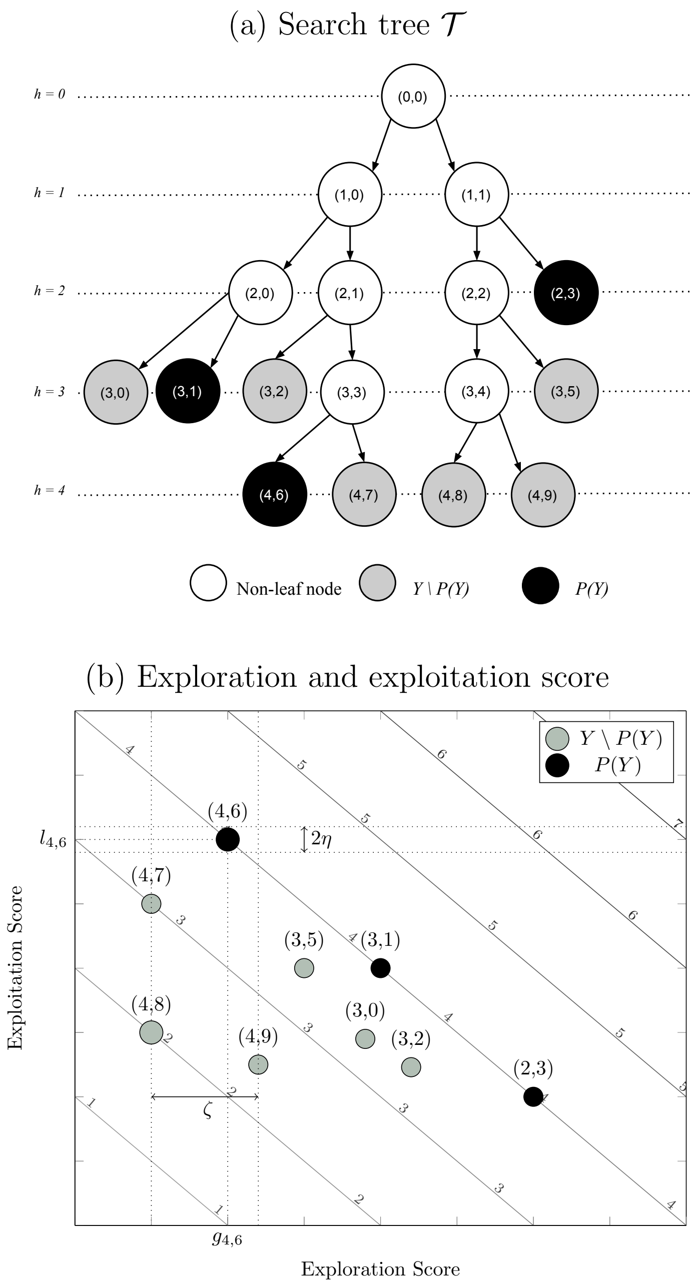


Figure 3.1: The process of computing exploration and exploitation scores for the set of leaf nodes $\mathcal{L} \subseteq \mathcal{T}$ (a) of an MSO algorithm $\mathcal{A}$ can be regarded as projecting them onto a 2-D Euclidean space (b) $\mathbb{R}^2$ via the mapping function $s : \mathcal{L} \rightarrow \mathbb{R}^2$. Y represents $\mathcal{L}$'s image under s and $P(Y) \subseteq Y$ is the potentially optimal set. Here $P(Y)$ lies on the level set of $b(x) = l_x + \lambda \cdot g_x$ that corresponds to the value 4 (the greatest among Y's). $\mathcal{P}$ and $\mathcal{Q}$ are chosen from the set of leaf nodes whose image under s is $P(Y)$.

of the same depth have the same exploration score or may differ up to a certain limit $\zeta \geq 0$:

$$|g_{h,i} - g_{h,j}| \leq \zeta \quad \forall h \in \mathbb{Z}_0^+,\ i,j \in \{0,\dots,K^h - 1\},\ i \neq j \tag{3.10}$$

In the exploration-exploitation plane, each node (h,i) of $\mathcal{L}$ is represented by the point $(g_{h,i}, l_{h,i})$. Let Y be the set of these points. In other words, $Y \overset{\text{def}}{=} \{(g(x), l(x)) : x \in \mathcal{L}\}$. $\mathcal{A}$ defines a measure-of-optimality function $b : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that $\mathcal{P}$ and $\mathcal{Q}$ are chosen from the set of leaf nodes whose exploration-exploitation points are the *potentially optimal set* $P(Y) \overset{\text{def}}{=} \{y \in Y : \{x \in Y : b(x) > b(y), x \neq y\} = \emptyset\}$. In other words, $P(Y)$ is the set of points $\subseteq Y$ whose level set of b (b-value) corresponds to the highest value among all points of Y. Generally, b is a weighted sum of l and g of the form:

$$b_x = b((g(x), l(x))) = l(x) + \lambda \cdot g(x), \quad x \in \mathcal{L},\ \lambda \geq 0 \tag{3.11}$$

trading off between local and global searches. It is important to note that g, l, b used for $\mathcal{P}$ may not be the same as for $\mathcal{Q}$.

We can visualize the process of computing these scores as projecting the leaf nodes of $\mathcal{T}$ (depicted in Figure 3.1(a)) onto a 2-D Euclidean space $\mathbb{R}^2$, as illustrated in Figure 3.1(b), whose vertical and horizontal coordinate axes represent the domain of values for exploitation score and exploration score, respectively. From Figure 3.1(b), $l_{h,i}$ (3.9) for the node $(4,6)$, $g_{h,i}$ (3.10) for nodes at depth 4, and $P(Y)$ for $\lambda = 1$ are shown.

Remark 2. *One can introduce heuristics to consider, compute, and compare the b-values only for a subset of $\mathcal{L}$ at a given iteration from which $\mathcal{P}$ and $\mathcal{Q}$ are chosen rather than the whole set $\mathcal{L}$.*

3.2.2 MSO Algorithms in the Literature

In the literature, several established algorithms satisfy the definition of MSO algorithms. Examples include Lipschitzian Optimization (`LO`) [54, 55], Dividing Rectangles (`DIRECT`) [56], Multilevel Coordinate Search (`MCS`) [27], Deterministic Optimistic Optimization (`DOO`) [34], Simultaneous Optimistic Optimization (`SOO`) [34], and their variants.

As described in Chapter 2, these algorithms can be grouped into two categories: one category requires the knowledge about f's smoothness, with the b-values being a weighted sum of the local and global scores, `LO` and `DOO` are examples of such a category. On the other hand, `DIRECT`, `MCS`, and `SOO` still make an assumption about f's smoothness, but knowledge about it may not be available. Nevertheless, these algorithms account for more than one possible setting of f's smoothness by grouping nodes based on their global scores for which local scores play a role in analyzing each group separately. We describe algorithms in these two categories in accordance with the generic procedure discussed in Section 3.2.1.

3.2.2.1 Lipschitzian Optimization (`LO`)

As discussed in Chapter 2, `LO` looks for the optimal solution by approximating f with a piece-wise linear bound $\hat{f}$ over partitions of the decision space $\mathcal{X}$.

With respect to the generic procedure outlined in Section 3.2.1, `LO` has the following settings: J is of $\mathcal{O}(2^n)$; $K = P = 2^n$, that is leaf nodes created in an iteration get evaluated in the next iteration; and $Q = 1$. The two steps of `LO` at iteration $t+1$ are summarized as follows.

1. *Leaf Node(s) Evaluation*: $\mathcal{P}_{t+1}$ is $\mathcal{L}_{\mathcal{T}_t} \setminus \mathcal{E}_t$, i.e., the set of leaf nodes that are not evaluated. For a node $(h,i) \in \mathcal{P}_{t+1}$, 2^n function evaluations—at (h,i)'s vertices—are performed (hence (h,i) is $\in \mathcal{E}_{t+1}$), and the b-value $(\hat{f}(\mathbf{x}_{h,i}))$ is computed as shown in Section 3.3.2.1.

2. *Leaf Node(s) Expansion*: $\mathcal{Q}_{t+1}$ is simply one node among those $\in \mathcal{E}_{t+1}$ whose b-value is the maximum, where ties are broken arbitrarily—that is, if there are more than one node whose b-value is the maximum, then any node of these is selected arbitrarily to be in $\mathcal{Q}_{t+1}$.

3.2.2.2 Deterministic Optimistic Optimization (`DOO`)

Munos [34] proposed `DOO` by assuming f to be locally smooth (around one of its global optima) with respect to a semi-metric ℓ. This assumption in `DOO` offers a relaxation over the restrictive assumption of `LO`. `DOO` estimates, based on local smoothness, the maximum upper bound of f within a partition. Similar to `LO`, this upper bound is used to guide the hierarchical partitioning of the space.

`DOO` constructs a tree on $\mathcal{X}$ whose settings with respect to the generic procedure outlined in Section 3.2.1, are the following: $J = 1$; K and P are equal (as with `LO`) and treated as a parameter by `DOO` with a default value of 3. The two steps of `DOO` at iteration $t+1$ are summarized as follows.

1. *Leaf Node(s) Evaluation*: $\mathcal{P}_{t+1}$ is $\mathcal{L}_{\mathcal{T}_t} \setminus \mathcal{E}_t$, i.e., the set of leaf nodes that are not evaluated. For a node $(h,i) \in \mathcal{P}_t$, one function evaluations at (h,i)'s center is performed (hence (h,i) becomes an evaluated node), and the b-value is computed as shown in Section 3.3.2.2.

2. *Leaf Node(s) Expansion*: $\mathcal{Q}_{t+1}$ is simply one node among those $\in \mathcal{E}_{t+1}$ whose b-value is the maximum, where ties are broken arbitrarily.

3.2.2.3 DIviding RECTangles (`DIRECT`)

First of all, `DIRECT` does not need the knowledge of the Lipschitz constant L. Instead, it carries out the search by using all possible values of L from zero to infinity in a simultaneous framework, thereby balancing global and local search and improving the convergence speed significantly. This is captured by the heuristic rule (3.14). In addition to that, it introduces an additional constraint—on $\mathcal{Q}$—whose tightness depends on a parameter $\epsilon \geq 0$ of a nontrivial amount. If f_{max} is the best current function value, then $\epsilon|f_{max}|$ is the minimum amount by which the b-values of $\mathcal{Q}$ must exceed f_{max} as will be illustrated later. This ϵ-constraint (3.13) helps in protecting the algorithm from excessive local search. Furthermore, `DIRECT` cuts down the computational complexity from $\mathcal{O}(2^n)$ to $\mathcal{O}(1)$ by evaluating the cells' center points (which are their base points as well) instead of their vertices.

With respect to the generic procedure outlined in Section 3.2.1, $J \leq 1$,[4] $K \leq 3n$, $P \leq 3n$, and $Q = \mathcal{O}(K^{\texttt{depth}(\mathcal{T})})$. Furthermore, let σ be a measure of a node's size such that a node (h,i) has a size of $\sigma_{h,i} = \sigma(\mathcal{X}_{h,i}) = ||d_{h,i}||_2$ where $d_{h,i}$ is $\mathcal{X}_{h,i}$'s diameter. Denote the set of evaluated-node sizes by $S_t \stackrel{\text{def}}{=} \{\sigma_{h,i} : (h,i) \in \mathcal{E}_t\}$; and denote the set of iteration indices $\{t, \ldots, t + |S_t| - 1\}$ by I_t, where I_1 is the first iteration batch, $I_{|S_t|+1}$ is the second iteration batch, and so on. With $\acute{t} \in I_t$, $\sigma^{\acute{t}}_{S_t}$ is the $(\acute{t} - t + 1)^{th}$ element of S_t in a descending order. Moreover, let f^t_{max} be the best function value achieved before t. The two steps of `DIRECT` at iteration $\acute{t} \in I_t$ are then summarized as follows.

1. *Leaf Node(s) Evaluation*: $\mathcal{P}_{\acute{t}}$ is $\mathcal{L}_{\mathcal{T}_{\acute{t}-1}} \setminus \mathcal{E}_{\acute{t}-1}$, i.e., the set of leaf nodes that are not evaluated. For a node $(h,i) \in \mathcal{P}_{\acute{t}}$, one function evaluation at (h,i)'s center is performed (hence (h,i) is $\in \mathcal{E}_{\acute{t}}$), and the b-value is computed as shown in Section 3.3.2.3.

[4]One function evaluation may belong to one or more nodes.

2. *Leaf Node(s) Expansion*: Let $\mathbb{Q}_{\acute{t}}^{I_t}$ be the set of evaluated nodes whose size is $\sigma_{S_t}^{\acute{t}}$; mathematically: $\mathbb{Q}_{\acute{t}}^{I_t} \overset{\text{def}}{=} \{(h,i) : (h,i) \in \mathcal{E}_t, \sigma_{h,i} = \sigma_{S_t}^{\acute{t}}\}$, and $b_{\acute{t}}$ is $\max_{(h,i)\in\mathbb{Q}_{\acute{t}}^{I_t}} b_{(h,i)}$, then $\mathcal{Q}_{\acute{t}}$ are set of nodes where each node $(h,i) \in \mathbb{Q}_{\acute{t}}^{I_t}$ such that:

$$b_{(h,i)} \quad = \quad b_{\acute{t}} \tag{3.12}$$

and there exists $\hat{L} \geq 0$ such that:

$$b_{(h,i)} + \hat{L}\sigma_{S_t}^{\acute{t}} \quad \geq \quad (1+\epsilon)f_{max}^{t} \tag{3.13}$$

$$b_{(h,i)} + \hat{L}\sigma_{S_t}^{\acute{t}} \quad \geq \quad b_{\hat{t}} + \hat{L}\sigma_{S_t}^{\hat{t}} \quad, \forall \hat{t} \in I_t \setminus \acute{t} \tag{3.14}$$

If such node does not exist, the algorithm proceeds to the next iteration without expanding any node at $\acute{t}$ (see [139] for more details on how (3.13) and (3.14) are tested).

One can notice that within a batch of iterations, nodes are first contested among others of the same size, then among others of different size.

3.2.2.4 Multilevel Coordinate Search (`MCS`)

Huyer and Neumaier [27] addressed the slow-convergence shortcoming of `DIRECT` in optimizing functions whose optimizers lie at the boundary of $\mathcal{X}$; and devised a global optimization algorithm (called Multilevel Coordinate Search (`MCS`)). `MCS` partitions $\mathcal{X}$ into hyperrectangles of uneven sizes whose base points are not necessarily the center points.

For a node $(h,i) \in$ its tree $\mathcal{T}$, `MCS` assigns a rank measure $s_{h,i} \geq h$, which is used in selecting the expandable set $\mathcal{Q}$. The measure $s_{h,i}$ captures how many times a node (h,i) has been part of/candidate for an expansion process. We refer to this measure as pseudo-depth because it does not reflect the actual depth of the node. The children of node (h,i) with pseudo-depth $s_{h,i}$, can have upon creation a pseudo-depth of $s_{h,i}+1$ or $\min(s_{h,i}+2, s_{max})$ based on its size with respect to its siblings. The expandable set $\mathcal{Q}$ is selected based on pseudo-depth.

A node (h,i) has a set of numbers $\{n_j^{(h,i)}\}_{1\leq j\leq n}$ where n_j denotes the number of times $\mathcal{X}_{h,i}$ has been part of an expansion along coordinate j. $\mathcal{T}$'s depth is controlled through a single parameter s_{max} which forces the tree to a maximal depth of s_{max}. Given a fixed budget of node expansions, greater s_{max} reduces the probability of expanding an optimal node in $\mathcal{T}$, and hence a greater regret bound.

There are two heuristic rules employed to post-process $\mathcal{Q}$. The first rule (3.16) is based on $\{n_j^{(h,i)}\}_{1\leq j\leq n}$ to expand nodes which have high pseudo-depths, yet there is at least one coordinate along which their corresponding hyperrectangles have not been part of an expansion very often. The second rule (3.17) is to expand a node along a coordinate where the maximal expected gain in function value is large enough; the gain $\hat{e}_j^{(h,i)}$ for a node (h,i) along coordinate j is computed using a local quadratic model [27]. Accordingly, if $\max_{1\leq j\leq n}\hat{e}_j^{(h,i)}$ is large enough, (h,i) is then eligible for expansion along the coordinate $\arg\max_{1\leq j\leq n}\hat{e}_j^{(h,i)}$. If any of these rules does not hold for a node $\in \mathcal{Q}$, it is removed from $\mathcal{Q}$ and its pseudo-depth is increased by one. Base points at depth s_{max} are put into a *shopping basket*, assuming them to be useful points. One can accelerate convergence by starting local searches from these points before putting them into the shopping basket.

With respect to the generic procedure outlined in Section 3.2.1, the settings of MCS are the following: $J=1$; Based on the partitioning coordinate, P is $L_i \geq 3$ where L_i is the number of sampled points along coordinate i. Consequently, K could be $2L_i$, $2L_i-1$, or $2L_i-2$; and $Q=1$. Furthermore, let I_t be the set of iteration indices $\{t,\ldots,t+s_{max}-1\}$; I_1 is the first iteration batch, $I_{s_{max}}$ is the second iteration batch, and so on. The two steps of MCS at iteration $\acute{t}\in I_t$ are summarized as follows.

1. *Leaf Node(s) Evaluation*: $\mathcal{P}_{\acute{t}}$ is $\mathcal{L}_{\mathcal{T}_{\acute{t}-1}}\setminus\mathcal{E}_{\acute{t}-1}$, i.e., the set of leaf nodes that are not evaluated. For a node $(h,i)\in\mathcal{P}_{\acute{t}}$, one function evaluation is performed (hence $(h,i)\in\mathcal{E}_{\acute{t}}$), and the b-value is computed as shown in Section 3.3.2.4.

2. *Leaf Node(s) Expansion*: Let $\mathbb{Q}_{\acute{t}}^{I_t}$ be $\{(h,i):(h,i)\in\mathcal{E}_{\acute{t}}, s_{h,i}=\acute{t}-t\}$ (if $\mathbb{Q}_{\acute{t}}^{I_t}=\emptyset$, the current iteration is simply skipped to $\acute{t}+1$), and $b_{\acute{t}}$ is $max_{(h,i)\in\mathbb{Q}_{\acute{t}}^{I_t}}b_{(h,i)}$ then, $\mathcal{Q}_{\acute{t}}$ is simply the node $(h,i)\in\mathbb{Q}_{\acute{t}}^{I_t}$ such that:

$$b_{(h,i)} \quad = \quad b_{\acute{t}} \tag{3.15}$$

and fulfills the one of the two heuristics:

$$\acute{t}-t \quad > \quad 2n\left(\min_{1\leq j\leq n} n_j^{(h,i)}+1\right) \tag{3.16}$$

or:

$$b_{(h,i)} \quad \geq \quad f_{max}-\max_{1\leq j\leq n}\hat{e}_j^{(h,i)} \tag{3.17}$$

where ties are broken arbitrarily. If none of the heuristic rules holds, the node's pseudo-depth is set to $s_{h,i}+1$ and proceeds to the next iteration where it may be considered again.

3.2.2.5 Simultaneous Optimistic Optimization (SOO)

SOO [34] tries to approximate DOO's behavior when ℓ is unknown. It expands simultaneously all the nodes (h, i) of its tree $\mathcal{T}$ for which there exists a semi-metric ℓ such that the corresponding upper bound would be the greatest. This is simulated by expanding at most a leaf node per depth if such node has the greatest $f(\mathbf{x}_{h,i})$ with respect to leaf nodes of the same or lower depths. In addition to that, the algorithm takes a function $p \rightarrow h_{max}(p)$, as a parameter, which forces $\mathcal{T}$ to a maximal depth of $h_{max}(p) + 1$ after p node expansions (e.g., $h_{max}(p) = p^{\epsilon}$ where $\epsilon > 0$).

With respect to the generic procedure outlined in Section 3.2.1, the settings of SOO are the following: $J = 1$; K and P are equal and treated as a parameter by SOO with a default value of 3; and $Q = 1$. Furthermore, let I_t be the set of iteration indices $\{t, \ldots, t + h_{max}(p)\}$; I_1 is the first iteration batch, $I_{h_{max}(p)}$ is the second iteration batch, and so on. The two steps, according to [186], of SOO at iteration $\acute{t} \in I_t$ are summarized as follows.

1. *Leaf Node(s) Evaluation*: $\mathcal{P}_{\acute{t}}$ is $\mathcal{L}_{\mathcal{T}_{\acute{t}-1}} \setminus \mathcal{E}_{\acute{t}-1}$, i.e., the set of leaf nodes that are not evaluated. For a node $(h, i) \in \mathcal{P}_{\acute{t}}$, one function evaluation at (h, i)'s center is performed (hence (h, i) is $\in \mathcal{E}_{\acute{t}}$), and the b-value is computed as shown in Section 3.3.2.5.

2. *Leaf Node(s) Expansion*: Let $\mathbb{Q}^{I_t}_{\acute{t}}$ be $\{(h, i) : (h, i) \in \mathcal{E}_{t-1}, h = \acute{t} - t\}$, and $b_{\acute{t}}$ is $\max_{(h,i) \in \mathbb{Q}^{I_t}_{\acute{t}}} b_{(h,i)}$ then $\mathcal{Q}_{\acute{t}}$ is simply the node $(h, i) \in \mathbb{Q}^{I_t}_{\acute{t}}$ such that:

$$b_{(h,i)} = b_{\acute{t}} \tag{3.18}$$

$$b_{(h,i)} \geq b_{\hat{t}} \quad \forall \hat{t} \; t \leq \hat{t} < \acute{t} \tag{3.19}$$

where ties are broken arbitrarily. If no such node exists, the current iteration is simply skipped to $\acute{t} + 1$.

The Expandable Set $\mathcal{Q}$ While the selected-to-be-evaluated set $\mathcal{P}$ is almost the same for all the algorithms (a node is evaluated upon its creation), the process of selecting $\mathcal{Q}$ differs among them. Figure 3.2 shows a typical scenario of what $\mathcal{Q}$ could be in one (a batch of) iteration(s). Similar to Figure 3.1(b), it shows the evaluated leaves projected into the exploration-exploitation space. In LO, and

DOO; each iteration is independent of each other; and a node is selected if it is the first node to lie on the curve from above. In DIRECT, MCS, and SOO; the case is different, iterations within a batch of iterations are co-dependent; a node is selected if it is among the first nodes to lie on the corresponding curve from above. However, from their visualizations, it can be argued that SOO and DIRECT have a greedier behavior than MCS as they only expand a set of Pareto-optimal nodes in the exploration-exploitation plane.

3.3 Convergence Analysis of MSO algorithms

In this section, we propose a theoretical methodology for finite-time analysis of MSO algorithms. It is then applied to analyze different MSO algorithms in the literature.

3.3.1 Theoretical Framework for Convergence

We derive a measure of convergence rate by analyzing the complexity of the regret (2.2) as a function of the number of node expansions p. We upper-bound $r(p)$ by quantifying the amount of exploration required in each of the multi-scale partitions to achieve a near-optimal solution. In line with [34], three basic assumptions are made; the first two assumptions assist in establishing a bounded (finite) bound on $r(p)$, whereas the third assumption helps in computing it in finite time.

3.3.1.1 Bounding the regret $r(p)$

To bound $r(p)$ (2.2), one needs to assume the characteristics of f. In LO, f is assumed to be Lipschitz-continuous [54], whereas DOO and SOO assume local smoothness on f [34]. Here, we impose the local smoothness assumption on f, defined formally as local Hölder continuity. Let $\mathcal{T}$ be a tree on $\mathcal{X}$ created by an MSO algorithm and $\ell : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ be a semi-metric such that $\ell(\mathbf{x}, \mathbf{y}) = L||\mathbf{x} - \mathbf{y}||^\alpha$ with L and α being positive real constants.

Assumption 1 (local Hölder continuity)**.** *f is **Hölder-continuous with respect to (at least) one of its global optimizer $\boldsymbol{x}^*$**, that is to say, for all $\boldsymbol{x} \in \mathcal{X}$, it satisfies Hölder condition [187] around $\boldsymbol{x}^*$:*

$$|f^* - f(\boldsymbol{x})| \leq L||\boldsymbol{x}^* - \boldsymbol{x}||^\alpha = \ell(\boldsymbol{x}^*, \boldsymbol{x}) \tag{3.20}$$

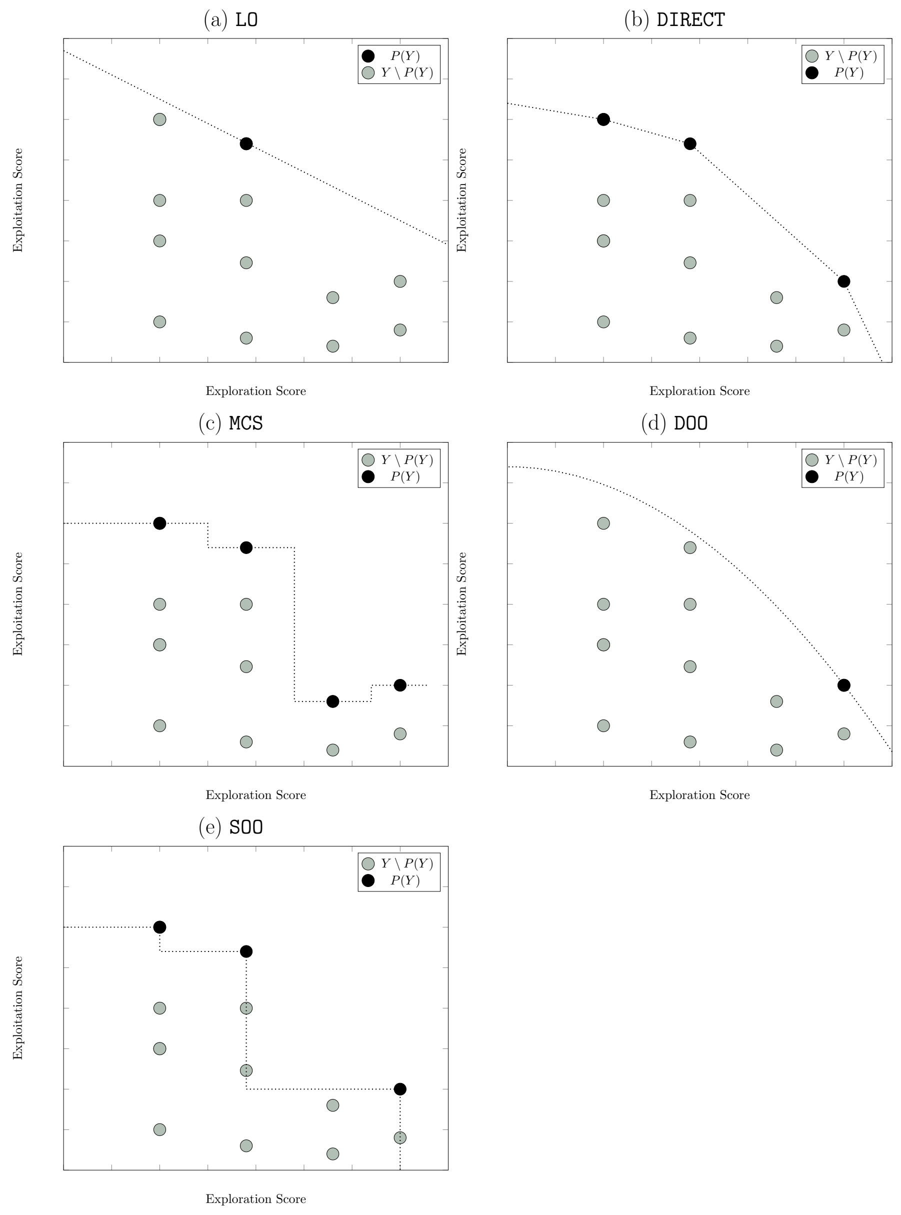


Figure 3.2: Selecting expandable leaf node(s) $\mathcal{Q}$ (represented by black dots) for an iteration in `LO` (a), for a batch of iterations in `DIRECT` (b) , a batch of iterations in `MCS` (c), an iteration in `DOO` (d), and a batch of iterations in `SOO` (e). The set Y, whose elements are represented by black and gray dots, is the set of projected evaluated leaves into the exploration-exploitation space.

Remark 3. *The class of Hölder-continuous functions is very broad. In fact, it has been shown in [52, 139] that among the Lipschitz-continuous functions (which are Hölder-continuous with $\alpha = 1$) are convex/concave functions over a closed domain and continuously differentiable functions.*

Before applying Assumption 1, we define the following.

Definition 13. *A node $(h, i) \in \mathcal{T}$ is* ***optimal*** *if $\boldsymbol{x}^* \in \mathcal{X}_{h,i}$.*

Definition 14. *$h^*_p \in \mathbb{Z}^+_0$ is the depth of the deepest optimal node that has been expanded up to p expansions* [5] *and $(h^*_p, i^*_p) \in \mathcal{T}$ is the* ***deepest expanded optimal node****.*

Given that (h^*_p, i^*_p) is known, we can bound the regret as follows.

$$\begin{aligned}
r(p) &= f^* - f(\mathbf{x}(p)) && (3.21)\\
&\leq f^* - f(\mathbf{x}_{h^*_p,i^*_p}) && (3.22)\\
&\leq \ell(\mathbf{x}^*, \mathbf{x}_{h^*_p,i^*_p}) \qquad \text{from (3.20)} && (3.23)\\
&\leq \sup_{\mathbf{x}\in\mathcal{X}_{h^*_p,i^*_p}} \ell(\mathbf{x}, \mathbf{x}_{h^*_p,i^*_p}) && (3.24)
\end{aligned}$$

Presume that the evaluation of a node's children is always coupled with its expansion.[6] This means that $f(\mathbf{x}_{h^*_p+1,i^*_{nk}})$ for $1 \leq k \leq K$ are known. Consequently, there exists a tighter bound on $r(p)$ than (3.24) as $\mathbf{x}^*$ is in one of (h^*_p, i^*_p)'s children:

$$r(p) \leq \sup_{1\leq k\leq K} \sup_{\mathbf{x}\in\mathcal{X}_{h^*_p+1,i^*_{pk}}} \ell(\mathbf{x}, \mathbf{x}_{h^*_p+1,i^*_{pk}}) \leq \sup_{\mathbf{x}\in\mathcal{X}_{h^*_p,i^*_p}} \ell(\mathbf{x}, \mathbf{x}_{h^*_p,i^*_p}) \tag{3.25}$$

In order to have a bounded (finite) bound to on $r(p)$, cells of $\mathcal{T}$'s nodes need to be bounded. The next assumption implies that the bound on $r(p)$ is bounded:

Assumption 2 (Bounded Cells)**.** *There exists a decreasing sequence $\delta(h) = c\rho^h$ in h such that*

$$\sup_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{X}_{h,i}} \ell(\boldsymbol{x}, \boldsymbol{y}) \leq \delta(h) \quad \forall (h, i) \in \mathcal{T} \tag{3.26}$$

where c is a positive real number and $\rho \in (0, 1)$.

Consequently, from (3.25) the regret is bounded by:

$$r(p) \leq \delta(h^*_p + 1) \tag{3.27}$$

Thus, finding h^*_p is the key to bound the regret. This is discussed in the next section.

[5] We are treating MSO as a v-round sequential decision making process. For MSO algorithms, v may correspond to the number of iterations, evaluations, or expansions. All three quantities are interrelated. Generally, we mean by v the number of evaluations if not stated otherwise.

[6] we shall consider this presumption throughout our analysis. In other words, for p node expansions, there are $v = Kp$ node evaluations.

3.3.1.2 Finding the depth of the deepest expanded optimal node, h^*_p

MSO algorithms may inevitably expand non-optimal nodes as part of their exploration -vs.-exploitation strategy. Hence, the relationship between the number of expansions n and h^*_p therefore may not be straight forward. Let us take an MSO algorithm $\mathcal{A}$ whose $\mathcal{T}$'s deepest expanded optimal node after p expansions is at depth $\acute{h} < \mathtt{depth}(\mathcal{T})$- i.e., $h^*_p = \acute{h}$). For any node $(\acute{h}+1, i)$ to be expanded before $(\acute{h}+1, i^*)$, the optimal node at depth $\acute{h}+1$; the following must hold for $f(\mathbf{x}_{\acute{h}+1,i})$:

$$\begin{aligned} b_{(\acute{h}+1,i)} &\geq b_{(\acute{h}+1,i^*)} && (3.28)\\ l_{\acute{h}+1,i} + \lambda\cdot g_{\acute{h}+1,i} &\geq l_{\acute{h}+1,i^*} + \lambda\cdot g_{\acute{h}+1,i^*} && (3.29)\end{aligned}$$

from (3.9) and (3.10):

$$f(\mathbf{x}_{\acute{h}+1,i}) + 2\cdot\eta + \lambda\cdot\zeta \geq f(\mathbf{x}_{\acute{h}+1,i^*}) \qquad (3.30)$$

from (3.24) and (3.26):

$$f(\mathbf{x}_{\acute{h}+1,i}) + 2\cdot\eta + \lambda\cdot\zeta + \delta(\acute{h}+1) \geq f^* \qquad (3.31)$$

Let us state three definitions to quantify and analyze nodes that satisfy (3.31).

Definition 15. *The set $\mathcal{X}_\epsilon \overset{\text{def}}{=} \{\boldsymbol{x} \in \mathcal{X}, f(\boldsymbol{x}) + \epsilon \geq f^*\}$ is the set of ϵ-**optimal states** in $\mathcal{X}$.*

Definition 16. *The set $\mathcal{I}^\epsilon_h \overset{\text{def}}{=} \{(h,i) \in \mathcal{T} : f(\boldsymbol{x}_{h,i}) + \epsilon \geq f^*\}$ is the set of ϵ-**optimal nodes at depth** h in $\mathcal{T}$. For instance, $\mathcal{I}^{2\cdot\eta+\lambda\cdot\zeta+\delta(\acute{h}+1)}_{\acute{h}+1}$ is the set of nodes that satisfy (3.31).*

Definition 17. *$h^\epsilon_p \in \mathbb{Z}^+_0$ is the depth of the deepest ϵ-optimal node that has been expanded up to p expansions.*

Let $\varepsilon(h) = 2\cdot\eta + \lambda\cdot\zeta + \delta(h)$. If $\mathcal{A}$ allows expanding more than one node per depth, then we are certain that the optimal node at depth h gets expanded if all the nodes in $\mathcal{I}^{\varepsilon(h)}_h$ are expanded. Hence, h^*_p is guaranteed to be greater than or equal to h. Mathematically,

$$h^*_p \geq argmax\ \{h : h \in \{0, \ldots, \mathtt{depth}(\mathcal{T}) - 1\},\ x \text{ is expanded } \forall x \in \mathcal{I}^{\varepsilon(h)}_h\} \qquad (3.32)$$

Therefore, the relationship between n and h^*_p can be established by finding out how many expansions n are required to expand, at a given depth h, all the nodes in $\mathcal{I}^{\varepsilon(h)}_h$. It depends on two factors:

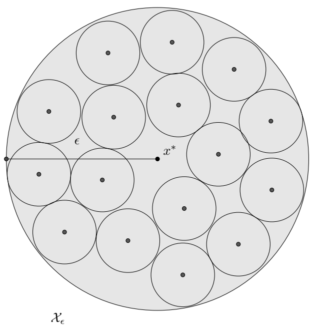


Figure 3.3: $\mathcal{X}_\epsilon$ in a 2-dimensional space is an $\acute{\ell}$-circle (Here, $\acute{\ell}$ and ℓ are the l_2 norms, with α and β set to 1) centered at $\mathbf{x}^*$ with a radius of ϵ.

1. The number of the $\varepsilon(h)$-optimal nodes at depth h, $|\mathcal{I}_h^{\varepsilon(h)}|$.
2. $\mathcal{A}$'s strategy in expanding $\mathcal{I}_h^{\varepsilon(h)}$. If $\mathcal{A}$ has nodes at depth h, where $\acute{h}+1 < h < \texttt{depth}(\mathcal{T})$, then such nodes are not necessarily $\varepsilon(h)$-optimal. In other words, only a portion of the p expansions is dedicated $\cup_{h<\texttt{depth}(\mathcal{T})}\mathcal{I}_h^{\varepsilon(h)}$. This depends on $\mathcal{A}$'s expansion strategy.

While the second factor is $\mathcal{A}$-dependent, the first depends on f, ℓ, and how the nodes are shaped. The first factor is discussed in the next section. In Theorem 1 of section 3.3.1.4, we shall demonstrate how these two factors play a role in identifying the regret bound for a class of MSO algorithm.

Remark 4 (Another bound for $r(p)$)**.** *The condition* (3.31) *gives us another bound on the regret:*

$$\begin{aligned} r(p) &\leq 2\cdot\eta + \lambda\cdot\zeta + \delta(h)\ \forall h \in 0,\dots,h_p^{\varepsilon(h)} && (3.33)\\ &\leq 2\cdot\eta + \lambda\cdot\zeta + \delta(h_p^{\varepsilon(h)}) && (3.34)\end{aligned}$$

Note that $h_p^* \geq \hat{h}$ implies $h_p^{\varepsilon(h)} \geq \hat{h}$.

3.3.1.3 Bounding $|\mathcal{I}_h^{\varepsilon(h)}|$

Consider the ϵ-optimal states $\mathcal{X}_\epsilon$ in an n-dimensional space. Assume that $\acute{\ell}(\mathbf{x}^*,\mathbf{x}) \leq f(\mathbf{x}^*) - f(\mathbf{x})$ where $\acute{\ell}(\mathbf{x}^*,\mathbf{x}) = \acute{L}||\mathbf{x}^*-\mathbf{x}||^{\beta}$, $\acute{L}$ and β are positive real constants.[7]

[7]The purpose of presenting $\acute{\ell}$ is to quantify how big $\mathcal{X}_\epsilon$ is, and consequently to introduce the near-optimality dimension (presented shortly). It does not always hold. Consider a constant function (e.g., $f(\mathbf{x}) = 5$), for which $\acute{\ell}(\mathbf{x}^*,\mathbf{x})$ should be 0. This violates the definition of $\acute{\ell}$ with $\acute{L}$ being a positive real constant. Nevertheless, it implies that $\ell(\mathbf{x}^*,\mathbf{x}) = 0 \leq \epsilon$, for all $\mathbf{x} \in \mathcal{X}$ and hence $\mathcal{X}_\epsilon = \mathcal{X}$. In other words, all the nodes at all depths have to be expanded to find the optimal node, yet each one of them is an optimal node. In summary, if $\acute{\ell}$ does not exist, the whole search space is considered as ϵ-optimal. Note that Assumption 1 implies $\beta \geq \alpha$.

From Definition 15, $\acute{\ell}(\mathbf{x}^*, \mathbf{x}) \leq \epsilon$, $\mathcal{X}_\epsilon$ is then an $\acute{\ell}$-hypersphere of radius ϵ centered at $\mathbf{x}^*$. From Definition 16, $\mathcal{I}_h^\epsilon$ have their base points $\mathbf{x}_{h,i}$ in $\mathcal{X}_\epsilon$. Since $\mathbf{x}_{h,i}$ can be anywhere in $\mathcal{X}_{h,i}$, covering $\mathcal{X}_\epsilon$ by $\mathcal{I}_h^\epsilon$ is regarded as a packing problem of $\mathcal{I}_h^\epsilon$ into $\mathcal{X}_\epsilon$. Note that, in general, cells of $\mathcal{I}_h^\epsilon$ will not be spheres. A bound on $|\mathcal{I}_h^\epsilon|$ is formulated as the ratio of $\mathcal{X}_\epsilon$'s volume to the smallest volume among the cells of $\mathcal{I}_h^\epsilon$:

$$
\begin{aligned}
|\mathcal{I}_h^\epsilon| &\leq \frac{\sigma(\mathcal{X}_\epsilon)}{\min_{(h,i)\in\mathcal{I}_h^\epsilon} \sigma(\mathcal{X}_{h,i})} && (3.35)\\
&= \mathcal{O}\left(\frac{\epsilon^{\frac{n}{\beta}}}{\min_{(h,i)\in\mathcal{I}_h^\epsilon} \sigma(\mathcal{X}_{h,i})}\right) . && (3.36)
\end{aligned}
$$

To simplify the bound further, we make an assumption about the cells shape in line with Assumption 2.

Assumption 3 (Cells Sphericity)**.** *There exists $1 \geq v > 0$ such that for any $h \in \mathbb{Z}_0^+$, any cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $v\delta(h)$.*

Remark 5. *The purpose of Assumption 3 is to fit a $v\delta(h)$-radius ℓ-hypersphere within cells of depth h, and hence the size of $\mathcal{I}_h^{\delta(h)}$ can be bounded more accurately. This depends on the hierarchical partitioning of the algorithm rather than the problem and holds seamlessly if the algorithm does not skew the shape of the cells. Almost all MSO algorithms maintain a coordinate-wise partition that does not skew the shape of their cells, and therefore, it is a reasonable assumption.*

Cells sphericity lets us have an explicit bound on the number of $\delta(h)$-optimal nodes at depth h, $|\mathcal{I}_h^{\delta(h)}|$- i.e., the number of nodes that cover the $\acute{\ell}$-hypersphere of radius $\delta(h)$ at depth h. Subsequently, $|\mathcal{I}_h^{\delta(h)}|$ has a bound of:

$$
\begin{aligned}
|\mathcal{I}_h^{\delta(h)}| &= \mathcal{O}\left(\frac{\delta(h)^{\frac{n}{\beta}}}{\delta(h)^{\frac{n}{\alpha}}}\right) && (3.37)\\
&= \mathcal{O}\left(\rho^{-h\left(n(\frac{1}{\alpha}-\frac{1}{\beta})\right)}\right) && (3.38)
\end{aligned}
$$

which comes in one of three cases:

1. $\alpha < \beta$: At a given depth h, $|\mathcal{I}_h^{\delta(h)}|$ scales exponentially with n.

2. $\alpha = \beta$: At a given depth h, $|\mathcal{I}_h^{\delta(h)}|$ is constant and independent of n.

3. $\alpha > \beta$: This is not possible as it violates Assumption 1. Nevertheless, it tells us that if ℓ was chosen such that there is no guarantee that $f(\mathbf{x}^*) - f(\mathbf{x}) \leq \ell(\mathbf{x}^*, \mathbf{x})$, then it is possible that $\mathcal{A}$ may expand nodes other $\delta(h)$-optimal nodes at depth h and possibly converges to a local optimum. In such case, $r(p)$ is of $\mathcal{O}(1)$.

Let d_v be $n(\frac{1}{\alpha} - \frac{1}{\beta})$. From (3.38), d_v can be seen as a quantitative measure of how much exploration is needed to guarantee optimality. For better understanding, Figure 3.3 shows $\mathcal{X}_\epsilon$ for $n = 2$; it can be packed with $\left(\frac{\epsilon^{\frac{1}{\beta}}}{(v\epsilon)^{\frac{1}{\alpha}}}\right)^2 = v^{-\frac{2}{\alpha}}$ ℓ-circles of radius $v\epsilon$. In such case, d_v is 0.

In accordance with the Assumptions 1, 2, and 3, the next definition generalizes d_v and refers to it as the *near-optimality dimension*:

Definition 18. *The v-**near-optimality dimension** is the smallest $d_v \geq 0$ such that there exists $C > 0$ and $c > 0$ such that for any $1 \geq \epsilon > 0$ and $1 \geq v > 0$, the maximal number of disjoint ℓ-hyperspheres of radius $vc\epsilon$ and contained in $\mathcal{X}_{c\epsilon}$ is less than $C\epsilon^{-d_v}$.*

The next lemma reformulates and generalizes the bound of (3.38) in the light of the near-optimality definition.

Lemma 1. $|\mathcal{I}_h^{m\delta(h)}| \leq C\rho^{-hd_{v/m}}$*, where $m > 0$.*

Proof. From Assumption 3, a cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $\frac{v}{m} \cdot m\delta(h)$. As a result, if $|\mathcal{I}_h^{m\delta(h)}|$ exceeds $C\rho^{-hd_{v/m}}$, then there are more than $C\rho^{-hd_{v/m}}$ ℓ-hypersphere of radius $\frac{v}{m} \cdot m\delta(h)$ in $\mathcal{X}_{m\delta(h)}$ which contradicts the definition of $d_{v/m}$. □

Using the above lemma, the bound of (3.38) can be reformulated as $|\mathcal{I}_h^{\delta(h)}| \leq C\rho^{-hd_v}$. Having constructed an explicit bound on $|\mathcal{I}_h^{\delta(h)}|$, let us bound $|\mathcal{I}_h^{\varepsilon(h)}|$ formally in the next lemma.

Lemma 2. *For $h < h_\varepsilon$, we have $|\mathcal{I}_h^{\varepsilon(h)}| \leq C\rho^{-hd_{v/2}}$ such that*

$$h_\varepsilon \overset{\text{def}}{=} \min\{h : \delta(h) < 2 \cdot \eta + \lambda \cdot \zeta\} \tag{3.39}$$

and $d_{v/2}$ is the $v/2$-near-optimality dimension.

Proof. Bounding $|\mathcal{I}_h^{\varepsilon(h)}|$ in a similar way to $|\mathcal{I}_h^{\delta(h)}|$ requires that a cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $\acute{v}(\delta(h)+2\cdot\eta+\lambda\cdot\zeta)$ where $1\geq\acute{v}>0$. From Assumption 3, we know that a cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $v\delta(h)$ where $1\geq v>0$. Thus, for a cell $\mathcal{X}_{h,i}$ to have an ℓ-hypersphere of radius $\acute{v}(\delta(h)+2\cdot\eta+\lambda\cdot\zeta)$, we need to ensure that $\acute{v}(\delta(h)+2\cdot\eta+\lambda\cdot\zeta)\leq v\delta(h)$. With respect to a $v/2$-near-optimality dimension, this holds for any depth h where $\delta(h)\geq 2\cdot\eta+\lambda\cdot\zeta$ because $\frac{v}{2}\cdot 2\cdot\delta(h)\geq\acute{v}(\delta(h)+2\cdot\eta+\lambda\cdot\zeta)$ where $\acute{v}=\frac{v}{2}$. Now, since $\delta(h)$ is a decreasing sequence in h, this is valid for depths less than $\min\{h:\delta(h)<2\cdot\eta+\lambda\cdot\zeta\}$ denoted by h_ε.

Up to this point, we know that $|\mathcal{I}_h^{\varepsilon(h)}|$ for $h<h_\varepsilon$ is upper-bounded by the maximal number of disjoint ℓ-hypersphere of $\frac{v}{2}\cdot 2\cdot\delta(h)$ packed in $\mathcal{X}_{2\delta(h)}$. Consequently, from Definition 18 and similar to the proof of Lemma 1, we have:

$$|\mathcal{I}_h^{\varepsilon(h)}|\leq C\rho^{-hd_{v/2}},\quad\forall h<h_\varepsilon \tag{3.40}$$

where $d_{v/2}$ is the $v/2$-near-optimality dimension. □

3.3.1.4 A Convergence Theorem

In this section, we present a theorem on the convergence of a class of MSO algorithms that adopt an expansion strategy of minimizing their trees' depths by connecting $r(p)$, h_p^*, $h_p^{\varepsilon(h)}$, and $\mathcal{I}_h^{\varepsilon(h)}$. Afterwards, three examples are worked out to see the effect of some parameters on the complexity of $r(p)$.

Theorem 1. *Let the assumptions of local Hölder continuity (Assumption 1), bounded cells (Assumption 2), and cells sphericity (Assumption 3) hold and let $\mathcal{A}$ be an MSO algorithm with a partition factor of K whose local and global score functions satisfy (3.9) and (3.10), respectively. Furthermore, $\mathcal{A}$ expands for each $\varepsilon(h)$-optimal node at most $M-1$ other nodes with an expansion strategy of minimizing its tree $\mathcal{T}$'s depth $\texttt{depth}(\mathcal{T})$.*[8] *Then the regret of $\mathcal{A}$ after p expansions is bounded as:*

$$r(p)\leq\delta(\max(\acute{h}(p),\min(h(p),h_\varepsilon)))+2\cdot\eta \tag{3.41}$$

[8]The purpose of Theorem 1 is to provide a recipe for finite-time analysis of any proposed algorithm under MSO framework. Nonetheless, the recipe needs to know the behavior of the algorithm. As a possible behavior, we have just assumed that the algorithm, analyzed in Theorem 1, minimizes its depth.

where $\acute{h}(p)$ is the smallest $h \in \mathbb{Z}_0^+$, such that:

$$M \sum_{l=0}^{\acute{h}(p)} K^l \geq p \tag{3.42}$$

and $h(p)$ is the smallest $h \in \mathbb{Z}_0^+$, such that:

$$CM \sum_{l=0}^{h(p)} \rho^{-ld_v/2} \geq p \tag{3.43}$$

and h_ε is the smallest $h \in \mathbb{Z}_0^+$ that satisfies (3.39).

Proof. Consider any $h(p) \leq h_\varepsilon$. From the definition of $h(p)$ (3.43) and Lemma 2, we have:

$$M \sum_{l=0}^{h(p)-1} |\mathcal{I}_l^{\varepsilon(l)}| \leq CM \sum_{l=0}^{h(p)-1} \rho^{-ld_v/2} < p \tag{3.44}$$

Since for each $\varepsilon(h)$-optimal node at most M other nodes are expanded in a way that minimizes the tree's depth, we deduce from (3.44) that the depth of deepest expanded optimal node $h_p^* \geq h(p) - 1$ and the depth of the deepest expanded $\varepsilon(h)$-optimal node $h_p^{\varepsilon(h)} \geq h(p)$. On the other hand, for $h(p) > h_\varepsilon$, there is no valid bound on $|\mathcal{I}_l^{\varepsilon(l)}|$. Thus, $h_p^* \geq \min(h_\varepsilon, h(p)) - 1$ and $h_p^{\varepsilon(h)} \geq \min(h_\varepsilon, h(p))$. With h_p^* and $h_p^{\varepsilon(h)}$ at hand, we have two bounds on $r(p)$. Consider first the bound based on h_p^*. Let (h^*, i^*) be the child node of the deepest expanded optimal node that contains $\mathbf{x}^*$, we have:

$$r(p) \leq f^* - f(\mathbf{x}_{h^*,i^*}) \tag{3.45}$$

Now, since $\mathcal{A}$ deals with the approximate l (3.9) of f values at the nodes' base points, the regret bound, with respect to $\mathcal{A}$'s solution $l(\mathbf{x}(p))$, is expressed as:

$$r(p) \leq f^* - f(\mathbf{x}(p)) + \eta \qquad \text{since } |l(\mathbf{x}(p)) - f(\mathbf{x}(p))| \leq \eta\ , \tag{3.46}$$

$$\leq f^* - f(\mathbf{x}_{h^*,i^*}) + 2 \cdot \eta \qquad \text{since } l(\mathbf{x}(p)) \geq l(x_{h^*,i^*})\ . \tag{3.47}$$

From Assumptions 1 & 2 and since $h^* = h_p^* + 1 \geq \min(h_\varepsilon, h(p))$, we have:

$$r(p) \leq \delta(\min(h_\varepsilon, h(p))) + 2 \cdot \eta \tag{3.48}$$

On the other hand, consider the bound based on $h_p^{\varepsilon(h)} \geq \min(h_\varepsilon, h(p))$. From Remark 4, the regret is bounded as $r(p) \leq 2 \cdot \eta + \lambda \cdot \zeta + \delta(\min(h_\varepsilon, h(p)))$. Clearly, the bound (3.48) is tighter. However, it relies on h_ε and hence can be really pessimistic (e.g. when $h_\varepsilon = 0$). We may achieve a better bound by utilizing the fact that $\mathcal{A}$'s strategy is to minimize its tree's depth; and that for p expansions, there are at least $\lfloor \frac{n}{M} \rfloor$ expanded $\varepsilon(h)$-optimal nodes. From the definition of $\acute{h}(p)$ and $\mathcal{A}$'s strategy, we deduce that $h_p^* \geq \acute{h}(p) - 1$ and $h_p^{\varepsilon(h)} \geq \acute{h}(p)$. Thus:

$$r(p) \leq \delta(\acute{h}(p)) + 2 \cdot \eta \tag{3.49}$$

Therefore, $r(p)$ here has two bounds, viz. (3.48), (3.49) from which we choose the tightest as in (3.41). □

It is important to note here that not all MSO algorithms aim to minimize their trees' depths. Nevertheless, the aim of the theorem is to stress that there are usually two possible approaches to obtain a regret bound: the first involves identifying the link between n and h_p^* (Section 3.3.1.2); and the second is based on identifying the link between n and $h_p^{\varepsilon(h)}$ (Remark 4). Furthermore, it showed that even when the two approaches are infeasible, an MSO algorithm's expansion strategy may help in establishing a better bound. The following examples evaluate the regret bound for different settings of Theorem 1's parameters.

Example 1 ($r(p)$ for $d_{v/2} = 0$, $\acute{h}(p) < \min(h(p), h_\varepsilon)$). *From (3.43), we have:*

$$\begin{aligned} CM \sum_{l=0}^{h(p)} \rho^{-ld_{v/2}} &\geq p & & \qquad (3.50) \\ CM(h(p)+1) &\geq p & \textit{since } d_{v/2} = 0 & \qquad (3.51) \\ h(p) &\geq \frac{p}{CM} - 1 & & \qquad (3.52) \end{aligned}$$

We have two cases with respect to h_ε:

1. $\eta = 0$ *and* $\zeta = 0$, *for which* $h_\varepsilon = \infty$. *Therefore,* $r(p)$ *decays exponentially with* p*:*

$$r(p) \leq c\rho^{\frac{p}{CM}} \tag{3.53}$$

2. $\eta > 0$ *or* $\zeta > 0$, *for which* $h_\varepsilon = \acute{h} < \infty$. *Therefore:*

$$r(p) \leq c\rho^{\min(\frac{p}{CM}, \acute{h})} + 2 \cdot \eta \tag{3.54}$$

Clearly, for $1 \leq p \leq CM\acute{h}$, *the regret decays exponentially with* p *towards* $2 \cdot \eta$. *For* $p > CM\acute{h}$, *the best bound on* $r(p)$ *equals* $c\rho^{\acute{h}} + 2 \cdot \eta$.

Example 2 ($r(p)$ for $d_{v/2} > 0$, $\acute{h}(p) < \min(h(p), h_\varepsilon)$)**.** *From (3.43), we have:*

$$CM \sum_{l=0}^{h(p)} \rho^{-l d_{v/2}} \geq p \tag{3.55}$$

From the sum of geometric series formula:

$$CM(\rho^{-d_{v/2}(h(p)+1)} - 1) \geq p(\rho^{-d_{v/2}} - 1) \tag{3.56}$$

$$\rho^{-d_{v/2}h(p)} > \frac{p}{CM}(1 - \rho^{d_{v/2}}) \tag{3.57}$$

$$\rho^{h(p)} < \left(\frac{CM}{1 - \rho^{d_{v/2}}}\right)^{\frac{1}{d_{v/2}}} \cdot p^{-\frac{1}{d_{v/2}}} \tag{3.58}$$

$h(p)$ is logarithmic with p. We have two cases with respect to h_ε:

1. *$\eta = 0$ and $\zeta = 0$, for which $h_\varepsilon = \infty$. Therefore, $r(p)$ decays polynomially with p:*

$$r(p) < c\left(\frac{CM}{1 - \rho^{d_{v/2}}}\right)^{\frac{1}{d_{v/2}}} \cdot p^{-\frac{1}{d_{v/2}}} \tag{3.59}$$

2. *$\eta > 0$ or $\zeta > 0$, for which $h_\varepsilon = \acute{h} < \infty$. Therefore:*

$$r(p) \leq c \cdot \max(\rho^{\acute{h}}, \left(\frac{CM}{1 - \rho^{d_{v/2}}}\right)^{\frac{1}{d_{v/2}}} \cdot p^{-\frac{1}{d_{v/2}}}) + 2 \cdot \eta \tag{3.60}$$

Example 3 ($r(p)$ for $\acute{h}(p) > \min(h(p), h_\varepsilon)$)**.** *From (3.42), we have:*

$$M \sum_{l=0}^{\acute{h}(p)} K^l \geq p \tag{3.61}$$

$$K^{\acute{h}(p)+1} \geq \frac{p}{M}(K - 1) + 1 \tag{3.62}$$

$$\acute{h}(p) \geq \lceil \log_K(\frac{p}{M}) + \log_K(K - 1) - 1 \rceil \tag{3.63}$$

$\acute{h}(p)$ is logarithmic with p and $r(p)$ is bounded as:

$$r(p) \leq c\rho^{\lceil \log_K(\frac{p}{M}) + \log_K(K-1) - 1 \rceil} + 2 * \eta \tag{3.64}$$

Remark 6 (MSO vs. Uniform Sampling)**.** *It is interesting to note that for an n-dimensional function where $|f^* - f(\boldsymbol{x})| = ||\boldsymbol{x}^* - \boldsymbol{x}||^\beta$, a uniform grid of Kp samples exhibits a polynomially decaying regret of $\mathcal{O}(p^{-\frac{\beta}{n}})$ whereas an MSO algorithm with a partition factor K and p expansions may have an exponentially decaying regret of $\mathcal{O}(\rho^{\frac{p}{CM}})$ (Example 1) irrespective of the problem dimensions n.*

3.3.2 Analysis of MSO Algorithms

Using the theoretical framework established in the previous section, we analyze the finite-time behavior of different MSO algorithms in the literature.

3.3.2.1 Analysis of `LO`

Several researchers have addressed the convergence analysis of `LO` techniques [33, 137]. Here, we present a complimentary analysis under the framework of MSO. Let $n = 1$ which implies that $\mathcal{X}_{h,i}$ is an interval $[c_{h,i}, d_{h,i}]$ where $c_{h,i}$ and $d_{h,i}$ are its endpoints. Furthermore, $\ell(\mathbf{x}, \mathbf{y})$ here is $L|x-y|$. Using the Lipschitz condition (2.3), $\hat{f}(\mathbf{x}_{h,i})$—and eventually $b_{(h,i)}$—is computed as:

$$b_{(h,i)} = \hat{f}(\mathbf{x}_{h,i}) = \frac{f(c_{h,i}) + f(d_{h,i})}{2} + L\frac{d_{h,i} - c_{h,i}}{2} \tag{3.65}$$

This can be made equivalent to the exploitation-exploration trade-off (3.11) where the local score $l_{h,i} = \frac{f(c_{h,i})+f(d_{h,i})}{2}$, $\lambda = L$, and the global score $g_{h,i} = \frac{d_{h,i}-c_{h,i}}{2}$. From the Lipschitz condition (2.3) and Assumption 2, we have:

$$0 \leq \hat{f}(\mathbf{x}_{h,i}) - f(\mathbf{x}_{h,i}) \leq L(d_{h,i} - c_{h,i}) \leq \delta(h)\ . \tag{3.66}$$

The next lemma shows that `LO` expands $2\delta(h)$-optimal nodes in search for the optimal node at depth h.

Lemma 3. *In `LO` and at depth h, a node (h, i) is expanded before the optimal node (h^*, i^*), if:*

$$f(\boldsymbol{x}_{h,i}) + 2\delta(h) \geq f^*\ . \tag{3.67}$$

Proof. In `LO`, expanding more than one node per depth is possible; a node (h, i) is expanded before the optimal node (h^*, i^*) when $f(\mathbf{x}_{h,i})$ satisfies the following:

$$b_{(h,i)} \geq b_{(h^*,i^*)} \tag{3.68}$$

$$\hat{f}(\mathbf{x}_{h,i}) \geq \hat{f}(x_{h^*,i^*}) \tag{3.69}$$

From (3.66):

$$f(\mathbf{x}_{h,i}) + \delta(h) \geq f(\mathbf{x}_{h^*,i^*}) \tag{3.70}$$

From the Lipschitz condition (2.3) which is in line with Assumption 1 and Assumption 2, we have $f^* - f(\mathbf{x}_{h^*,i^*}) \leq \sup_{\mathbf{x} \in \mathcal{X}_{h,i}} \ell(\mathbf{x}_{h,i}, \mathbf{x}) \leq \delta(h)$; from which (3.67) follows. □

Therefore, we are certain that $h_p^* \geq h$ if all the $2\delta(h)$-optimal nodes at depth h are expanded. The following theorem establishes the regret bound for `LO` algorithms.

Theorem 2. *(Convergence of `LO`) Let $\mathcal{T}$ be `LO`'s tree on $\mathcal{X}$ and $h(p)$ be the smallest $h \in \mathbb{Z}_0^+$, such that:*

$$C \sum_{l=0}^{h(p)} \rho^{-l d_{v/2}} \geq p \tag{3.71}$$

where $d_{v/2}$ is the $v/2$-near-optimality dimension and p is the number of expansions. Then the regret of `LO` is bounded as $r(p) \leq 2\delta(h(p))$.

Proof. From Lemma 3, `LO` expands only nodes that are $2\delta(h)$-optimal for $0 \leq h \leq \texttt{depth}(\mathcal{T})$. As a result, the depth of deepest expanded $2\delta(h)$-optimal node $h_p^{2\delta(h)}$ is $\texttt{depth}(\mathcal{T}) - 1$ and hence bounding the regret in the light of Remark 4 as follows:

$$r(p) \leq 2\delta(\texttt{depth}(\mathcal{T}) - 1) \tag{3.72}$$

Clearly, the bound depends on $\texttt{depth}(\mathcal{T})$. Let us try to make this bound more explicit by finding out the minimum $\texttt{depth}(\mathcal{T})$ with p expansions. In line with Lemma 1, we have $|\mathcal{I}_h^{2\delta(h)}| \leq C\rho^{-hd_{v/2}}$. This implies along with the definition of $h(p)$:

$$\sum_{l=0}^{h(p)-1} |\mathcal{I}_l^{2\delta(l)}| \leq C \sum_{l=0}^{h(p)-1} \rho^{-l d_{v/2}} < p \tag{3.73}$$

that $h_p^{2\delta(h)} \geq h(p)$. Therefore, $r(p) \leq 2\delta(h(p))$. □

Similar analysis can be applied when $n > 1$. Here, $\mathcal{X}_{h,i}$ is a hyperrectangle and $\delta(h)$ bounds a norm $L||\mathbf{x} - \mathbf{y}||$ rather than an interval.

3.3.2.2 Analysis of `DOO`

With respect to the theoretical framework, [34] provides the analysis of `DOO` for $\delta(h) < 1$. Here, we provide a generalized analysis (including $\delta(h) \geq 1$) by modifying [34, Theorem 1]. Let us start with the b-value of a node (h, i):

$$b_{(h,i)} = f(\mathbf{x}_{h,i}) + \delta(h) \tag{3.74}$$

With reference to (3.11), $l_{h,i} = f(\mathbf{x}_{h,i})$, $\lambda = 1$, and $g_{h,i} = \delta(h)$. The next lemma shows that `DOO` expands $\delta(h)$-optimal nodes in search for the optimal node at depth h.

Lemma 4. *In* `DOO` *and at depth h, a node (h, i) is expanded before the optimal node (h^*, i^*), if:*

$$f(\boldsymbol{x}_{h,i}) + \delta(h) \geq f^* \; . \tag{3.75}$$

Proof. In `DOO`, expanding more than one node per depth is possible; a node (h, i) is expanded before the optimal node (h^*, i^*) when $f(\mathbf{x}_{h,i})$ satisfies the following:

$$b_{(h,i)} \geq b_{(h^*,i^*)} \tag{3.76}$$

$$f(\mathbf{x}_{h,i}) \geq f(\mathbf{x}_{h^*,i^*}) \tag{3.77}$$

From Assumptions 1 & 2, we have $f^* - f(\mathbf{x}_{h^*,i^*}) \leq \sup_{\mathbf{x}\in\mathcal{X}_{h,i}} \ell(\mathbf{x}_{h,i}, \mathbf{x}) \leq \delta(h)$; from which (3.75) follows. □

Therefore, we are certain that $h^*_p \geq h$ if all the $\delta(h)$-optimal nodes at depth h are expanded. The following theorem establishes the regret bound for `DOO`.

Theorem 3 (Convergence for `DOO`). *Let us write $h(p)$ the smallest $h \in \mathbb{Z}^+_0$ such that*

$$C \sum_{l=0}^{h(p)} \rho^{-ld_v} \geq p \tag{3.78}$$

where d_v is the v-near-optimality dimension and p is the number of expansions. Then the regret of `DOO` *is bounded as $r(p) \leq \delta(h(p))$.*

Proof. From Lemma 4, `DOO` expands only nodes that are $\delta(h)$-optimal for $0 \leq h \leq$ `depth`$(\mathcal{T})$. As a result, the depth of deepest expanded $\delta(h)$-optimal node $h^{\delta(h)}_p$ is `depth`$(\mathcal{T}) - 1$, and hence bounding the regret in the light of Remark 4 as follows:

$$r(p) \leq \delta(\texttt{depth}(\mathcal{T}) - 1) \tag{3.79}$$

Clearly, the bound depends on `depth`$(\mathcal{T})$. Let us try to make this bound more explicit by finding out the minimum `depth`$(\mathcal{T})$ with p expansions. In line with Lemma 1, we have $|\mathcal{I}^{\delta(h)}_h| \leq C\rho^{-hd_v}$. This implies along with the definition of $h(p)$:

$$\sum_{l=0}^{h(p)-1} |\mathcal{I}^{\delta(l)}_l| \leq C \sum_{l=0}^{h(p)-1} \rho^{-ld_v} < p \tag{3.80}$$

that $h^{\delta(h)}_p \geq h(p)$. Therefore, $r(p) \leq \delta(h(p))$. □

3.3.2.3 Analysis of `DIRECT`

The b-value of a node in `DIRECT` is its local score l, with $l_{h,i} = f(\mathbf{x}_{h,i})$, $\lambda = 0$, and $g_{h,i} = \sigma_{h,i} = ||d_{h,i}||_2$ where $d_{h,i}$ is $\mathcal{X}_{h,i}$'s diameter. As $\lambda = 0$, `DIRECT` may seem to be an exploitative MSO algorithm, which is not the case. The global score is employed in a heuristic (represented by (3.13) and (3.14)) for selecting $\mathcal{Q}$ within a batch of iterations, balances the exploration-vs.-exploitation trade-off. Given that the optimal node at depth $\acute{h} - 1$ has been expanded, for any node $(\acute{h}, i)$ to be expanded before the optimal node $(\acute{h}, i^*)$, the following must hold (assuming (3.13) and (3.14) hold):

$$
\begin{aligned}
b_{(\acute{h},i)} &\geq b_{(\acute{h},i^*)} && (3.81)\\
f(\mathbf{x}_{\acute{h},i}) + \delta(\acute{h}) &\geq f^* && (3.82)
\end{aligned}
$$

For such depths, `DIRECT` expands $\delta(h)$-optimal nodes at depth h if the heuristic rules hold. However, there is no guarantee that the heuristic rules hold [36]. In [36, Theorem 2], it has been shown that `DIRECT` may behave as an exhaustive grid search expanding nodes based solely on their sizes. In Theorem 4, we provide a finite-time[9] regret bound on `DIRECT` by exploiting [36]'s findings that within a batch of iteration I_t, there exists at least one node $\in \mathcal{E}_{t-1}$ that satisfies (3.13) and (3.14) and hence gets expanded within I_t. Such node is simply the node $(h_*, i_*) \in \mathcal{E}_{t-1}$ such that $b_{(h_*,i_*)} = b_t$.

Theorem 4 (Convergence of `DIRECT`)**.** *Let us define $h(p)$ as is the smallest $h \in \mathbb{Z}_0^+$ such that:*

$$\sum_{l=0}^{h(p)} 3^l \geq n_I \tag{3.83}$$

where n_I is the greatest positive integer number such that:

$$n_I(n_I - 1) \leq \frac{2}{n^2}(p - n) \tag{3.84}$$

where p is the number of expansions. Then the regret of `DIRECT` with $Q = 1$ is bounded as:

$$r(p) \leq \delta(h(p)) \tag{3.85}$$

[9]To the best of our knowledge, there is no finite-time analysis of `DIRECT` (only the consistency property $\lim_{n\to\infty} r(p) = 0$ given by Jones *et al.* [56] which was proven again in [29] by Finkel and Kelley. Furthermore, they showed, based on non-smooth analysis, that certain samples of `DIRECT` may converge to points that satisfy the necessary conditions for optimality defined by Clarke [188]).

Proof. `DIRECT` expands at least one node per batch of iterations; and this node is one among those of the largest size among the leaf nodes. Thus, as `DIRECT` has a partitioning factor of 3; given a number of batches n_I; and from the definition of $h(p)$, at depth $h(p)-1$, all the nodes (optimal and non-optimal) are expanded. Hence, $r(p) \leq \delta(h(p))$.

Since we are interested in bounding the regret as a function of the number of expansions p, we need to find what is the maximum number of expansions for n_I iteration batches. In a given batch I_t, the maximum number of expansions with $Q=1$ is $n \cdot \texttt{depth}(\mathcal{T}_{t-1})$, with $\texttt{depth}(\mathcal{T})$ growing at most by n per iteration batch. Thus, with 3 iteration batches, for instance, we can have at most $n+1\cdot n\cdot n+2\cdot n\cdot n$ expansions. For m batch of iterations, there are at most $n+n^2\sum_{i=0}^{m-1} i = n+\frac{1}{2}\cdot n^2 \cdot m(m-1)$ expansions. Therefore, for p expansions, the minimum possible number of completed iteration batches is the greatest positive integer n_I iteration such that:

$$n + \frac{1}{2}\cdot n^2 \cdot n_I(n_I - 1) \leq p$$

from which (3.84) follows. □

3.3.2.4 Analysis of `MCS`

The following factors influence the analysis of `MCS`. First, the set of nodes to be considered for expansion are not of the same scale, and hence, no statement can be made about the optimality of the expanded nodes. Second, even if `MCS` is considering near-optimal nodes for expansion, the heuristics (3.16) and (3.17) may not hold which results in moving such nodes into groups of different pseudo-depth. Third, the fact that two nodes of consecutive depths h and $h+1$ may have the same size makes Assumption 2 more associated with the node's first pseudo-depth value rather than its depth h (i.e. for a node (h,i) whose s upon creation is $s_{h,i}$, then $\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}_{h,i}} \ell(\mathbf{x},\mathbf{y}) \leq \delta(s_{h,i})$ rather than $\delta(h)$).

The b-value of a node in `MCS` is its local score, with $l_{h,i} = f(\mathbf{x}_{h,i})$, $\lambda = 0$, and $g_{h,i} = -s_{h,i}$. The global score is used to group the nodes considered for expansion at an iteration. Assume that all the nodes keep their initial pseudo-depth; given that the optimal node at a pseudo-depth $s-1$ has been expanded, its optimal child node $(\acute{h}, i^*)$ may have a pseudo-depth of $s+1$ for which no statement can be made about the nodes with a pseudo-depth of s. Nonetheless, if $(\acute{h}, i^*)$ is at pseudo-depth s, then for any node (h,i) of the same pseudo-depth to be expanded before $(\acute{h},i^*)$, the following must hold (assuming either of the heuristics (3.16) or (3.17) holds as well):

$$b_{(h,i)} \geq b_{(\acute{h},i^*)} \tag{3.86}$$

$$f(\mathbf{x}_{h,i}) + \delta(s) \geq f^* \tag{3.87}$$

For such case, `MCS` expands $\delta(s)$-optimal nodes (with regards to `MCS`, we refer to a node that satisfies (3.87), where s is the node's initial pseudo-depth as a $\delta(s)$-optimal node, and denote the set of such nodes by $\mathcal{I}^{\delta(s)}$) and may expand non-$\delta(s)$-optimal nodes, otherwise. Clearly, the analysis is complicated. However, we can simplify it with an assumption about the structure of the maximal expected gain $\max_{1\leq j\leq n} \hat{e}_j^{(h,i)}$ for $\delta(s)$-optimal nodes. With this assumption, the relationship between h_p^* and p can be established. This is demonstrated in the next lemma.

Lemma 5. *In `MCS`, for any depth $0 \leq h < s_{max}$, whenever $n \geq \sum_{l=0}^{s} |\mathcal{I}^{\delta(l)}| (s_{max}-1)$ and $\max_{1\leq j\leq n} \hat{e}_j^{(h,i)} \geq \delta(s)$, for all $\delta(s)$-optimal node $\in \mathcal{T}$ where $0 \leq s \leq s_{max}-1$, we have $h_p^* \geq s$, where p is the number of expansions.*

Proof. We know that $h_p^* \geq 0$ and hence the above statement holds for $s = 0$. For $s \geq 1$, we are going to prove it by induction.

Assume that the statement holds for $0 \leq s \leq \hat{s}$. We prove it for $s \geq \hat{s}+1$. Let $p \geq \sum_{l=0}^{\hat{s}+1} |\mathcal{I}^{\delta(l)}|(s_{max}-1)$. Consequently, we know that $p \geq \sum_{l=0}^{\hat{s}} |\mathcal{I}^{\delta(l)}|(s_{max}-1)$ for which $h_p^* \geq \hat{s}$. Here, we have two cases: $h_p^* \geq \hat{s}+1$, for which the proof is done; or $h_p^* = \hat{s}$. In the second case, any node (h,i) at pseudo-depth $\hat{s}+1$ that is expanded before the optimal node of the same pseudo-depth has to be $\delta(\hat{s}+1)$-optimal. However, the heuristics of `MCS` may possibly cause the expansion of such nodes to be skipped and expanding at the same time non-$\delta(s)$-optimal nodes at higher pseudo-depths. Nonetheless, if the computed expected gain for a $\delta(\hat{s}+1)$-optimal node (h,i) satisfies $\max_{1\leq j\leq n} \hat{e}_j^{(h,i)} \geq \delta(\hat{s}+1)$, then we are certain that heuristic (3.17) will always hold for such nodes. This can be proved as follows. Let (h,i) be a $\delta(\hat{s}+1)$-optimal node, we have:

$$f(\mathbf{x}_{h,i}) + \delta(\hat{s}+1) \geq f^* \tag{3.88}$$

$$f(\mathbf{x}_{h,i}) + \max_{1\leq j\leq n} \hat{e}_j^{(h,i)} \geq f_{max} \tag{3.89}$$

Thus, (3.17) holds for $\delta(\hat{s}+1)$-optimal nodes and they will always get expanded. Since there are $|\mathcal{I}^{\delta(\hat{s}+1)}|$ of such nodes, we are certain that the optimal node at pseudo-depth $\hat{s}+1$ is expanded after at most $|\mathcal{I}^{\delta(\hat{s}+1)}|(s_{max}-1)$ expansions. Thus, $h^*_p \geq \hat{s}+1$. □

The next theorem builds on Lemma 5 to present a finite-time analysis of `MCS` with an assumption on the structure of a node's gain. To the best of our knowledge there is no finite-time analysis of `MCS` with/without local search (only the consistency property $\lim_{n\to\infty} r(p) = 0$ for `MCS` without local search in [27]).

Theorem 5 (Convergence of `MCS`). *Assuming* $\max_{1\leq j\leq n} \hat{e}^{(h,i)}_j \geq \delta(s)$*, for all* $\delta(s)$*-optimal node* $\in \mathcal{T}$ *where* $0 \leq s \leq s_{max}-1$*, let us write* $s(p)$ *the smallest* $s \in \mathbb{Z}^+_0$ *such that*

$$C(s_{max}-1)\sum_{l=0}^{s(p)} \rho^{-ld_v} \geq p \tag{3.90}$$

where d_v *is the* v*-near-optimality dimension. Then the regret of `MCS` without local search is bounded as*

$$r(p) \leq \delta(\min(s(p), s_{max}))$$

Proof. From Lemma 1, and the definition of $s(p)$ (3.90), we have $|\mathcal{I}^{\delta(s)}| \leq C\rho^{-sd_v}$ and:

$$\sum_{l=0}^{s(p)-1} |\mathcal{I}^{\delta(l)}|(s_{max}-1) \leq C(s_{max}-1)\sum_{l=0}^{s(p)-1} \rho^{-ld_v} < n \tag{3.91}$$

which implies from Lemma 5 and $\texttt{depth}(\mathcal{T}) \leq s_{max}$ that $h^*_p \geq \min(s(p)-1, s_{max}-1))$. Thus, from (3.27), we have $r(p) \leq \delta(\min(s(p), s_{max}))$. □

3.3.2.5 Analysis of `SOO`

The b-value of a node in `SOO` is its local score, with $l_{h,i} = f(\mathbf{x}_{h,i})$, $\lambda = 0$, and $g_{h,i} = -h$. Similar to `MCS`, the global score is used as a heuristic to filter the nodes considered for expansion at an iteration. Given that the optimal node at depth $\acute{h}-1$ has been expanded, for any node $(\acute{h}, i)$ to be expanded before the optimal node $(\acute{h}, i^*)$, the following must hold:

$$\begin{aligned} b_{(\acute{h},i)} &\geq b_{(\acute{h},i^*)} && (3.92)\\ f(\mathbf{x}_{\acute{h},i}) + \delta(\acute{h}) &\geq f^* && (3.93)\end{aligned}$$

For such depths, `SOO` may expand $\delta(h)$-optimal nodes . However, in contrary to `DOO`, `SOO` may expand non-$\delta(h)$-optimal nodes at depths $\acute{h} < h \leq \texttt{depth}(\mathcal{T})$. Nevertheless, the relationship between h^*_p and p can be established due to `SOO`'s strategy in sweeping $\mathcal{T}$. This is demonstrated in [34, Lemma 2]. With respect to the theoretical framework, [34] provides the analysis of `SOO` for $\delta(h) < 1$. We provide a generalized analysis (including $\delta(h) \geq 1$) by modifying [34, Theorem 2]:

Theorem 6 (Convergence of `SOO`)**.** *Let us write $h(p)$ the smallest $h \in \mathbb{Z}^+_0$ such that*

$$Ch_{max}(p) \sum_{l=0}^{h(p)} \rho^{-ld_v} \geq p \tag{3.94}$$

where d_v is the v-near-optimality dimension and p is the number of expansions. Then the regret of `SOO` is bounded as

$$r(p) \leq \delta(\min(h(p), h_{max}+1))$$

Proof. In line with Lemma 1, we have $|\mathcal{I}^{\delta(h)}_h| \leq C\rho^{-hd_v}$. Thus, from the definition of $h(p)$ (3.94) and [34, Lemma 2], the following:

$$h_{max} \sum_{l=0}^{h(p)-1} |\mathcal{I}^{\delta(l)}_l| \leq Ch_{max}(n) \sum_{l=0}^{h(p)-1} \rho^{-ld_v} < p$$

implies that $h^*_p \geq \min(h(p)-1, h_{max}(p))$. Thus, from (3.27), we have

$$r(p) \leq \delta(\min(h(p), h_{max}+1))$$

□

Effect of $h_{max}(p)$. `SOO` controls $\mathcal{T}$'s exploration behavior (deep vs. broad) through a single parameter, namely $h_{max}(p)$. Given a fixed budget of node expansions, greater h_{max} reduces the likelihood of expanding an optimal node in $\mathcal{T}$, and hence a greater regret bound. It is interesting to consider `SOO`'s behavior when $h_{max}(p)$ is set to ∞. Although Theorem 6 may imply that, for any n, the regret is bounded as $r(p) \leq \delta(0)$—i.e., by a constant—when $h_{max}(p) = \infty$, this is not really the case for `SOO`. When $h_{max}(p)$ is set to ∞, the regret of `SOO` is related to number of iteration batches the algorithm may have for a number of expansions p. The next corollary establishes a regret bound for `SOO` with $h_{max}(p) = \infty$. It exploits the fact that for an iteration batch I_t, the number of expansions is less than or equal $\texttt{depth}(\mathcal{T}_{t-1})$; and after each batch of iterations, $\mathcal{T}$'s depth is increased at most by one.

Corollary 1 (Convergence of `SOO` with $h_{max} = \infty$)**.** *Let us define $h(p)$ as is the smallest $h \in \mathbb{Z}_0^+$ such that:*

$$C \sum_{l=0}^{h(p)} \rho^{-ld_v} \geq n_I \tag{3.95}$$

where n_I is the greatest positive integer number such that:

$$n_I(n_I + 1) \leq 2 \cdot p \tag{3.96}$$

where p is the number of expansions. Then the regret of `SOO` with $h_{max} = \infty$ is bounded as:

$$r(p) \leq \delta(h(p)) \tag{3.97}$$

Proof. Let $h_p^* = \acute{h}$ after $\acute{n}$ complete batches of iteration. Then, each of the next batches expands a $\delta(\acute{h}+1)$-optimal node (if any). Since there are $|\mathcal{I}_h^{\delta(\acute{h}+1)}| \leq C\rho^{-(\acute{h}+1)d_v}$ of such nodes, we know, after at most $\acute{n} + C\rho^{-(\acute{h}+1)d_v}$ batches, that $h_p^* \geq \acute{h} + 1$. Now, for m batches of iterations, `SOO` can have at most $\frac{m(m+1)}{2}$ expansions. Thus, from the definition of $h(p)$ and (3.27), we have $h_p^* \geq h(p) - 1$ and $r(p) \leq \delta(h(p))$, respectively. $\square$

3.4 Discussion

This section presents an outline of the theoretical findings and complements it with optimization problems in practice. Table 3.1 summarizes the convergence rate in terms of the regret bound for the algorithms discussed. These bounds do not imply a comparative performance, but rather report their worst-case behavior. Each algorithm employs different partitioning rules for which $\delta(h)$ can be different. Nevertheless, since `LO` and `DOO` are theoretical propositions, one could comment on their comparative performance. Based on Table 3.1, we can conclude the following:

1. While `LO` and `DOO` assumes the knowledge about f's smoothness; for a Lipschitz-continuous function, `DOO` is more preferable than `LO` as the latter's regret bound is double the former's for the same number of expansions, provided

that $d_v = d_{v/2}$, not to mention that DOO comes with a less restrictive assumption of the function smoothness. In practice, both algorithms are inapplicable unless some approximations on the function smoothness are made.

2. If $s_{max} = h_{max} + 1$, then SOO and MCS without local search, following the same partitioning rule, share the same regret bound, under the assumption $\max_{1\leq j\leq n} \hat{e}_j^{(h,i)} \geq \delta(s)$, for all $\delta(s)$-optimal node $\in \mathcal{T}$.

3. DIRECT has the most over-pessimistic regret bound requiring a number of expansions n that grows quadratically in the number of problem dimensions n.

Table 3.1: Convergence rate of different MSO algorithms. These rates hold provided that Assumptions 1, 2, & 3 are satisfied. The *Condition* column provides the relation between the number of expansions p and the depth $h(p)/s(p)$ besides other algorithm-specific conditions. h_{max} and s_{max} define the maximum depth for the trees of SOO and MCS, respectively.

Algorithm	**Convergence Rate**	**Condition**
LO	$r(p) \leq 2\delta(h(p))$	$C\sum_{l=0}^{h(p)} \rho^{-ld_{v/2}} \geq p$
DOO	$r(p) \leq \delta(h(p))$	$C\sum_{l=0}^{h(p)} \rho^{-ld_v} \geq p$
DIRECT	$r(p) \leq \delta(h(p))$	$\sum\limits_{l=0}^{h(p)} 3^l \geq n_I$, $n_I(n_I-1) \leq \frac{2}{n^2}(p-n)$, $Q=1$
SOO	$r(p) \leq \delta(\min(h(p), h_{max}(p)+1))$	$Ch_{max}(p)\sum_{l=0}^{h(p)} \rho^{-ld_v} \geq p$
MCS	$r(p) \leq \delta(\min(s(p), s_{max}))$	$C(s_{max}-1)\sum_{l=0}^{s(p)} \rho^{-ld_v} \geq p$, $\max_{1\leq j\leq n} \hat{e}_j^x \geq \delta(s), \forall x \in \mathcal{I}^{\delta(s)}\ 0 \leq s < s_{max}$

A Worked Example. Here, we present a worked example, where the loss measure bounds are computed using *symbolic maths* and compared with respect to the empirical results of the algorithms: DIRECT, SOO, and MCS.[10] Consider the function to be minimized $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R} = ||\mathbf{x}-\mathbf{x}^*||_\infty^\alpha$ over the decision space $\mathcal{X} = [-1,1]^n$, where $n = 1$. As these algorithms evaluate their cells at the center, the decreasing sequence $\delta(h)$ can be defined as $K^{-3\alpha\lfloor h/n \rfloor}$ with a partitioning factor of $K = 3$ and $v = 1/2$. As shown in Figure 3.4, the bounds of SOO and MCS are tightly following their empirical regrets. On the other hand, DIRECT's pessimistic behavior makes the loss bound lagging behind.

[10]LO and DOO are hypothetical propositions for which only practical/adapted implementation exists.

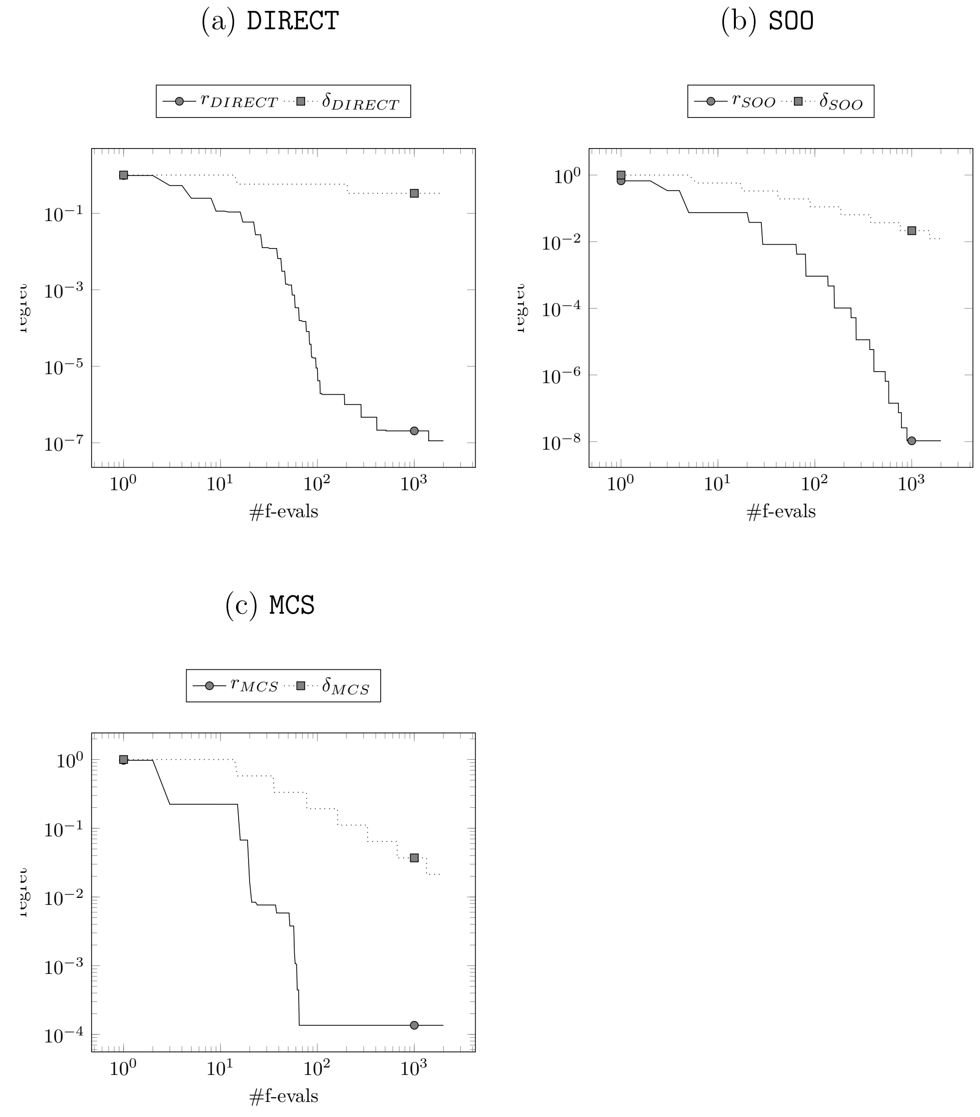


Figure 3.4: The empirical convergence rate and its theoretical bound with respect to the number of function evaluations `#f-evals` for the algorithms `DIRECT` (a), `SOO` (b), and `MCS` (c) in minimizing $||\mathbf{x} - \mathbf{x}^*||_\infty^\alpha$, with $n = 1$.

3.5 Conclusions

This chapter has consolidated a broad category of algorithms that search for the (or one) optimal solution $\mathbf{x}^*$ by partitioning the search space $\mathcal{X}$ over multiple scales for solving black-box global optimization problems, under the Multi-scale Search Optimization (MSO) framework. In line with MSO, a theoretical methodology has been presented to analyze these algorithms based on three basic assumptions: i). local Hölder continuity of the objective function f; ii). partitions boundedness;

and iii). partitions sphericity. As a result, we are able to provide a theoretical bound on the regret of several state-of-the-art MSO algorithms, including `LO`, `DOO`, `DIRECT`, `MCS` and `SOO`.

Chapter 4

Naive Multi-Scale Optimization

"Be Fearful When Others Are Greedy and Greedy When Others Are Fearful"

- Warren Buffett

The scope of this chapter is expensive optimization, that is to say find a near-optimal solution given a limited number of function evaluations. In such settings, it is likely that acting optimally according to current knowledge (exploitation) is more rewarding than learning more about the decision space (exploration). While MSO algorithms are inherently exploratory due to their systematic sampling [36], attempts have been made to couple them with local search (e.g., `MCS` by Huyer and Neumaier [27]). This chapter aims to incorporate the local search into the MSO framework rather than being a separate procedure. To this end, we propose an algorithm within the framework of Multi-scale Search Optimization (MSO) and refer to it as the Naive Multi-Scale Optimization (`NMSO`) algorithm. Similar to other MSO methods, `NMSO` creates partitions of $\mathcal{X}$ at multiple scales. It expands its $\mathcal{X}$-partitioning tree depth-wise, with the following policy: *exploit until no further improvement is observed.* When compared with other MSO algorithms, `NMSO`'s exploitative search component is more dominant than its exploratory one. `NMSO`'s theoretical finite-time and asymptotic analysis is provided in line with the presented methodology in Chapter 3.

In real-world problems, solving a problem with an acceptable degree of accuracy is adversely impacted by the evaluation budget (number of function evaluations). For instance, an evaluation of one candidate solution might take a whole laboratory experiment (e.g., [13, 16]). Although `NMSO` asymptotically converges to the optimal solution; given a limited evaluation budget, it may only find a local optimal solution due to its exploitative nature. With this regard, the search trade-off has to reduce the exploitation dominance to increase the chances of hitting the optimal solution. In order to handle limited/expensive-budget settings, NMSO can delay/push further exploration of a well-explored region after exploring less-explored regions. The efficacy of NMSO for both expensive- and cheap-budget settings is demonstrated on the noiseless black-box optimization benchmarking (BBOB) testbed. Thus, our contribution here is of two folds:

1. NMSO: an algorithm with a finite-time and asymptotic performance, sitting at the exploitation end of exploration-vs-exploitation dilemma. NMSO's analysis provides the basis for analyzing a class of MSO optimization algorithms that are exploitation-biased.

2. A thorough empirical analysis and comparison of the leading MSO optimization algorithms on the noiseless BBOB testbed under both expensive- and cheap-budget settings.

The chapter is organized as follows. Section 4.1 presents NMSO as an exploitation-biased partition algorithm for solving the black-box optimization problem (2.1) and provides a theoretical finite-time and asymptotic analysis on its convergence. An empirical assessment of NMSO on the noiseless BBOB testbed is discussed and compared with other MSO optimization algorithms in Section 4.2. Furthermore, NMSO is also compared with the state-of-the-art stochastic algorithms as a supplement work. The standing of NMSO in the black box optimization competition (BBComp) within the GECCO'15 conference is shown and discussed in Section 4.3, supporting the suitability of NMSO for optimization problems with expensive-budget settings. Finally, Section 4.4 concludes the chapter.

4.1 Naive Multi-scale Search Optimization (NMSO)

In this section, we present and analyze a Multi-scale Search Optimization (MSO) algorithm that satisfies both finite-time convergence and consistency property. We refer to it as the Naive Multi-scale Search Optimization (NMSO) algorithm.

4.1.1 The NMSO Algorithm

Similar to other MSO algorithms, NMSO is based on a hierarchical partitioning of the search space $\mathcal{X}$ with the representative state $\mathbf{x}_{h,i}$ of a node (h,i) being the center of $\mathcal{X}_{h,i}$. However, NMSO is more exploitative in its exploration-vs.-exploitation trade-off. The partitioning of (h,i) is carried out by splitting its cell $\mathcal{X}_{h,i}$ along the $(h \mod n+1)^{\text{th}}$ dimension into K cells corresponding to K child nodes. NMSO's bias towards exploitation is demonstrated by its expanding $\mathcal{T}$ at most one node (h,i) per depth h whose $f(\mathbf{x}_{h,i})$ is among the minimum of $f(\mathbf{x}_{h,j})_{0\leq j\leq K^h-1}$, starting from the root until *no further improvement is expected.* Intuitively, no further improvement in the quality of solution within the subspace $\mathcal{X}_{h,i}$ is expected if f's rate of change around $\mathcal{X}_{h,i}$ is small, and if $\mathcal{X}_{h,i}$ is a tiny portion of the search space. This is simulated by imposing some conditions (rules) on the node (h,i). If these conditions are met, NMSO assumes that no better solution can be found within $\mathcal{X}_{h,i}$ other than those found and starts, from the root node $(0,0)$, a new sequence of depth-wise expansions.

Denote by $\triangle f_{h,i}$ the finite difference of f along the $(h \mod n+1)^{\text{th}}$ dimension at $\mathcal{X}_{h,i}$ computed as:

$$\triangle f_{h,i} = f(\mathbf{x}_{h+1,iK+\lceil K/2\rceil}) - f(\mathbf{x}_{h+1,iK+\lfloor K/2\rfloor-1}) \tag{4.1}$$

and denote by $\triangle \mathbf{x}_{h,i}$ the ℓ_1-norm:

$$\triangle \mathbf{x}_{h,i} = ||\mathbf{x}_{h+1,iK+\lceil K/2\rceil} - \mathbf{x}_{h+1,iK+\lfloor K/2\rfloor-1}||_1 \tag{4.2}$$

Where $\lceil . \rceil$ and $\lfloor . \rfloor$ denote the ceiling and floor functions, respectively. NMSO decides whether to start a new sequence of expansions or continue its current one through the following rule: if the depth of the last expanded node (h,i) is of the form $mn-1$ where $m \in \mathbb{N}_1$, and

$$\nabla^f_{h,i} = max_{(\hat{h},\hat{i})\in\mathcal{S}_{(h,i)}} \triangle f_{\hat{h},\hat{i}} \;\leq\; \alpha \tag{4.3}$$

$$\nabla^{\mathbf{x}}_{h,i} = max_{(\hat{h},\hat{i})\in\mathcal{S}_{(h,i)}} \triangle \mathbf{x}_{\hat{h},\hat{i}} \;\leq\; \beta \tag{4.4}$$

then NMSO starts a new sequence of depth-wise expansions (starting from depth 0), otherwise it continues its current sequence (continuing from depth $h+1$); where α and β are arbitrarily small non-negative constants, and $\mathcal{S}_{(h,i)}$ is the set of nodes comprising of the last expanded node (h,i) and its $n-1$ last ancestors. In other words, the algorithm checks for potential improvement in the solution quality within the cell of the last expanded node, by approximating f gradients along the n dimensions. If these approximations are within a certain value, NMSO assumes no further improvement is possible and starts a new sequence of expansions of the tree from its root.

In an expansion sequence, NMSO explores new regions of the search space. Nonetheless, it is possible that the algorithm considers expanding a node whose parent node was the last expanded node in one of the previous expansion sequences. Let us denote the set of such nodes by $\mathcal{B}$. Under such scenario, NMSO may miss the optimal solution in another node of the same depth, since the algorithm has already assumed that no further improvement is possible, thereby converging to a local optimal solution. One possible remedy is to prune the tree at nodes of $\mathcal{B}$. As a result, cells of these nodes are never explored. Such approach is not desired as it violates the consistency property–the optimal solution may lie in the never-explored region. Alternatively, NMSO keeps track of how many expansion sequences in which a node (h, i) in $\mathcal{B}$ has been visited, denoted by $v_{h,i}$. Subsequently, if $v_{h,i}$ is significant enough– say, $\geq V \in \mathbb{N}_1$, the algorithm expands (h, i).

The pseudo-code of NMSO is shown in Algorithm 7. At each round, it expands at most one node from the set of evaluated leaf nodes $\mathcal{E}$ per scale/depth (if any). Sorted by the function values at their representative states in an ascending order, NMSO expands the first node in $\mathcal{E}$ at depth h that satisfies one of the following; it does not belong to $\mathcal{B}$, or it belongs to $\mathcal{B}$ and has been visited V times. Otherwise, the algorithm moves to the next scale in the sequence. When a node is expanded, its child nodes are evaluated, i.e., $\mathcal{E} = \mathcal{L}$. NMSO decides whether to go to the next scale of partition or starts a new sequence from the root, based on the rule of expected improvement described earlier.

The NMSO algorithm comes with four parameters to set: $\alpha > 0$, of Eq. (4.3); $\beta > 0$, of Eq. (4.4); $K \geq 2$, the partitioning factor; and $V > 0$, the number of visits for a node $\in \mathcal{B}$ to be expandable. Although NMSO can be regarded as an exploitation-biased algorithm, one can reduce its exploitative nature dominance through the parameter V. Setting V to a large value forces NMSO to explore more nodes outside $\mathcal{B}$ and increases NMSO's chances of hitting the optimal for a problem of limited/expensive-budget settings.

Algorithm 7 Naive Multi-Scale Optimization (NMSO)

```
    Input:
 1: $\mathcal{T} \leftarrow$ initial tree with one evaluated node $(0,0)$ with
        $\mathcal{X}_{0,0} = \mathcal{X}$;
 2: $\mathbf{x}(v) \leftarrow \mathbf{x}_{0,0}$                                    ▷ current best solution
 3: $h \leftarrow 0$;                                                       ▷ current scale
 4: $\mathcal{B} \leftarrow \emptyset$;          ▷ a bag to collect the last expanded node from each sequence
 5: while evaluation budget is not exhausted do
 6:     $\mathcal{C} \leftarrow \{\text{node} \in \mathcal{E} \text{ of depth } h\}$;
 7:     while $|\mathcal{C}| > 0$ do
 8:         select the node $(h, i^*)$, such that:
                $$i^* \in \arg\min_{i:(h,i)\in\mathcal{C}} f(\mathbf{x}_{h,i})$$
 9:         if $(h, i^*) \in \mathcal{B}$ and $v_{h,i^*} < V$ then
10:             $v_{h,i^*} \leftarrow v_{h,i^*} + 1$ ;
11:             $\mathcal{C} \leftarrow \mathcal{C} \setminus (h, i^*)$;
12:         else
13:             break;
14:         end if
15:     end while
16:     if $|\mathcal{C}| == 0$ then
17:         $h \leftarrow h + 1$;                                          ▷ go to the next scale
18:         continue;
19:     end if
20:     expand $(h, i^*)$ into its child nodes $\{(h+1, i^*_k)\}_{1\le k\le K}$;
21:     add $\{(h+1, i^*_k)\}_{1\le k\le K}$ to $\mathcal{T}$;
22:     evaluate $\{(h+1, i^*_k)\}_{1\le k\le K}$;
23:     update the best solution $\mathbf{x}(v)$;
24:     if $h + 1$ is a multiple of $n$ and
25:             $\nabla^f_{h,i^*} \le \alpha$ and
26:             $\nabla^{\mathbf{x}}_{h,i^*} \le \beta$
27:        then                                         ▷ start a new sequence of expansions
28:         $h \leftarrow 0$;
29:         put $\{(h+1, i^*_k)\}_{1\le k\le K}$ in $\mathcal{B}$;
30:         $v_{h+1,i^*_k} \leftarrow 0$ , $\forall 1 \le k \le K$                          ▷ initialize a
31:             counter of visits for each
32:     else                                                        ▷ go to the next scale
33:         $h \leftarrow h + 1$;
34:     end if
35: end while
36: return $\mathbf{x}(v)$;
```

4.1.2 Convergence Analysis

In line with the methodology presented in Chapter 3, a theoretical analysis of NMSO is discussed in this section. For the sake of completeness and simplicity, assumptions—similar to those in Chapter 3—about the function and the hierarchical partitioning are stated. Then, we analyze the performance of NMSO and upper bound the loss (2.2) as a function of the number of expansion sequences.[1] Towards the end of this section, we present the main result on NMSO's finite-time performance and consistency (asymptotic performance) with their proofs.

Assumptions. There exists a semi-metric $\ell : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ (i.e., ℓ satisfies symmetry, non-negativity, and the coincidence axiom) such that:

A4.1 **(local smoothness of f)**:

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \ell(\mathbf{x}, \mathbf{x}^*),\ \forall \mathbf{x} \in \mathcal{X}$$

Thus, ensuring that f does not change too fast around its one global optimizer $\mathbf{x}^*$.

A4.2 **(bounded cells diameters)**: For all $(h, i) \in \mathcal{T}$, there exists a decreasing sequence $\delta(h) > 0$ such that: $\sup_{\mathbf{x} \in \mathcal{X}_{h,i}} \ell(\mathbf{x}_{h,i}, \mathbf{x}) \leq \delta(h)$ and $\lim_{h\to\infty} \delta(h) = 0$. Thus, ensuring the regularity of the cells' sizes which decrease with their depths in $\mathcal{T}$.

A4.3 **(well-shaped cells)**: For all $(h, i) \in \mathcal{T}$, there exists $v > 0$ such that a cell $\mathcal{X}_{h,i}$ contains an ℓ-ball of radius $v\delta(h)$ centered in $\mathbf{x}_{h,i}$. Thus, ensuring that the cells' shapes are not skewed in some dimensions.

Finite-Time Performance. In order to derive a bound on the loss, we use the near-optimality dimension—defined in Chapter 3—as a measure of the quantity of near-optimal solutions (states in $\mathcal{X}$). For the sake of simplifying the terminology used, the near-optimality dimension is redefined here similar to [34]. Prior to that, some of the terminology, used in Chapter 3, is revised briefly. For any $\epsilon > 0$,

[1]Typically, v in Eq. (2.2) represents the number of sampled points (function evaluations). Nevertheless, it can denote other growing-with-time quantities (e.g., the number of iterations, the number of expansion sequences). In the rest of this chapter, we refer to the number of the: function evaluations and expansion sequences, by v and n_s, respectively.

let us denote the set of ϵ-optimal *states*, $\{\mathbf{x} \in \mathcal{X} : f(\mathbf{x}) \leq f(\mathbf{x}^*) + \epsilon\}$, by $\mathcal{X}_\epsilon$. Subsequently, denote the set of *nodes* at depth h whose representative states are in $\mathcal{X}_{\delta(h)}$ by $\mathcal{I}_h$, i.e., $\mathcal{I}_h \overset{\text{def}}{=} \{(h,i) \in \mathcal{T} : 0 \leq i \leq K^h - 1, \mathbf{x}_{h,i} \in \mathcal{X}_{\delta(h)}\}$. A node (h,i) is optimal $\iff$ $\mathbf{x}^* \in \mathcal{X}_{h,i}$. After t expansion sequences, we denote the depth of the deepest expanded optimal node by h_t^*. Now, we define the near-optimality dimension.

Definition 19 (Near-optimality dimension)**.** *The near-optimality dimension is the smallest $d_v \geq 0$ such that there exists $C > 0$ such that for any $\epsilon > 0$, the maximal number of disjoint ℓ-balls of radius $v\epsilon$ and center in $\mathcal{X}_\epsilon$ is less than $C\epsilon^{-d_v}$.*

After t expansion sequences, NMSO's tree $\mathcal{T}$ has its deepest expanded optimal node at depth h_t^*. Assume that $(h_t^* + 1, i^*)$ is the optimal node at depth $h_t^* + 1$. Furthermore, assume that none of the nodes at depth $h_t^* + 1$ belongs to $\mathcal{B}$. Since $(h_t^* + 1, i^*)$ has not been expanded yet, any node at depth $h_t^* + 1$ that is later selected (line 8 in Algorithm 7) and expanded before $(h_t^* + 1, i^*)$ must satisfy the following:

$$f(\mathbf{x}_{h_t^*+1,i}) \leq f(\mathbf{x}_{h_t^*+1,i^*}) \tag{4.5}$$

$$f(\mathbf{x}_{h_t^*+1,i}) \leq f(\mathbf{x}^*) + \delta(h_t^* + 1) \ , \tag{4.6}$$

where Eq. (4.6) comes from combining Assumptions A4.1 and A4.2: $f(\mathbf{x}_{h_t^*+1,i^*}) \leq f(\mathbf{x}^*) + \ell(\mathbf{x}_{h_t^*+1,i}, \mathbf{x}^*) \leq f(\mathbf{x}^*) + \delta(h_t^* + 1)$. Thus, from the definition of $\mathcal{I}_h$, we are certain that $(h_t^* + 1, i^*)$ gets expanded after $|\mathcal{I}_{h_t^*+1}|$ node expansions at depth $h_t^* + 1$ in the worst-case scenario. Now, consider the case where any of these nodes can be in $\mathcal{B}$. Such node needs to be visited in V expansion sequences before it can be expanded. Thus, at most $V|\mathcal{I}_{h_t^*+1}|$ expansions at depth $h_t^* + 1$ are needed to expand the optimal node $(h_t^* + 1, i^*)$. From this observation, we deduce the following lemma.

Lemma 6. *In NMSO, after n_s expansion sequences, for any depth $h \in \mathbb{N}_0$ whenever $n_s \geq V \sum_{l=0}^{h} |\mathcal{I}_l|$, we have $h_{n_s}^* \geq h$.*

Proof. We know that $h_{n_s}^* \geq 0$ and hence the above statement holds for $h = 0$. For $h \geq 1$, we are going to prove it by induction. Assume that the statement holds for $0 \leq h \leq \hat{h}$. Let us then prove it for $h \geq \hat{h} + 1$. Let $n_s \geq V \sum_{l=0}^{\hat{h}+1} |\mathcal{I}_l|$, and hence, $n_s \geq V \sum_{l=0}^{\hat{h}} |\mathcal{I}_l|$ for which we know by our assumption that $h_n^* \geq \hat{h}$. Here, we have two cases: $h^*_n \geq \hat{h}+1$, for which the proof is done; or $h^*_n = \hat{h}$. In the second case, any node at depth $\hat{h}+1$, which is expanded before the optimal node at the same depth, is one of two types. Either it belongs to $\mathcal{I}_{\hat{h}+1}$, or it has been selected and expanded in one of the $V-1$ visits to a node in $\mathcal{I}_{\hat{h}+1}$. Since there are $V|\mathcal{I}_{\hat{h}+1}|$ of such nodes, and the fact that only one node per depth gets expanded in one sequence of expansions; we are certain that the optimal node at depth $\hat{h}+1$ is expanded after at most $V|\mathcal{I}_{\hat{h}+1}|$ sequences. Therefore, we have $h^*_n \geq \hat{h}+1$. □

In other words, the size of $\mathcal{I}_h$ gives a measure of how much exploration is need provided that the optimal node at depth $h-1$ has expanded. The near-optimality dimension lets us bound the size of $\mathcal{I}_h$. From Lemma 1, we have $|\mathcal{I}_h| \leq C\delta(h)^{-d_v}$. We now provide `NMSO`'s loss bound in terms of the number of expansion sequences n_s.

Theorem 7 ($r(n_s)$ for `NMSO`). *Let us define $h(n_s)$ as the smallest $h \in \mathbb{N}_0$ such that:*

$$VC\sum_{l=0}^{h(n_s)} \delta(l)^{-d_v} \geq n_s \,, \tag{4.7}$$

where n_s is the number of expansion sequences. Then, the loss of `NMSO` is bounded as:

$$r(n_s) \leq \delta(h(n_s)) \tag{4.8}$$

Proof. From Lemma 1 and the definition of $h(n_s)$, we have:

$$\sum_{l=0}^{h(n_s)-1} |\mathcal{I}_l| \leq C \sum_{l=0}^{h(n_s)-1} \delta(l)^{-d_v} < n_s \tag{4.9}$$

Thus, from Lemma 6, we have $h^*_{n_s} \geq h(n_s)-1$. Now, let $(h^*_{n_s}+1, i^*)$ be the deepest non-expanded optimal node (which is a child node of the deepest expanded optimal node at depth $h^*_{n_s}$), then the loss is bounded as

$$r(n_s) \leq f(\mathbf{x}_{h^*_{n_s}+1,i^*}) - f(\mathbf{x}^*) \leq \delta(h^*_{n_s}+1) \tag{4.10}$$

Since $h^*_{n_s} \geq h(n_s)-1$, therefore, we have $r(n_s) \leq \delta(h(n_s))$. □

Remark 7. *(A generic finite-time analysis for multi-restart algorithms) Under the Assumptions A4.1, A4.2, and A4.3; our finite-time analysis can be applied to any exploitative algorithm whose strategy includes a multi-restart component within the MSO framework, provided no node is pruned. i.e., the theoretical analysis here holds irrespective of the expansion sequence stopping rule, as well as the rule for expanding a node from $\mathcal{B}$.*

Consistency Property. Theorem 7 addressed the finite-time performance of NMSO. Let us consider its consistency property. An algorithm is consistent if it asymptotically converges to the globally optimal fitness value. NMSO guarantees that no portion of $\mathcal{X}$ is disregarded. Accordingly, if an optimal node happens to be the last expanded node of an expansion sequence, then its optimal child node will get expanded in one of the next expansion sequences. As the number of expansion sequences n_s grows bigger, the base points sampled by NMSO form a dense subset of $\mathcal{X}$ such that for an arbitrary small $\epsilon \geq 0$ and with our assumption on f's smoothness, there exists a base point $\mathbf{x}$ such that $f(\mathbf{x}) - f(\mathbf{x}^*) \leq \epsilon$. The next theorem puts formally our proposition about the consistency property of NMSO.

Theorem 8. *For NMSO,* $\lim_{n_s\to\infty} r(n_s) = 0$.

Proof.

$$\begin{aligned}
\lim_{n_s\to\infty} r_{n_s} &\leq \lim_{n_s\to\infty} \delta(h(n_s)) && \text{from Theorem 7,}\\
&\leq \lim_{h(n_s)\to\infty} \delta(h(n_s)) && \text{from the definition of } h(n_s),\\
&\leq 0 && \text{from Assumption A4.2.}
\end{aligned}$$

Thus, as $r(n_s) \geq 0$, we have $\lim_{n_s\to\infty} r(n_s) = 0$. □

4.2 Numerical Assessment

In this section, empirical assessment and comparison of NMSO with the state-of-the-art MSO algorithms are presented.[2] We have adopted the Comparing Continuous

[2]The online supplement #1, at https://www.dropbox.com/s/a1hdwcno9allxnj/supplement-materials-1.pdf??dl=0, provides the algorithm's assessment with respect to the state-of-the-art stochastic algorithms.

Optimizer (COCO) methodology [58] to distinguish the good features of one algorithm over the others, and show the difference in the algorithms behavior over several settings of an optimization problem. The COCO platform comes with the black-box optimization benchmarking (BBOB) testbed, which has twenty-four scalable noiseless functions given in [189], addressing such real-world difficulties as ill-conditioning, multi-modality, and dimensionality. It has been used for the BBOB workshops that took place during the GECCO conference in 2009, 2010, 2012, and 2013, besides the CEC conference in 2015.

The rest of the section discusses the numerical assessment in terms of: setup, evaluation procedure, and algorithms comparison with regard to performance and computational complexity.

4.2.1 Experimental Setup

NMSO and five of the state-of-the-art MSO algorithms, namely BB-LS [190], DIRECT [56], MCS [27], SOO [34] and BaMSOO [5] are benchmarked on twenty-four functions (with $N_{trial} = 15$ trials/runs per function) of the BBOB testbed using an evaluation budget of $10^4 \cdot n$ function evaluations v (#f-evaluations) per trial. The goal in each trial is to reach a target function value f_t over a search space of $[-5, 5]^n$ for $n \in \{2, 3, 5, 10, 20, 40\}$. The numerical results are presented in two forms: data profile plots and tables displaying quantitative ERT values (and additionally providing statistical significance tests). Here, we only report the plots (the tables can be found in the online supplement #2 at https://www.dropbox.com/s/69tovqcden39yy6/supplement-materials-2.pdf?dl=0)

The algorithms terminate in one of two cases: exhausting the evaluation budget; or hitting f_t. Only algorithms parameters related to the evaluation budget are modified accordingly, other parameters are set to their standard values. MATLAB implementations/wrappers[3] of the algorithms are retrieved from the sources shown in Table 4.1 and used for our experiments.

NMSO's main parameters, viz. α, β, and V, are set as shown in Table 4.2: in accordance with many single-objective MSO algorithms, the partition factor K is set to be 3. V can be adjusted according to the evaluation budget v to suit various budget settings, but since V's purpose is to guarantee asymptotic convergence,

[3]BB-LS is implemented in C, and interfaced to MATLAB using a mex file interface.

Table 4.1: List of MSO algorithms considered for empirical analysis and their implementation. We have used our own implementation of BaMSOO [5].

Algorithm	Source
BB-LS	http://tomopt.com/
DIRECT	www4.ncsu.edu/~ctk/Finkel_Direct/
SOO	sequel.lille.inria.fr/Software/StoSOO
MCS	www.mat.univie.ac.at/~neum/software/mcs/
BaMSOO	http://ash-aldujaili.github.io/BaMSOO/
NMSO	http://ash-aldujaili.github.io/NMSO/

Table 4.2: Parameters setting of NMSO.

Parameter	Value
α	$10^{-8} \cdot n$
β	$10^{-8} \cdot n \cdot \max_{1\leq i\leq n}(u_i - l_i)$
V	$1000 \cdot n$
K	3

one can set it to a large value (e.g., $1000 \cdot n$). On the other hand, as NMSO's principle is *exploit until no further improvement is observed*, α and β are set with a small multiplicative tolerance factor 10^{-8}, similar to several algorithms' stopping criterion from the literature. Nevertheless, they are made proportional to the problem dimensionality n to scale the exploration in line with the search space's volume.

An essential aspect of the algorithm is the expansion/splitting mechanism. As mentioned in Section 4.1, a cell $\mathcal{X}_{h,i}$ is expanded along the $(h \mod n+1)^{\text{th}}$ dimension. In other words, the expansions in an expansion sequence take place along one dimension after the other in a round-robin fashion over an ordered sequence of the dimensions $\{1,\ldots,n\}$. However, some fitness functions—primarily ill-conditioned—vary across one dimension (e.g., the fifth) more rapidly than another (e.g., the second), and hence splitting the cell along the fifth before the second dimension is more likely to result in a big improvement in the solution quality than otherwise.

In our experiments, the dimensions are ordered according to an empirical measure of how much f varies along each of them. Prior to growing the tree, along each dimension i, two points $\mathbf{x}_1^i, \mathbf{x}_2^i$ are sampled and the absolute difference between their f values is computed, $\lambda_i = |f(\mathbf{x}_1^i) - f(\mathbf{x}_2^i)|$. The splitting mechanism is then restated as follows. A cell $\mathcal{X}_{h,i}$ is split along the dimension i whose λ_i ranked $(h \mod n+1)^{\text{th}}$ among $\{\lambda_i\}_{1\leq i\leq n}$.

4.2.2 Performance Evaluation Procedure

The performance experiments are set up according to [58], where each algorithm is run on the twenty-four functions with N_{trial} trials per function. A set of target function values is specified per function. The algorithms are evaluated based on the number of function evaluations required to reach a target. The Expected Running Time (ERT) [191] is used here to measure the performance of the algorithms in reaching these targets. The ERT depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f_t$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [58, 191]. Statistical significance is tested with the rank-sum test for a given target Δf_{t} using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration. For more details on ERT and statistical test procedure, one should refer to [58].

4.2.3 Performance Evaluation Presentation

In line with the performance evaluation procedure discussed in Section 4.2.2, the numerical results are presented in two forms: plots and tables. Based on a bootstrapped distribution of the ERT, the figures (e.g., Figure 4.3) show the Empirical Cumulative Distribution Function (ECDF) of the problems solved (targets hit) as a function of the evaluation budget divided by dimension (#f-evaluations/Dim) aggregated over all the objective functions, as well as over different function categories, namely separable, moderate, ill-conditioned, multi-modal, and weakly structured multi-modal functions. The targets are chosen from $10^{[-8..2]}$ such that the bestGECCO2009 artificial algorithm just not reached them within a given budget of $k \times DIM$, with $k \in \{0.5, 1.2, 3, 10, 50\}$. The best 2009 line corresponds to the best ERT observed during BBOB 2009 for each selected target.

On the other hand, the tables (reported in supplement #1) present a statistical comparison of the algorithms' ERT, measuring their spreads for a given set of target function values and highlight statistically significantly better algorithms compared with others. Best results are printed in bold. Entries, followed by the symbol $\star$ are statistically significantly better compared to all others algorithms, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \ldots\}$ is the number following the $\star$, with Bonferroni correction by the number of instances. The symbol $\downarrow$ indicates the same tested against the best algorithm of BBOB-2009. The ERT and in braces, as dispersion measure, the half difference between 90 and 10%-tile of bootstrapped run lengths appear for each algorithm and run-length based target, the corresponding best ERT (preceded by the target Δf-value in italics) in the first row. #succ is the number of trials that reached the target value of the last column. The median number of conducted function evaluations is additionally given in italics, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1, with Bonferroni correction by the number of instances.

4.2.4 Performance Evaluation Discussion

Results from the performance experiments for $n \in \{2, 3, 5, 10, 20, 40\}$ are presented in Figures 4.1 to 4.6, respectively. The discussion of the results is broken down according to three key points, viz. *function category, dimensionality and evaluation budget.*

4.2.4.1 Function Category

- *Separable* functions f_1–f_5: Aggregated ECDF graphs of the ERT for these (uni- and multi-modal) functions in $n \in \{2, 3, 5, 10, 20, 40\}$ are shown in the upper left part of Figures 4.1 to 4.6, respectively. NMSO and MCS dominate the rest of the algorithms especially in their initial phase (#f-evaluations $\leq 10 \cdot n$) with a steep convergence, solving about 90% of the problems– those associated with the separable functions. In a later phase (#f-evaluations $\geq 1000 \cdot n$), the performance of the rest of the algorithms (BaMSOO followed

Figure 4.1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 2-D.

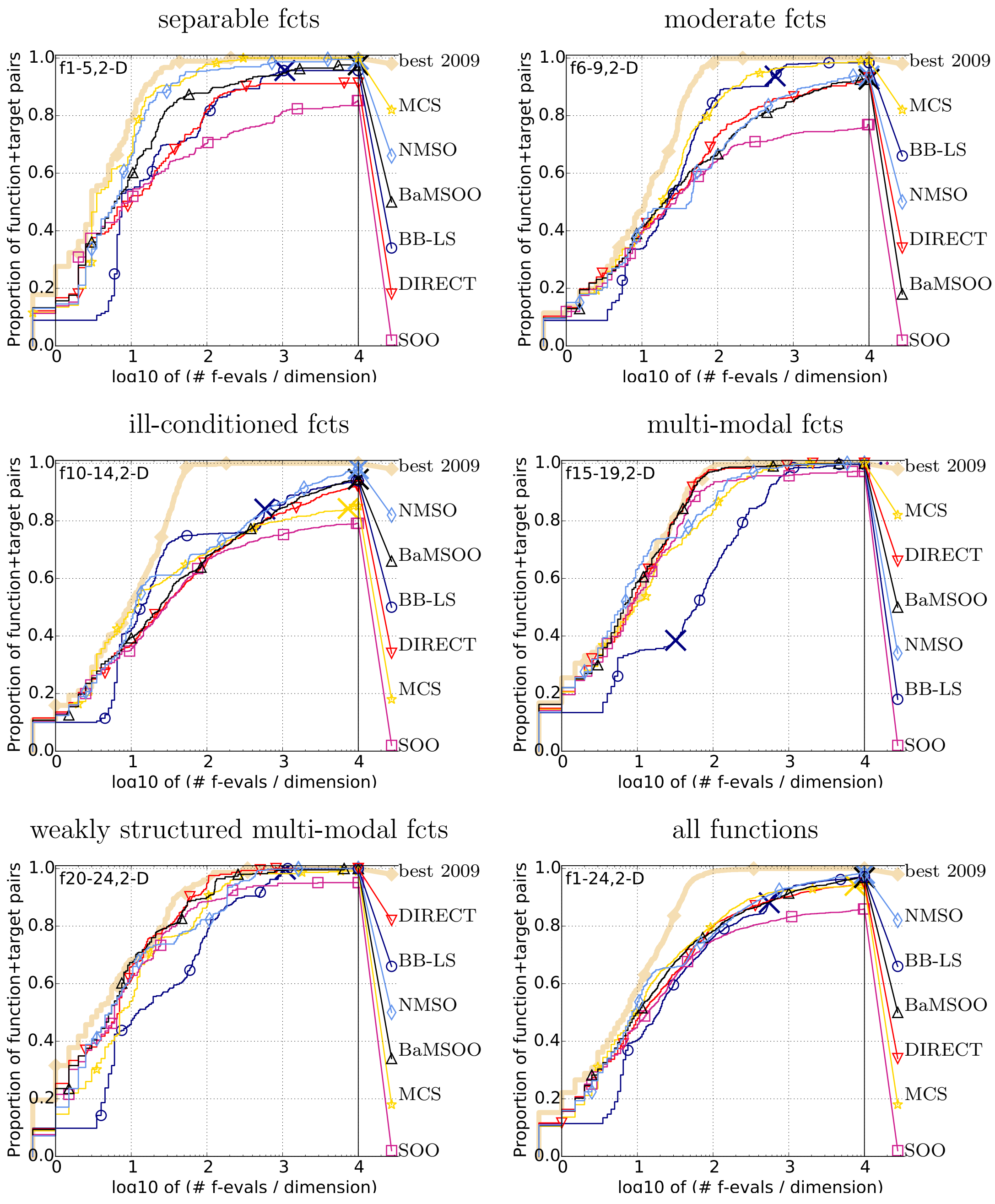


Figure 4.2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 3-D.

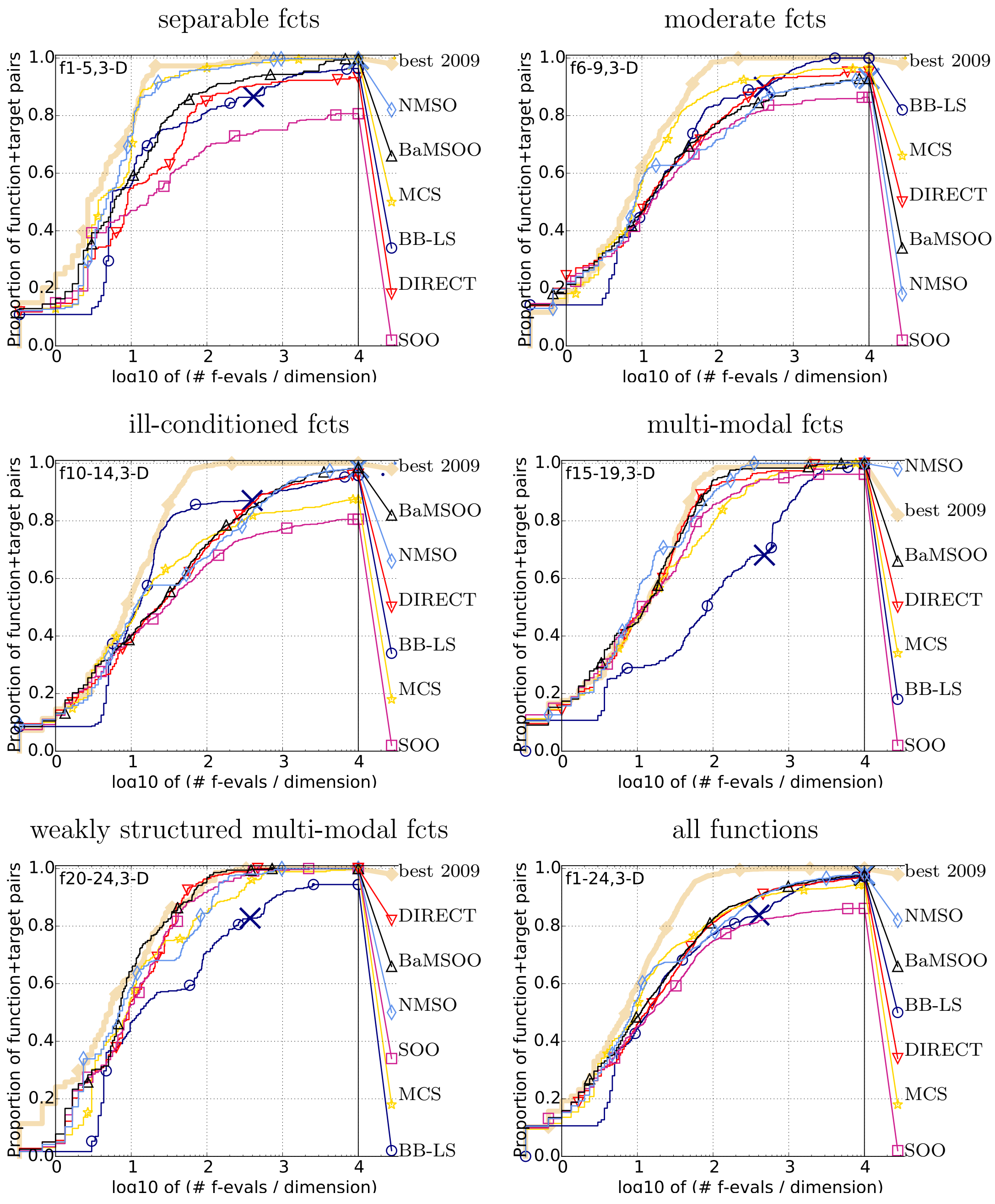


Figure 4.3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 5-D.

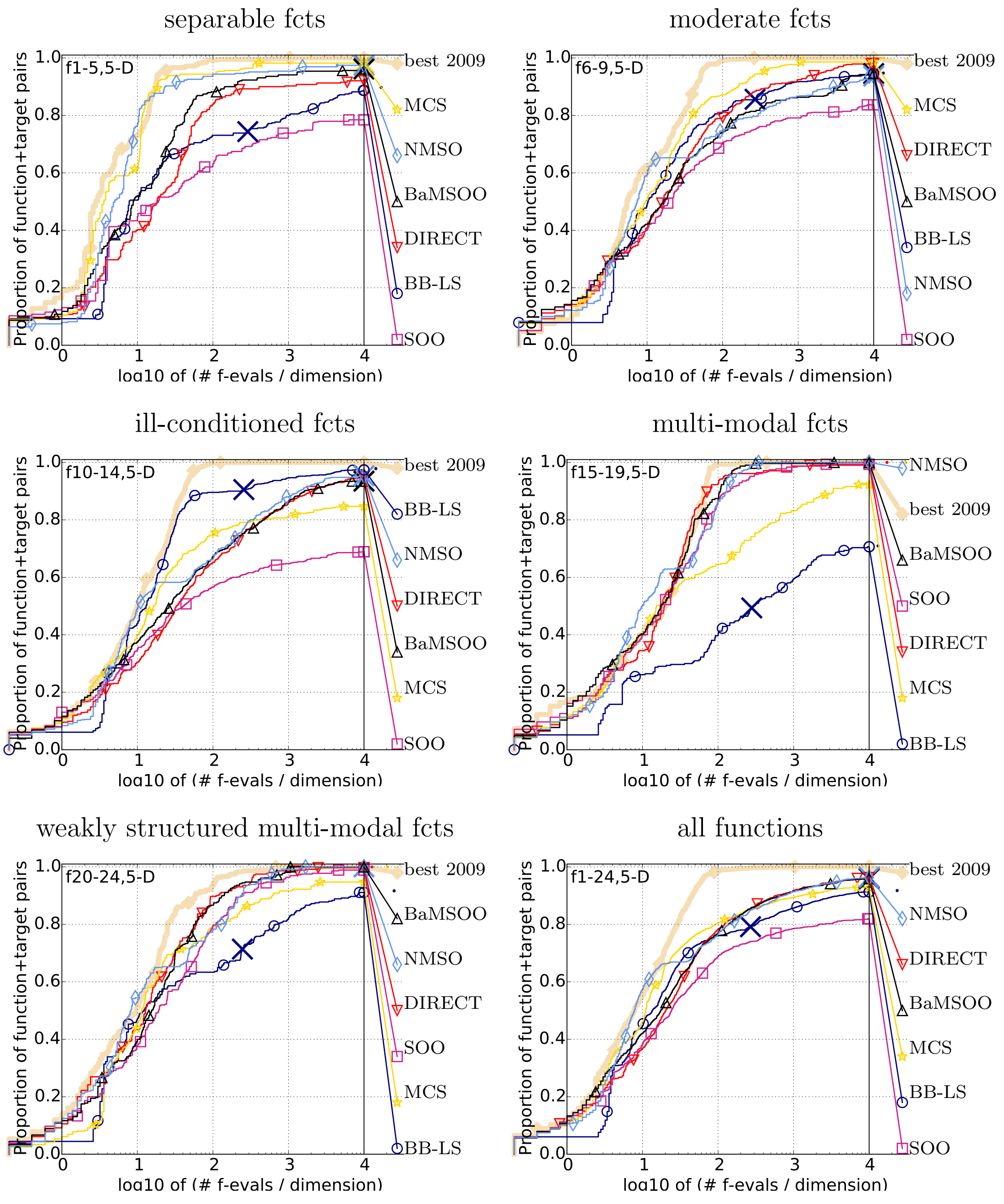


Figure 4.4: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 10-D.

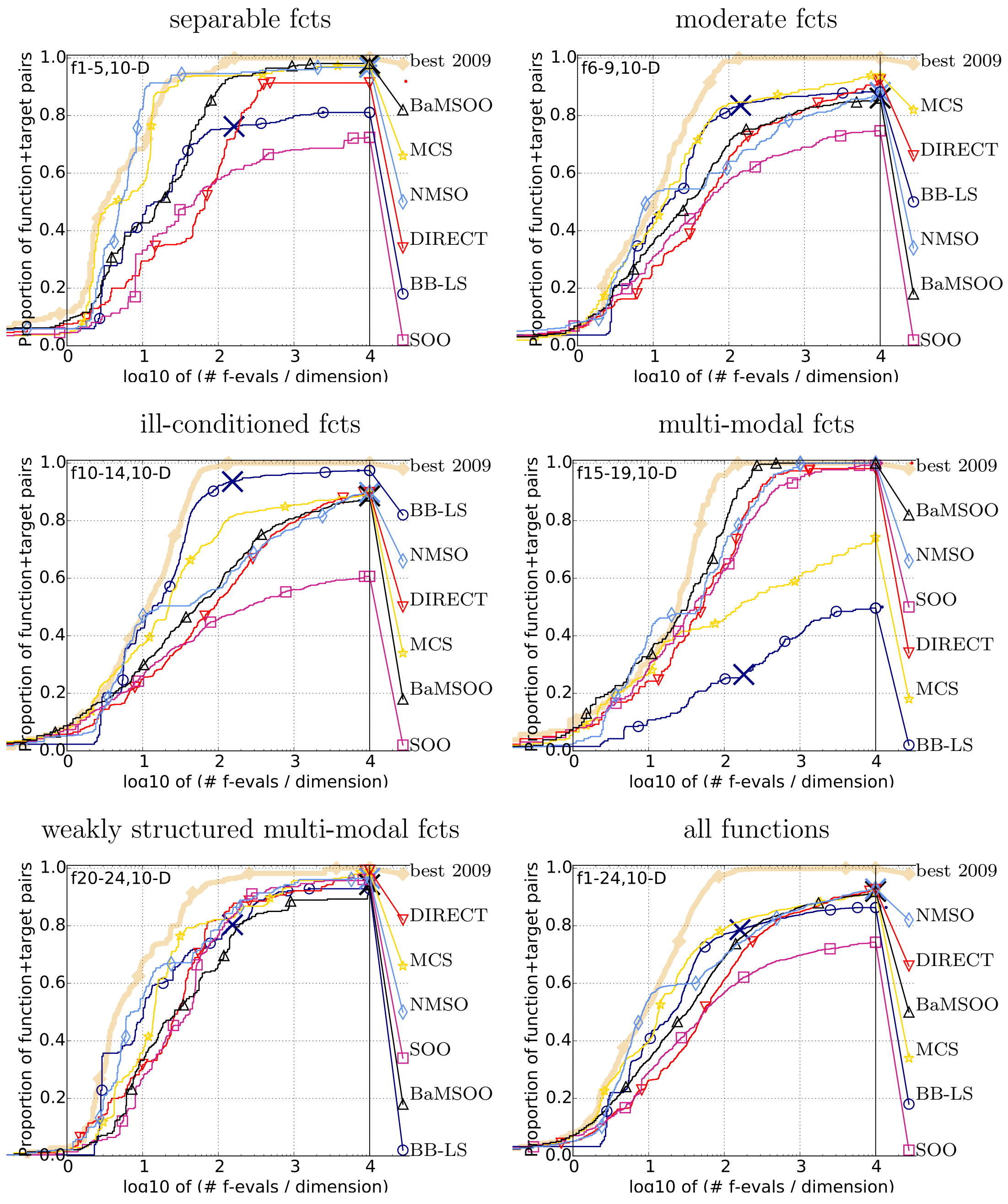
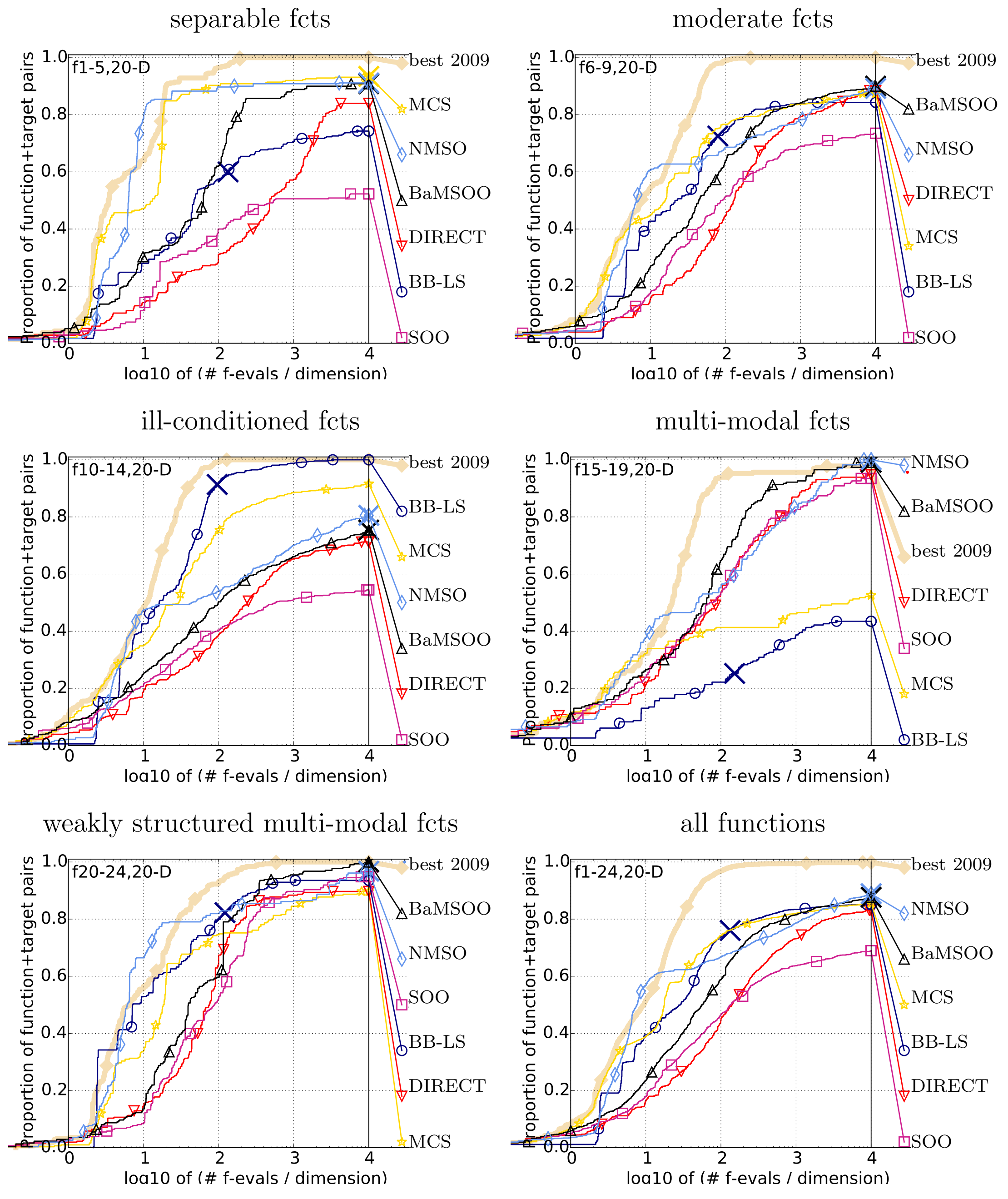


Figure 4.5: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 20-D.



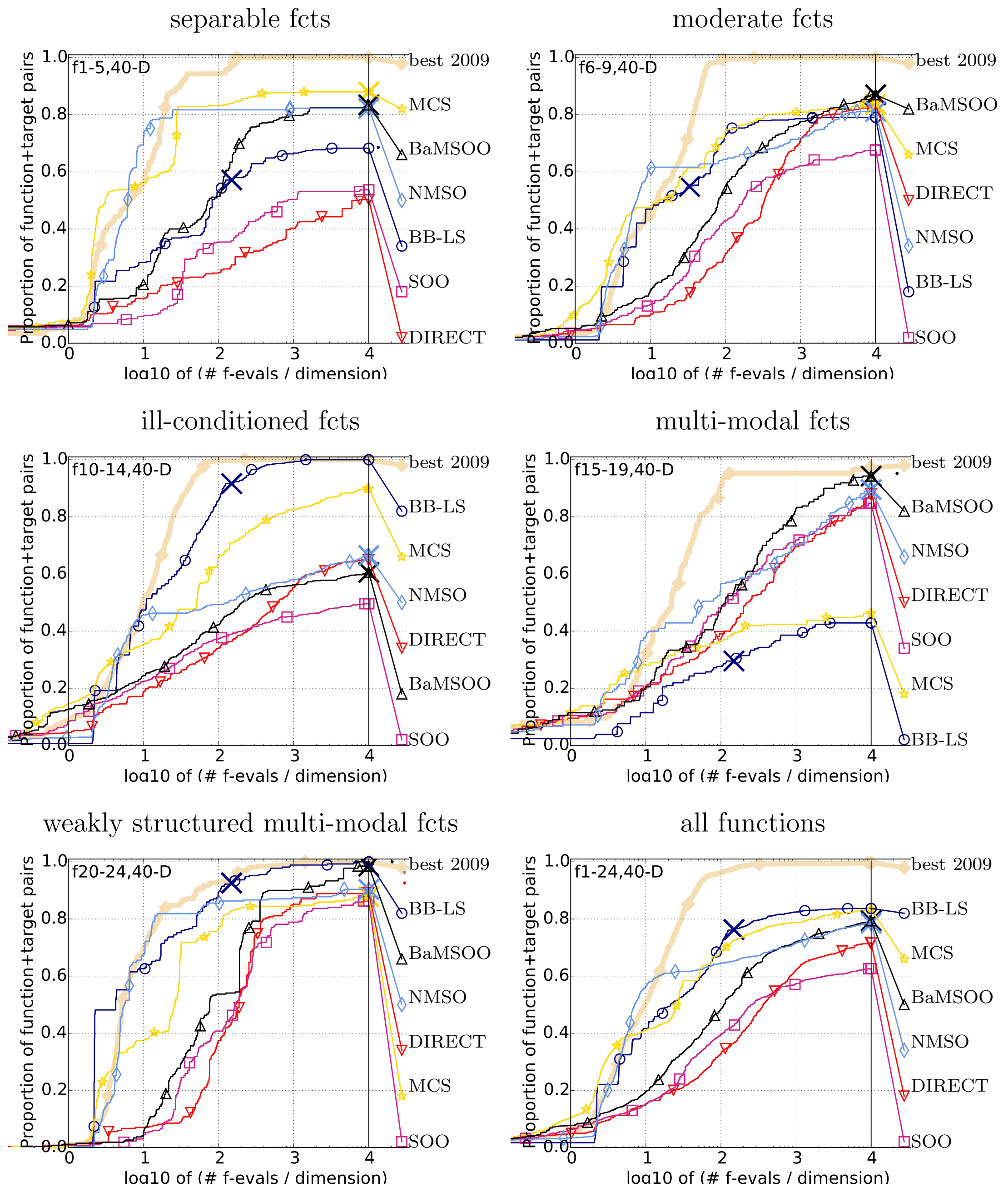


Figure 4.6: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 40-D.

by BB-LS, DIRECT, and SOO) follow that of NMSO and MCS. The performance of BB-LS, DIRECT, and SOO is clearly decoupled from that of MCS, NMSO, and BaMSOO as the problem's dimensionality increases.

- *Moderate* functions f_6–f_9: Aggregated ECDF graphs of the ERT for these functions in $n \in \{2, 3, 5, 10, 20, 40\}$ are shown in the upper right part of Figures 4.1 to 4.6, respectively. Except for 2-D problems, NMSO solves around 60% of the problems very rapidly (#f-evaluations $\approx 10 \cdot n$), dominating even the bestGECCO2009 artificial algorithm. With more function evaluations, the rest of the algorithms leaving out SOO solve more problems (more than 80% across all the dimensions).

- *Ill-conditioned* functions f_{10}–f_{14}: Aggregated ECDF graphs of the ERT for these functions in $n \in \{2, 3, 5, 10, 20, 40\}$ are shown in the middle left part of Figures 4.1 to 4.6, respectively. In spite of its initial slow convergence (#f-evaluations $< 10 \cdot n$), BB-LS shows a sharp convergence behavior for all the associated problems and across all the dimensionality settings. This may be due to its adaptive partitioning nature that the rest of the MSO algorithms do not employ. Nonetheless, NMSO, BaMSOO, and DIRECT solve nearly an equal portion of problems in low dimension (2-D, 3-D, and 5-D), which degrades with dimensionality. On the other hand, MCS shows a consistent performance against dimensionality, solving around 90% of the problems. SOO's performance gap with respect to BB-LS grows bigger with dimensionality.

- *Multi-modal* functions f_{15}–f_{19}: Aggregated ECDF graphs of the ERT for these functions in $n \in \{2, 3, 5, 10, 20, 40\}$ are shown in the middle right part of Figures 4.1 and 4.6, respectively. In lower dimensions of 2-D and 5-D, all the algorithms perform well, solving almost all the problems. Nevertheless, the convergence behavior of BB-LS is slow and different compared to the rest of algorithms whose graphs follow closely that of the bestGECCO2009 artificial algorithm. With increasing dimensionality, the performance of BB-LS and MCS drops significantly (solving less than 50%), while the rest maintain a consistence convergence behavior. NMSO beats the bestGECCO2009 artificial algorithms in 5-D and 20-D.

- *Weakly structured multi-modal* functions f_{20}–f_{24}: Aggregated ECDF graphs of the ERT for these functions in $n \in \{2, 3, 5, 10, 20, 40\}$ are shown in the lower left part of Figures 4.1 to 4.6, respectively. All the algorithms exhibit a similar convergence in solving the problems, with a performance gap, in their initial phase, which grows with dimensionality. BB-LS, NMSO, and MCS solve in their initial phase a large portion of the problems (around 60%) faster with dimension than the rest of algorithms. However, with more function evaluations, BaMSOO emerges as a suitable alternative for these algorithms.

4.2.4.2 Dimensionality

Examining the data profiles of all functions across all the dimensions tested (the lower right part of, e.g., Fig. 4.1), the following observations can be stated about the algorithms suitability with respect to the problem dimensionality, which have been categorized into low- and high-dimensionality groups in line with the COCO platform [58].

- *Low*-dimensionality ($n \in \{2, 3, 5\}$): In general, there is no big performance gap among the algorithms in problems of low-dimensionality. Nevertheless, NMSO and MCS appear to do well over different settings of the evaluation budget.

- *High*-dimensionality ($n \in \{10, 20, 40\}$): The performance gap among the algorithms grows bigger with higher dimensions. Over different settings of the evaluation budget MCS, BB-LS, BaMSOO, and NMSO are suitable for high-dimensional optimization problems.

4.2.4.3 Evaluation Budget

Looking across the data profiles with two scenario: expensive optimization (#f-evaluations $\leq 10^2 \cdot n$) and cheap optimization (#f-evaluations $> 10^2 \cdot n$), the following observations can be stated about the algorithms suitability with respect to the evaluation budget.

- *Expensive*-budget settings (#f-evaluations $\leq 10^2 \cdot n$): Overall, NMSO and MCS exhibit a very fast convergence making them suitable solvers for expensive problems. For problems with an ill-conditioned or weakly structured multi-modal function, BB-LS appear to be equally effective.

- *Cheap*-budget settings (#f-evaluations $> 10^2 \cdot n$): In general, SOO's performance relatively degrades with a bigger evaluation budget. This holds true for BB-LS, and MCS when dealing with multi-modal functions. BaMSOO consistently solves more problems with more function evaluations, which suggests its suitability for cheap- rather than expensive-budget settings. Likewise, NMSO solves more problems with increasing function evaluations, particularly with multi-modal functions.

Limitations. Benchmarking NMSO on the BBOB testbed clearly shows its efficacy in solving optimization problems, especially in *low-dimensionality* and *expensive-budget* settings with objective functions that are *separable* or *multi-modal.* NMSO's coordinate-aligned sequential partitioning is the main drawback of the proposed technique. This is supported by the empirical validation on problems with ill-conditioned objectives (see the middle left part of, e.g., Fig. 4.6). One can improve its performance by integrating the adaptive encoding strategy, proposed by Hansen [192], as demonstrated in [193]. Another disadvantage of NMSO's partitioning scheme is its poor scalability with respect to search space dimensionality. Recently proposed *embedding* techniques (e.g., [194]) can be used to rectify this issue.

4.2.5 Empirical Computational Complexity

In order to evaluate the complexity of the algorithms (measured in time per function evaluation), we have run the algorithms on the function f_8 (arbitrarily chosen as a moderate function) for $100 \cdot n$ function evaluations with $n \in \{2, 3, 5, 10, 20, 40\}$. The empirical complexity of an algorithm is then computed as the running time (in seconds) of the algorithm summed over different settings of n divided by the total count of function evaluations. The results are shown in Fig. 4.7. From the figure, we can observe that BaMSOO and SOO are computationally complex when compared to the rest of the MSO algorithms. BaMSOO's computational complexity is very high due to the Gaussian process computations. While BB-LS's efficiency can be attributed to its C implementation, MCS and DIRECT are computationally cheap despite being implement in MATLAB.

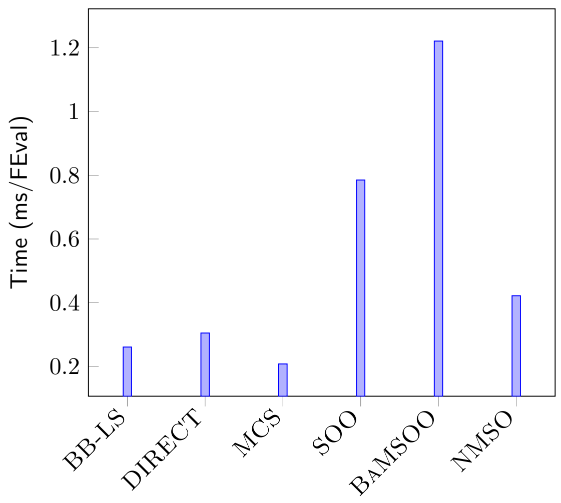


Figure 4.7: Timing Performance of NMSO compared with the state-of-the-art MSO optimization algorithms. The y-axis represents the time (in millisecond) per a function evaluation (FEval). All the algorithms were run on a PC with: 64-bit Ubuntu, i7-4702MQ @ 2.20GHz CPU, 8GB of memory.

4.3 Black Box Optimization Competition (BBComp)

The black box optimization competition (BBComp) within the GECCO'15 conference provides a testbed for evaluating optimization algorithms that is truly a black box to participants. The testbed consists of 1000 problems covering a wide range of problems with $n \in \{2, 4, 5, 8, 10, 16, 20, 32, 40, 64\}$. The only information known to the optimizer is the dimension of the problem, bounds on all variables, and a budget of black box queries $v \in [10, 100] \cdot n$. The competition allowed the participants to run their algorithms *only once* on the problem set—so no second chance or room for overfitting. The overall score/ranking was evaluated based on aggregated problem-wise ranks as follows. Similar to the *formula one* scoring system, the algorithms are sorted—for each problem—with respect to their best solution achieved. Based on this sort and the number of participants, each algorithm receives a score. These scores are computed in a way that amplifies the differences of top performing algorithms (low ranks) and suppresses those of bad performance (high ranks). For more details, one could refer to [57].

Table 4.3 lists the twenty-eight participating algorithms with overall scores (higher is better). NMSO finished *third* after the Nonlinear Interior point Trust Region Optimization (KNITRO) algorithm [195] and the Mean-Variance Mapping Optimization (MVMO) [196] algorithm. KNITRO, developed by Byrd et al. [195] at

Table 4.3: Algorithms Ranking in BBComp within GECCO'2015

Rank	Algorithm	Score
1	KNITRO [195]	1268.76
2	MVMO [196]	1191.54
3	NMSO	1099.04
4	A bag of algorithms: faced response surface design optimized with SCIP/YALMIP, CMA-ES, kriging-Surrogate, and Nelder-Mead	1092.39
5	Jumping Duplex	879.125
6	CTA : Curved Trajectories Algorithm [197]	689.558
7	RBFOpt [198]	669.860
8	SaDE: Self-adaptive DE [199]	607.344
9	b6e6rl [200]	568.964
10	Crossing differential evolution and CMA-ES	537.304
11	UNAMED	449.830
12	MCRS [201]	376.076
13	RD&R	323.617
14	UNAMED	293.915
15	Learn to optimization	278.599
16	TRDF : Trust-region Derivative-free Algorithm* [202]	277.811
17	Machine-learning-guided optimization	218.893
18	RBM-ES [203]	216.784
19	Memetic schema with selection features	204.664
20	MADS: Mesh Adaptive Direct Search [30]	202.549
21	CMA-ES flavors	171.661
22	DDS [204]	124.074
23	DE [205]	115.440
24	Self-adaptive, Two-Phase Differential Evolution	109.887
25	New Custom Method	98.5314
26	GACE - Genetic Algorithm with Cross Entropy [206]	91.7493
27	BEA-GECCO2015	61.7187
28	UNAMED	41.3182

Artelys, is a commercial optimization software package, which employs state-of-the-art active-set and interior-point methods for solving nonlinear optimization problems. On the other hand, the MVMO algorithm [196] is a population-based stochastic optimization technique, which applies a special mapping function for producing offspring solutions based on the mean and variance of the $k \geq 1$ best solutions obtained so far. Compared with NMSO finite-time and consistency properties, KNITRO is not consistent, while MVMO can find the optimum quickly with minimum risk of premature convergence. NMSO's standing validates its efficacy in solving black-box multi-dimensional optimization problems with expensive budget when compared with other state-of-the-art optimization algorithms coming from several (mathematical programming, evolutionary computation, machine learning, etc.) communities.

4.4 Conclusion

This chapter proposed and analyzed NMSO, Multi-scale Search Optimization (MSO) algorithm for solving global optimization problems given a finite number of function evaluations. It enjoys two main theoretical properties, namely the finite-time convergence rate and the consistency property. Moreover, the theoretical analysis of NMSO provides a generic basis for analyzing any multi-restart MSO optimization algorithm.

Along with its theoretical study, a numerical assessment of NMSO, comparing it with the leading state-of-the-art MSO algorithms has been conducted. Based on the results, NMSO demonstrates a better performance than the state-of-the-art MSO algorithms in scenarios with separability, multi-modality, limited (expensive) evaluation budget, and low dimensionality. This is in line of the Black-box Optimization Competition (BBComp) held at GECCO'15, where NMSO finished third out of twenty-eight competitors in solving 1000 black-box problems. Therefore, one can use NMSO as a suitable off-the-shelf solver for computationally-expensive black-box real-world optimization problems.

Chapter 5

Multi-Objective Multi-Scale Search

"The trade-off between speed and image quality is a key constraint of first-person action games, and the job of developing a workable engine involves constantly optimizing both elements. Gamers dream of the day they'll be able to haul their arsenals through three-dimensional environments of photographic clarity, playing 'Myst' with a meat ax"

- Marc Laidlaw

This chapter addresses the continuous black-box *Multi-objective Optimization Problem* (MOP) where it is desired to optimize two or more objectives. Such problem has applications in various science and engineering disciplines [10, 38, 40]. For instance, in manufacturing a product, the makers would like to minimize its cost and maximize its performance, simultaneously.

As the objectives of an MOP are often conflictual, there does not exist a single optimal global solution—which is often the case in single-objective optimization—but a set of incomparable optimal solutions: each is inferior to the other in some objectives and superior in other objectives, inducing a partial order on the set of feasible solutions to an MOP. The set of optimal feasible solutions according to this partial order is referred to as the *Pareto optimal set* and its corresponding image in the objective space is commonly named as the *Pareto front* of the problem. For more details on the background of multi-objective optimization and its conventional methods, one can refer to Section 2.4.2.

Among the multi-objective solvers that witnessed an experimental success is the Multi-objective Evolutionary Algorithm Based on Decomposition (`MOEA/D`) framework [184]. Such framework, taking inspiration from the *divide-and-conquer* paradigm, decomposes an MOP into a number of different single objective optimization subproblems (or simple multi-objective optimization subproblems) and then employs a population-based method to optimize these subproblems, at the same time. The research community has been extensively studying the `MOEA/D` framework in terms of its strengths, weaknesses, variants, generalizations, and applications (e.g., see [207, 208, 209, 210]).

In this chapter, we take inspiration from the *divide-and-conquer* paradigm from a different angle, viz. Mathematical Programming (MP). In particular, we are inspired to expand the frontier of Multi-scale Search Optimization (MSO) algorithms towards MOPs. These methods systematically sample the decision space, which may provide an insight onto one of the important lessons learned over the past decades from multi-objective solvers; that is, in order to generate a dense and good approximation set, one must maintain the set diversity and must not discard inferior solutions too easily, as some of them may pave the way towards rarely-visited regions of the Pareto front [44]. Furthermore, MSO's theoretical foundation may help towards better understanding of multi-objective solvers' performance. In general, two observations can be made about approaches similar to MSO methods within the context of multi-objective optimization. First, there has been very little/limited yet slowly growing research reported on such algorithms for multi-objective optimization. For instance, the focus has been distinctly on a discrete set of solutions [211], or solving a subproblem (e.g., selecting a genetic operator in evolutionary multi-objective algorithms by Li et al. [212]). Second, the algorithmic development and validation have been dominantly empirical [see, for instance, 132, 133, 213].

With this regard, in this chapter, the suitability of MSO methods for multi-objective optimization problems is formally investigated by proposing and analyzing two multi-objective algorithmic instances, viz. the Multi-Objective DIviding RECTangles (`MO-DIRECT`) and the the Multi-Objective Simultaneous Optimistic Optimization (`MO-SOO`) based on the single-objective MSO algorithms `DIRECT` [56] and `SOO` [34], respectively.

The rest of the chapter is organized as follows. Section 5.1 presents the `MO-DIRECT` and `MO-SOO` algorithms as multi-objective algorithmic instances of the MSO framework. Both of the algorithms are demonstrated on a worked example in Section 5.2.

This is followed by a theoretical analysis of the algorithms in Section 5.3. Section 5.4 concludes the chapter.

5.1 Multi-Scale Search for Multi-Objective Optimization

In this section, two MSO algorithmic instances are tailored towards multi-objective optimization problems based on the DIviding RECTangles (`DIRECT`) [56] and Simultaneous Optimistic Optimization (`SOO`) [34] algorithms. We refer to the proposed algorithms as the Multi-Objective DIviding RECTangles (`MO-DIRECT`) and the Multi-Objective Simultaneous Optimistic Optimization (`MO-SOO`) algorithms, respectively. `MO-DIRECT` partitions the search space into a set of shrinking hyperrectangles to find a good approximation set of the Pareto front. Likewise, `MO-SOO` partitions the search space to find a good approximation set of the Pareto front, but it does so at one scale at a time. At each step, `MO-SOO` expands leaf nodes of its space-partitioning tree that may optimistically contain *Pareto optimal* solutions. The main difference between the two algorithms is how the global search (exploration) is considered.

5.1.1 Multi-Objective DIviding RECTangles

In this section, we first describe a possible extension of `DIRECT` [56] to accommodate multiple objectives settings and then present the `MO-DIRECT` algorithm.

5.1.1.1 From Single- to Multi-Objective Optimization

The simplest way to approach multi-objective optimization is through aggregating the objectives into a single function (preference-based techniques). Despite the approach's simplicity, finding a set of trade-off solutions on the Pareto front requires multiple runs with different scalarization weights, yet there is no guarantee that a good approximation of the Pareto front is obtained [173]. Though preference-based techniques can be incorporated into `DIRECT`; from the concept of potentially-optimal hyperrectangles (Definition 1), one can observe that the algorithm naturally fits ideal-based approach: exploring multiple solutions at once. In essence, `DIRECT` is projecting a set of hyperrectangles $\mathcal{H}$ (more specifically, the solutions represented by their centers) into the 2-dimensional space of function values and hyperrectangle sizes. It then partitions hyperrectangles that are non-dominated with regard to *convex* Pareto optimality (see Fig. 2.3) to look for potentially better solutions. Likewise, solutions of an MOP can be projected into the $(m+1)$-dimensional space of the m-function space and hyperrectangles sizes. However, in MOPs, the interest is to find solutions that are Pareto optimal in the function space and not necessarily convex. Hence, a possible alternative of Definition 1 for an MOP is define potentially-optimal hyperrectangles as *the set of hyperrectangles that are non-dominated in the* $(m+1)$*-dimensional aforementioned space.* Note that now each hyperrectangle is represented by an $(m+1)$-point, where it is desired to minimize the first m values (objectives) in accordance with Problem (2.8) and maximize the $(m+1)$th value (size) in accordance with `DIRECT`'s framework (Fig. 2.3). Adding the hyperrectangle size as an additional objective may be both beneficial and obstructive in solving the problem at hand as shown by Brockhoff et al. [214], but it essentially provides a better exploration of the search space by favoring larger unexplored regions. In the single-objective setting, `DIRECT` trades off exploration vs. exploitation through the ϵ-constraint (2.5). This still can be simulated in MOPs by constraining the set of hyperrectangle that can be partitioned according to their sizes (e.g., a sliding/growing window over the range of size values or a threshold limit). Mathematically, given a set of hyperrectangles $\mathcal{H}$, the set of potentially-optimal hyperrectangles $\mathcal{I}$ can be defined as:

$$\mathcal{I} = \texttt{ND}(\{(\mathbf{f}(\mathbf{c}_i), \sigma_i) : i \in \mathcal{H}\ , \sigma_i \geq \sigma_t\})\ , \tag{5.1}$$

where $(\mathbf{f}(\mathbf{c}_i), \sigma_i)$ is the $(m+1)$-dimensional vector of hyperrectangle $i \in \mathcal{H}$ of center $\mathbf{c}_i$ and size σ_i, σ_t is the minimum size a hyperrectangle can have to be considered for potential optimality, and $\texttt{ND}(\cdot)$ is an operator on a set of vectors A such that $\texttt{ND}(A)$ is the set of non-dominated vectors in A, as defined next.

Definition 20 (The non-dominated operator $\texttt{ND}(\cdot)$)**.** *Let $A \subseteq \mathcal{Y}$ be a set of objective vectors. The operator $\texttt{ND}(\cdot)$ is defined such that $\texttt{ND}(A)$ is the set of all non-dominated vectors in A, i.e.,*

$$\texttt{ND}(A) \overset{\text{def}}{=} \underset{B \in \Omega, B \subseteq A}{\arg\max} |B|\ , \tag{5.2}$$

where Ω is the set of all possible approximation sets as stated by Definition 6.

One more aspect of DIRECT needs to be considered with regard to MOPs, namely the partitioning procedure (Algorithm 2). Because of the partial order on the objective vectors, the definition of w_j (2.6) does not suit the multi-objective settings. The rationale behind its definition is to increase the attractiveness of search near points with good function values by making the biggest rectangles contain the best solutions as bigger rectangles are more likely to be potentially-optimal, with everything else equal [56]. With respect to MOPs, one can define w_j to increase the *diversity/spread* of the solutions as follows:

$$w_j = \frac{1}{\min_{k\in\{1,-1\}} ||\mathbf{f}(\mathbf{c}_i + k \cdot \delta \cdot \mathbf{e}_j) - \mathbf{f}(\mathbf{c}_i)||} \; . \tag{5.3}$$

In other words, a hyperrectangle is divided such that the biggest produced hyperrectangles contain the distant solutions from that of the hyperrectangle, increasing the likelihood of visiting unexplored regions of the function space. Now, we are ready to present formally the MO-DIRECT algorithm, a multi-objective algorithmic instance of the dividing rectangles framework.

5.1.1.2 MO-DIRECT Algorithm

MO-DIRECT differs from DIRECT in two basic aspects: the criterion of potentially-optimal hyperrectangle (Eq. (5.1) instead of Eq. (2.5) & Eq. (2.4)) and the definition of w_j in Algorithm 2 (Eq. (5.3) rather than Eq. (2.6)). Furthermore, MO-DIRECT returns a set of non-dominated solutions, that is the approximation set from its samples: $\mathtt{ND}\big(\{\mathbf{f}(\mathbf{c}_i)\}_{i\in\mathcal{H}}\big)$. The pseudo-code of MO-DIRECT is shown in Algorithm 8 summarizing the concepts and the adaptations to MOPs presented earlier.

5.1.2 Multi-Objective Simultaneous Optimistic Optimization

This section describes a possible extension of SOO [34] to accommodate multiple objectives settings and then present the MO-SOO algorithm.

5.1.2.1 From Single- to Multi-Objective Optimization

One can regard SOO [34] as a tree of multi-armed bandits, where the B-value of each arm represents an optimistic bound on the values of the objective function

Algorithm 8 `MO-DIRECT`

```
    Input: vectorial function to be minimized $\mathbf{f}$, search space $\mathcal{X}$, evaluation budget
    $v$, hyprrectangle threshold $\sigma_t$, evaluation budget $v$
    Initialization: $\mathcal{H}_1 = \{\mathcal{X}\}$
    Output: approximation set of $\min_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x})$, $\mathcal{Y}_*^v$
 1: while evaluation budget $v$ is not exhausted do
 2:     Evaluate all the new hyperrectangles $\in \mathcal{H}_t$.
 3:     $\mathcal{I}_t \leftarrow \texttt{ND}(\{(\mathbf{f}(\mathbf{c}_i), \sigma_i) : i \in \mathcal{H}_t \,, \sigma_i \geq \sigma_t\})$.
 4:     Partition the hyperrectangles in $\mathcal{I}_t$ according to the procedure outlined in
    Algorithm 2 using Eq. (5.3) instead of Eq. (2.6).
 5:     $\mathcal{H}_{t+1} \leftarrow \mathcal{H}_t \setminus \mathcal{I}_t \cup \{\mathcal{I}_t\text{'s newly generated hyperrectangles}\}$
 6:     $t \leftarrow t+1$
 7: end while
 8: return $\mathcal{Y}_*^v = \texttt{ND}\big(\{\mathbf{f}(\mathbf{c}_i)\}_{i\in\mathcal{H}_t}\big)$
```

values over tree's nodes. In an iterative manner: it assesses a set of leaf nodes of its tree on the search space $\mathcal{X}$ and selectively expands a set of them. This is in line with the generic template presented in Chapter 3 (Algorithm 6). At iteration t, `SOO`'s $\mathcal{P}_t$ is the set of leaf nodes at the depth considered at iteration t, whereas $\mathcal{Q}_t$ is at most one node $\in \mathcal{P}_t$ that satisfies the condition in Algorithm 5: line 4. On this notion of sets, the `SOO` algorithm can be extended to multi-objective settings by defining the corresponding $\mathcal{P}$ and $\mathcal{Q}$.

Algorithm 9 `MO-SOO`

```
    Input: vectorial function to be minimized $\mathbf{f}$, search space $\mathcal{X}$, partition factor
    $K$, evaluation budget $v$, maximal depth function $t \rightarrow h_{max}(t)$
    Initialization: $\mathcal{T}_1 = \{(0,0)\}$, $t \leftarrow 1$
    Output: approximation set of $\min_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x})$, $\mathcal{Y}_*^v$
 1: while evaluation budget is not exhausted do
 2:     $\mathcal{V} \leftarrow \emptyset$
 3:     for $h \leftarrow 0$ to $\min(h_{max}(t), \texttt{depth}(\mathcal{T}_t))$ do
 4:         $\mathcal{P}_t \leftarrow \{\text{leaf nodes at depth } h\}$
 5:         $\mathcal{V} \leftarrow \texttt{ND}(\mathcal{P}_t \cup \mathcal{V})$
 6:         $\mathcal{Q}_t \leftarrow \mathcal{P}_t \cap \mathcal{V}$
 7:         Expand all the nodes in $\mathcal{Q}_t$; evaluate and add to $\mathcal{T}_t$ their $K\cdot|\mathcal{Q}_t|$ children,
                $$\mathcal{T}_{t+1} = \mathcal{T}_t \cup \big( \cup_{(h,i)\in\mathcal{Q}_t} \{(h+1, i_k)\}_{1\leq k\leq K}\big)$$
 8:         $t \leftarrow t+1$
 9:     end for
10: end while
11: return $\mathcal{Y}_*^v = \texttt{ND}\big(\{\mathbf{f}(\mathbf{x}_{h,i})\}_{(h,i)\in\mathcal{T}_t}\big)$
```

5.1.2.2 The `MO-SOO` Algorithm

Based on the generic template presented in Chapter 3 (Algorithm 6), a multi-objective algorithmic instance whose aim is to recognize Pareto optimal solutions can be realized. Taking inspiration from `SOO`, we refer to it as the Multi-Objective Simultaneous Optimistic Optimization (`MO-SOO`). `MO-SOO` iteratively considers leaf nodes, one depth at a time, starting from the root. The sets $\mathcal{P}$ and $\mathcal{Q}$ are defined as follows. Denote $\mathcal{T}$'s depth considered at iteration t by h_t, we have:

- $\mathcal{P}_t \overset{\text{def}}{=} \{\text{leaf nodes at depth } h_t\}$.
- $\mathcal{Q}_t \overset{\text{def}}{=}$ the subset of nodes $\in \mathcal{P}_t$ that are non-dominated with respect to $\mathcal{P}_t$ as well as all the expanded nodes in the previous h_t iterations, based on their representative objective vectors. Finding this set is captured by the operator `ND`$(\cdot)$ defined in Definition 20.

The pseudo-code of the proposed scheme is outlined in Algorithm 9. `MO-SOO` comes with three parameters, viz. i). the partition factor K; ii). the maximal depth function $h_{max}(t)$; iii). the splitting dimension per depth. All of these parameters contribute to the algorithm exploration-vs.-exploitation trade-off. Nevertheless, as it will be shown later, $h_{max}(t)$ has the most compelling impact on `MO-SOO`'s convergence.

5.2 A Worked Example

For a better understanding of the `MO-DIRECT` and `MO-SOO` algorithms, we show their application to the following problem:

$$\begin{aligned} &\text{minimize} \quad \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ &\text{s.t.} \qquad\quad \mathbf{x} = (x_1, x_2) \in \mathcal{X} = [-1, 1]^2 \ , \end{aligned} \tag{5.4}$$

where $f_1(\mathbf{x}) = (x_1 - 0.25)^2 + (x_2 - 0.66)^2$ and $f_2(\mathbf{x}) = (x_1 + 0.25)^2 + (x_2 - 0.66)^2$. Figure 5.1 shows the convergence of `MO-DIRECT`'s approximation set A towards a sampled set (numerically-obtained) of the Pareto front at different stages of the algorithm iterations. The reader can refer to Figure 5.1, which briefly describes the first stages of the algorithm. This demo as well as the source code of the algorithm can be retrieved from the project's website: https://sites.google.com/site/modirectmops/.

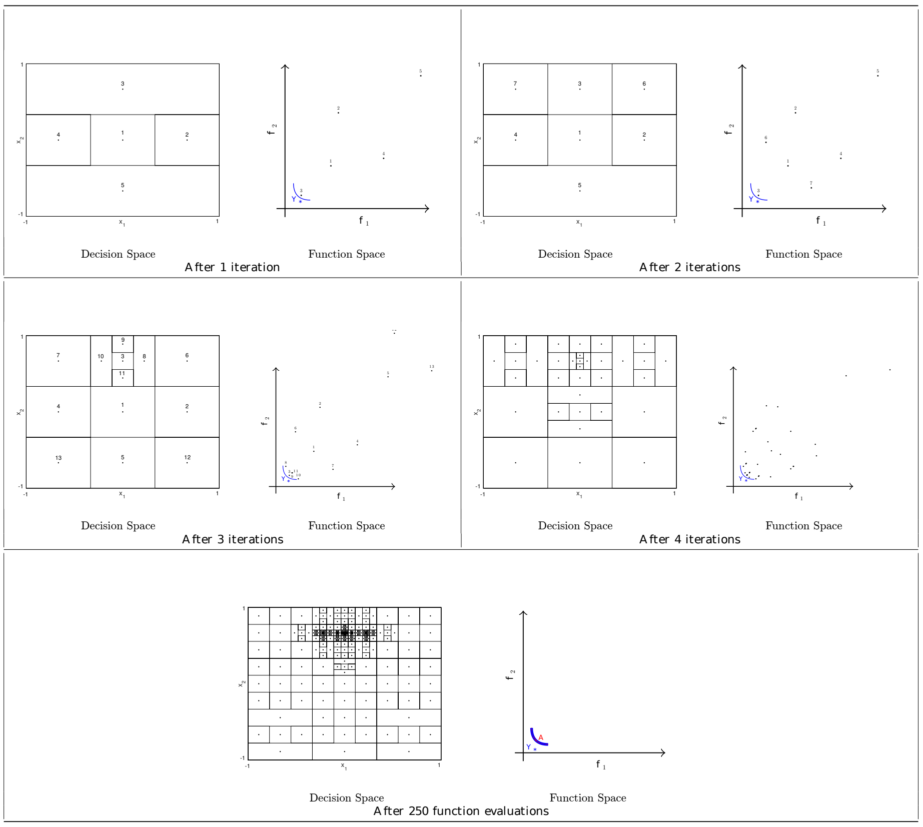


Figure 5.1: An illustration of `MO-DIRECT` (Algorithm 8). The technique identifies and expands potentially-optimal hyperrectangles to look for Pareto optimal solutions by partitioning their spaces along the decision space dimensions according to Algorithm 2, using Eq. (5.3) instead of Eq. (2.6). In each iteration t, `MO-DIRECT` selects a set of hyperrectangles whose sizes are greater than a certain value σ_t and expands those that are non-dominated in the $(m+1)$-dimensional space of the m-function space appended by the hyperrectangle size range where hyperrectangles of minimum m-function values and bigger sizes are preferred. Subsequently, one or more hyperrectangles can be expanded in one iteration; at the second iteration, for instance, only one hyperrectangle is partitioned (whose center is point 3) into smaller hyperrectangles (whose centers are the points 3, 6, and 7) as it is the only non-dominated point in $(m+1)$-dimensional space. With regard to its partitioning, it has only one dimension (x_1) of maximum side length. Hence, division takes along that dimension, generating three new hyperrectangles. On the other hand, at the third iteration two hyperrectangles are partitioned (resp., centers are the points 3 and 5) into smaller hyperrectangles (resp., centers are the points 3, 8, 9, 10, and 11; and 5, 12, and 13). Here, the division of hyperrectangle 3 takes place along the two dimensions as both sides have equal lengths; the procedure first divides along x_1 as the points 8 and 10 are farther from point 3 in the function space than the points 9 and 11. After 250 function evaluations, `MO-DIRECT`'s approximation set A closely coincides on a sampled set Y_* of the Pareto front of Problem (5.4).

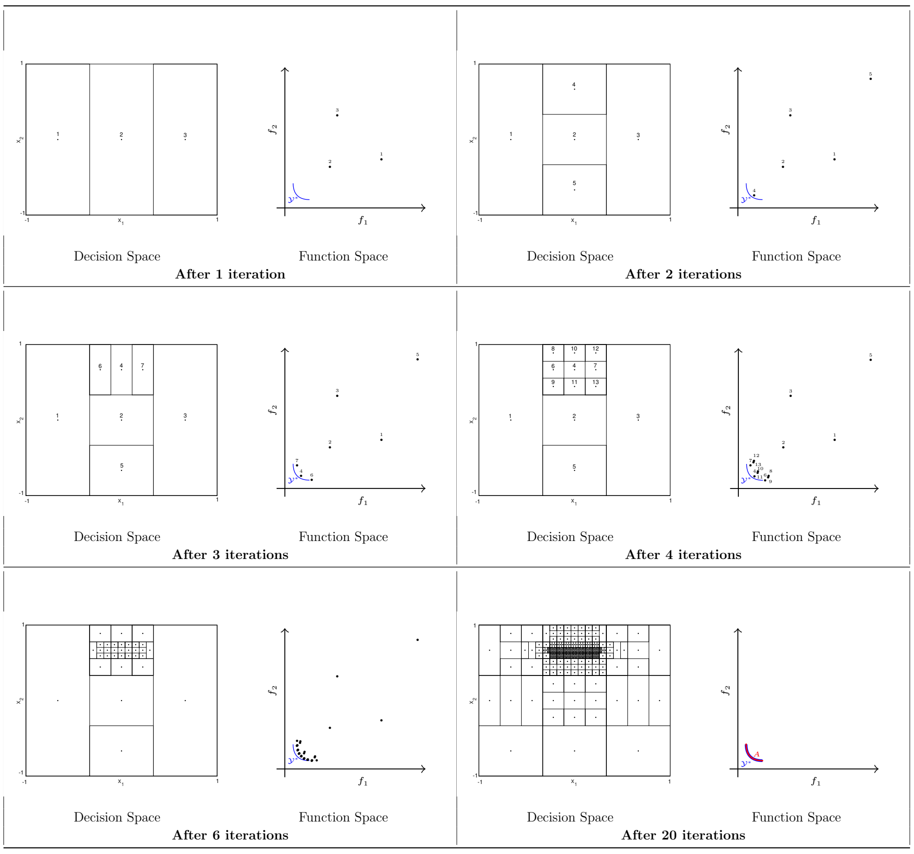


Figure 5.2: An illustration of `MO-SOO`. The algorithm expands its leaf nodes to look for Pareto optimal solutions by partitioning their cells along the decision space coordinates one at a time in a round-robin manner, with a partition factor of $K = 3$. Sweeping its tree from the root node depth till the maximal depth specified by $h_{max}(t)$. `MO-SOO` expands a set of leaf nodes per depth if they are non-dominated with respect to other leaf nodes in the same level and with respect to those expanded at lower depths in the current sweep. Subsequently, none or more nodes can be expanded in one iteration; at the third iteration, for instance, only one node is expanded (whose representative state is point 4) into its children (whose representative states are the points 4, 6, and 7). On the other hand, at the fourth iteration three nodes are expanded (resp., representative states are the points 4, 6, and 7) into their children nodes (resp., representative states are the points 4, 6, 7, 8, 9, 10, 11, 12, and 13). After 20 iterations, `MO-SOO`'s approximation set A closely coincides on a sampled set of the Pareto front $\mathcal{Y}^*$ of Problem (5.4).

On the other hand, Figure 5.2 shows the convergence of `MO-SOO`'s approximation set A towards a sampled set (numerically-obtained) of the Pareto front at different stages of the algorithm iterations—demo is available at `https://www.dropbox.com/s/1g52fexvylfxh4n/mosoo-demo.zip?dl=0`. The reader can refer to Figure 5.2 as we briefly describe the first stages of the algorithm.

Initialization. `MO-SOO` starts by initializing its tree with a root node $(0, 0)$ whose cell represents the decision space, i.e., $\mathcal{X}_{0,0} = \mathcal{X}$. The root's representative state $\mathbf{x}_{0,0} = (0,0)$—point 2 in Figure 5.2—is evaluated and $\mathbf{f}(\mathbf{x}_{0,0})$ is obtained.

Iteration 1. At this iteration, leaf nodes at depth $h = 0$ are considered for expansion. In other words, the root node is expanded by partitioning its cell along the first dimension of the decision space into $K = 3$ cells. Here, $\mathcal{P}_1 = \mathcal{Q}_1 = \{(0,0)\}$. For convenience, we shall refer to the nodes by their representative states, i.e., $\mathcal{P}_1 = \mathcal{Q}_1 = \{2\}$. The newly generated leaf nodes are added to the tree and evaluated at their representative states viz. the points 1, 2, and 3 in Figure 5.2. Note that having an odd partition factor ($K = 3$) saves one function evaluation for each node expansion (point 2 was already evaluated).

Iteration 2. At this iteration, leaf nodes at depth $h = 1$ are considered for expansion. We have $\mathcal{P}_2 = \{1,2,3\}$ and $\mathcal{V} = \{2\}$. Along with Lines 4–8 of Algorithm 9, $\mathcal{Q}_2$ becomes $\{2\}$, because point 2 dominates both points 1 and 3 as it can noted in the function space. Thus, node 2 is expanded and the tree grows to have the leaves $\mathcal{L} = \{1,\ldots,5\}$, each being evaluated at its representative state. The case is the same for *iteration 3* which considers nodes at $h = 2$ generating a new set of leaves $\mathcal{L} = \{1,\ldots,7\}$.

Iteration 4. At this iteration, leaf nodes at depth $h = 3$ are considered for expansion. Here, $\mathcal{P}_4 = \{4,6,7\}$ and $\mathcal{V} = \{4,6,7\}$. Along with Lines 4–8 of Algorithm 9, $\mathcal{Q}_4$ becomes $\{4,6,7\}$, because the points 4, 6, and 7 are non-dominated with respect to the nodes in $\mathcal{P}_4$ and $\mathcal{V}$ as it can seen in the function space. Thus, they all are expanded and the tree grows to have the leaves $\mathcal{L} = \{1,\ldots,13\}$.

Next Iterations. The same holds for the next iterations until the maximal depth—specified by $h_{max}(t)$—is reached. Then, $\mathcal{V}$ is set to $\emptyset$ and the tree is swept again from its root. After some iterations, `MO-SOO` closely approximates the Pareto front as shown in Figure 5.2.

Both algorithms provide a very simple and flexible framework to try other approaches in tackling an MOP (e.g., indicator-based or scalarization-based technique). However, they come with an increased space complexity when compared to evolutionary multi-objective algorithms, since an archive of all the samples has to be maintained rather than a population of samples. Nevertheless, this is becoming less of a concern with increasing complexity of MOPs where employed evaluation budgets are usually within practical storage limits.

5.3 Convergence Analysis

The analysis of multi-objective solvers is hindered by several issues; namely the diversity of approximation sets, the size of the Pareto front, and the convergence of approximation sets to the Pareto front [4]. While most theoretical studies have addressed finite-set and/or discrete problems [215, 216], others have provided probabilistic guarantees [217] or assumed a total order on the solutions [179]. The present section discusses theoretical aspects of the proposed techniques. While it is not difficult to prove the optimal limit behavior of the two algorithms, the finite-time analysis is complicated by the presence of multiple objectives. It is even more complicated in the case of `MO-DIRECT` as it considers—besides the conflicting objectives—the hyperrectangle size as a criterion for selection.

In the light of the above, this section is organized as follows. Section 5.3.1 discusses the asymptotic behavior of the `MO-DIRECT` algorithm. Section 5.3.2 builds on the theoretical methodology presented in Chapter 3 and studies `MO-SOO`'s finite- and asymptotic-time performance. On the one hand, the finite-time analysis establishes an upper bound on the Pareto-compliant unary additive epsilon indicator characterized by the objectives smoothness as well as the structure of the Pareto front with respect to its extrema. On the other hand, the asymptotic analysis indicates the consistency property of `MO-SOO`. Moreover, we validate the theoretical provable performance of the algorithm on a set of synthetic problems.

5.3.1 Convergence Analysis of `MO-DIRECT`

Emerging from a mathematical programming approach, `MO-DIRECT` comes with a theoretical property that it asymptotically converges to the Pareto front. Despite being practically infeasible, due to limits in available time and storage; this makes `MO-DIRECT` theoretically sounder than many MOP solvers. In line with analysis presented in [56], Theorem 9 puts formally the aforementioned proposition based on the following lemma.

Lemma 1. *As the number of iterations t goes to ∞, the fewest number of divisions $r(t)$ undergone by any hyperrectangle $i \in \mathcal{H}$ created by `MO-DIRECT` approaches ∞.*

Proof. Let us prove it by contradiction. If $\lim_{t\to\infty} r(t) < \infty$, then there must exist some iteration $\acute{t}$ after which $r(t)$ never increases, i.e., $\lim_{t\to\infty} r(t) = r(\acute{t})$. At the end of iteration $\acute{t}$, there is a finite number N of hyperrectangles in $\mathcal{H}_{\acute{t}}$ with a number of divisions equals to $r(\acute{t})$. As these hyperrectangles are the biggest in $\mathcal{H}_{\acute{t}}$, then after at most $\acute{t}+N$ iterations, all of these hyperrectangles will have been divided, as they are not dominated by any other hyperrectangle generated after the end of iteration $\acute{t}$. This implies that $r(\acute{t}+N) \geq r(\acute{t})+1$ which contradicts our assumption about $r(t)$ never increases above $r(\acute{t})$. Thus, $\lim_{t\to\infty} r(t) = \infty$. $\square$

Theorem 9. *As the number of iterations t goes to ∞, `MO-DIRECT` converges to the Pareto front of an MOP with continuous objectives.*

Proof. To simplify the proof, let the decision space $\mathcal{X} \subset \mathbb{R}^n$ be a unit hypercube. Since new hyperrectangles are created by partitioning existing ones into thirds along $\mathcal{X}$'s dimensions, a hyperrectangle $i \in \mathcal{H}$ that has been part of $r_i(t)$ partitions/divisions will have $j_i(t) = r_i(t) \bmod n$ sides of length $3^{-(k_i(t)+1)}$ and $n - j_i(t)$ sides of length $3^{-k_i(t)}$, where $\mathcal{H}$ is the set of hyperrectangles created by `MO-DIRECT` over $\mathcal{X}$ and $k_i(t) = \frac{r_i(t)-j_i(t)}{n}$. Since $r_i(t) \geq r(t)$, from Lemma 1 we have: $\lim_{t\to\infty} r_i(t) = \infty$. Thus, as the number of iterations approaches infinity, the radius $R_i(t)$ of hyperrectangle i (distance from i's center to its vertices) goes to zero, since: $\lim_{t\to\infty} R_i(t) = \lim_{t\to\infty} \frac{\sqrt{j_i(t)\cdot 3^{-2(k_i(t)+1)}+(n-j_i(t))\cdot 3^{-2k_i(t)}}}{2} = 0$. Therefore, given any $\delta > 0$, there exists $T > 0$ such that after $t > T$ iterations, all the hyperrectangles in $\mathcal{H}_t$ have a center-to-vertex distance less than δ. In other words, any point in the hypercube $\mathcal{X}$ is at a distance δ of some sampled point. Thus, if the MOP's objectives are continuous in the neighborhood of its Pareto front; `MO-DIRECT`'s approximation set will asymptotically converge to that front. $\square$

5.3.2 Convergence Analysis of `MO-SOO`

In this section, the finite-time performance of `MO-SOO` is presented besides its asymptotic convergence. The finite-time analysis of `MO-SOO` is established with respect to the Pareto-compliant quality indicator, the unary additive epsilon indicator $I^1_{\epsilon+}$ [168]. We do this in a two-step approach. First, we upper bound the loss measure introduced in Section 5.1.2.1, viz. $\mathbf{r}(v)$ of Eq. (2.15). The loss measure captures the convergence of `MO-SOO`'s approximation set $\mathcal{Y}^v_*$ to m points—on the Pareto front—that contribute to the problem's ideal point $\mathbf{y}^*$. Second, based on the presented loss bound and an intrinsic measure of the Pareto front, an upper bound on the unary additive epsilon indicator $I^1_{\epsilon+}$ is established.

In general, the design of MSO algorithms is driven by assumptions about the function smoothness. Here, we make three assumptions about the function $\mathbf{f}$ and the hierarchical partitioning in line with those presented in Chapter 3 for single-objective settings. In essence, these assumptions let us express the quality of `MO-SOO` solutions in relation to the number of iterations, by quantifying how much exploration is needed to expand nodes that contain objective-wise optimal solutions. The rest of this section is organized as follows. First, these assumptions are stated in Section 5.3.2.1. Then, in Section 5.3.2.2, the finite-time performance of `MO-SOO` is analyzed, where we first upper bound the loss (2.15) as a function of the number of iterations t.[1] Second, this objective-wise loss bound is employed to establish an upper bound on the $I^1_{\epsilon+}$ indicator. After presenting the main result on the finite-time performance of the algorithm, `MO-SOO`'s consistency property (asymptotic performance) is proved and illustrative examples are given. Towards the end of this section, an empirical validation of the theoretical findings is presented.

5.3.2.1 Assumptions

There exists a vector-valued function $\boldsymbol{\ell} : \mathcal{X} \times \mathcal{X} \rightarrow {\mathbb{R}^+}^m$ such that each entry is a semi-metric (i.e., $\{\ell_j\}_{1\leq j\leq m}$ satisfy symmetry, non-negativity, and the coincidence axiom) such that:

A5.1 **(Hölder continuity of $f_1, \ldots, f_m$):**

$$|f_j(\mathbf{x}) - f_j(\mathbf{y})| \leq \ell_j(\mathbf{x}, \mathbf{y}), \;\; \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}, j = 1, \ldots, m \;\; .$$

A5.2 **(bounded cells diameters)**: For $j = 1, \ldots, m$ and $\forall (h, i) \in \mathcal{T}$, $\exists$ a non-increasing sequence $\delta_j(h) > 0$ such that

$$\sup_{\mathbf{x}\in\mathcal{X}_{h,i}} \ell_j(\mathbf{x}_{h,i}, \mathbf{x}) \leq \delta_j(h)$$

and $\lim_{h\rightarrow\infty} \delta_j(h) = 0$. Thus, ensuring the regularity of the cells' sizes which decrease with their depths in $\mathcal{T}$.

[1] Typically, v in Eq. (2.15) and the approximation set $\mathcal{Y}^v_*$ represents the number of sampled points (function evaluations). Nevertheless, one can express the loss (and likewise the approximation set) with other growing-with-time quantities (e.g., the number of iterations, the number of node expansions). In the rest of this chapter, we refer to the number of the: function evaluations and iterations, by v and t, respectively, where one iteration represents executing the lines 4–8 of Algorithm 9, once.

A5.3 **(well-shaped cells)**: For $j = 1, \ldots, m$ and $\forall (h,i) \in \mathcal{T}$, $\exists\, s_j > 0$ such that a cell $\mathcal{X}_{h,i}$ contains an ℓ_j-ball of radius $s_j \delta_j(h)$ centered in $\mathbf{x}_{h,i}$. Thus, ensuring that the cells' shapes are not skewed in some dimensions.

5.3.2.2 Finite-Time Performance

In this section, we characterize the finite-time performance of `MO-SOO` in terms of the Pareto-compliant unary additive epsilon indicator based on the assumptions presented in Section 5.3.2.1. To this end, we upper bound the loss measure (2.15) with respect to the number of iterations t. This provides the basis upon which a bound for the ϵ-indicator is established with respect to the same.

Bounding the Loss Measure. In order to derive a bound on the loss, we employ the near-optimality dimension—defined in Chapter 3—as a measure of the quantity of objective-wise near-optimal solutions (states in $\mathcal{X}$). For the sake of completeness, we redefine the near-optimality dimension and revise some terminology, besides the terminology of Section 2.4.2.

For $j = 1, \ldots, m$; and for any $\epsilon > 0$; let us denote the set of ϵ-optimal *states* according to f_j, $\{\mathbf{x} \in \mathcal{X} : f_j(\mathbf{x}) \leq f_j(\mathbf{x}_j^*) + \epsilon\}$, by $\mathcal{X}_j^{\epsilon}$, as depicted in Figure 5.4. Subsequently, denote the set of *nodes* at depth h whose representative states are in $\mathcal{X}_j^{\delta_j(h)}$ by $\mathcal{I}_j^h$, i.e., $\mathcal{I}_j^h \overset{\text{def}}{=} \{(h,i) \in \mathcal{T} : 0 \leq i \leq K^h - 1, \mathbf{x}_{h,i} \in \mathcal{X}_j^{\delta_j(h)}\}$. A node (h,i) is Pareto optimal $\iff \exists \mathbf{x} \in \mathcal{X}^* : \mathbf{x} \in \mathcal{X}_{h,i}$. Furthermore, a Pareto optimal node (h,i) is j-optimal $\iff$ it is optimal with respect to f_j, i.e., $\mathbf{x}_j^* \in \mathcal{X}_{h,i}$. After t iterations, one can denote the depth of the deepest expanded j-optimal node by $h_{j,t}^*$ (as illustrated in Figure 5.3).

Now, we define the near-optimality dimension for f_j:

Definition 21 (s_j-near-optimality dimension)**.** *The s_j-near-optimality dimension for*

$\{f_j\}_{1 \leq j \leq m}$ is the smallest $d_{s_j} \geq 0$ such that there exists $C_j > 0$ and for any $\epsilon > 0$, the maximal number of disjoint ℓ_j-balls of radius $s_j\epsilon$ and center in $\mathcal{X}_j^{\epsilon}$ is less than $C_j \epsilon^{-d_{s_j}}$.

One can note that d_{s_j} is characterized by: the function f_j, the semi-metric ℓ_j, and the scaling factor s_j, i.e., it depends on the objectives smoothness and relates

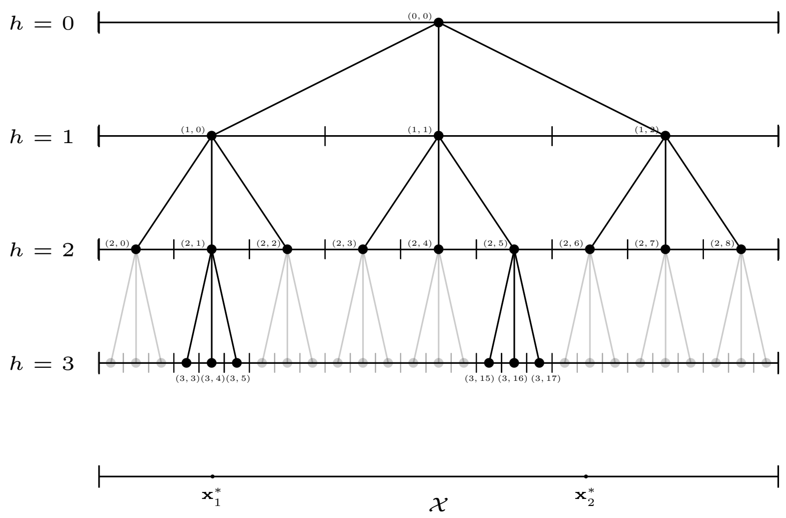


Figure 5.3: Hierarchical partitioning of the decision space $\mathcal{X}$ with a partition factor of $K = 3$ at iteration t represented by a K-ary tree. Consider a multi-objective problem where $m = 2$ and the global optimizers of the first and second objective functions ($\mathbf{x}_1^*$ and $\mathbf{x}_2^*$, respectively) are shown. Thus, the nodes $\{(3,4),(2,1),(1,0),(0,0)\}$ and $\{(2,6),(1,2),(0,0)\}$ are 1- and 2-optimal nodes, respectively. Furthermore, $h^*_{1,t} = 3$ and $h^*_{2,t} = 2$.

to the partitioning strategy of the space through the scaling factor s_j. Based on Assumption A5.3 and Definition 21, we have:

$$|\mathcal{I}_j^h| \leq C_j \delta_j(h)^{-d_{s_j}} \; .^{2} \tag{5.5}$$

Now, let us assume for simplicity that the $\mathtt{ND}(\cdot)$ operator in Algorithm 9 is replaced by $\mathtt{ND}_{min}(A) = \cup_{1\leq j\leq m} \arg\min_{\mathbf{y}\in A} y_j$; that is to say, in each iteration, m or less nodes are expanded whose representative objective vectors $\mathbf{f}(\mathbf{x}_{h,i})$ have the minimum entries with respect to the m objectives. Furthermore, for $j = 1,\dots,m$; assume that $h^*_{j,t} = \acute{h}$ and denote the j-optimal node at depth $\acute{h}+1$ by $(\acute{h}+1, j^*)$. Since $(\acute{h}+1, j^*)$ has not been expanded yet, any node at depth $\acute{h}+1$ that is selected at later iterations and expanded before $(\acute{h}+1, j^*)$ (line 7 in Algorithm 9) must satisfy the following:

$$\begin{aligned} f_j(\mathbf{x}_{\acute{h}+1,i}) &\leq f_j(\mathbf{x}_{\acute{h}+1,j^*}) \\ f_j(\mathbf{x}_{\acute{h}+1,i}) &\leq f_j(\mathbf{x}_j^*) + \delta_j(\acute{h}+1) \end{aligned} \tag{5.6}$$

[2] From Lemma 1.

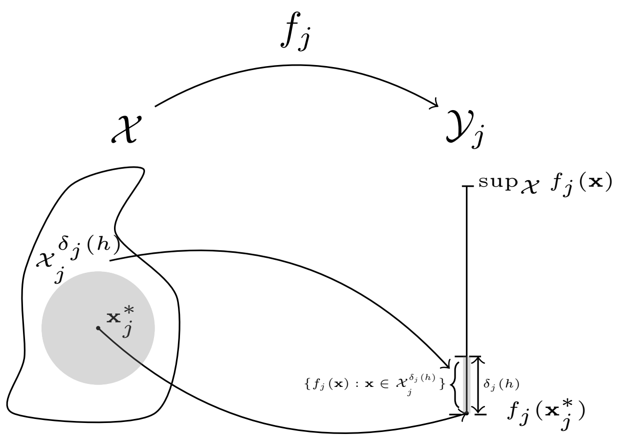
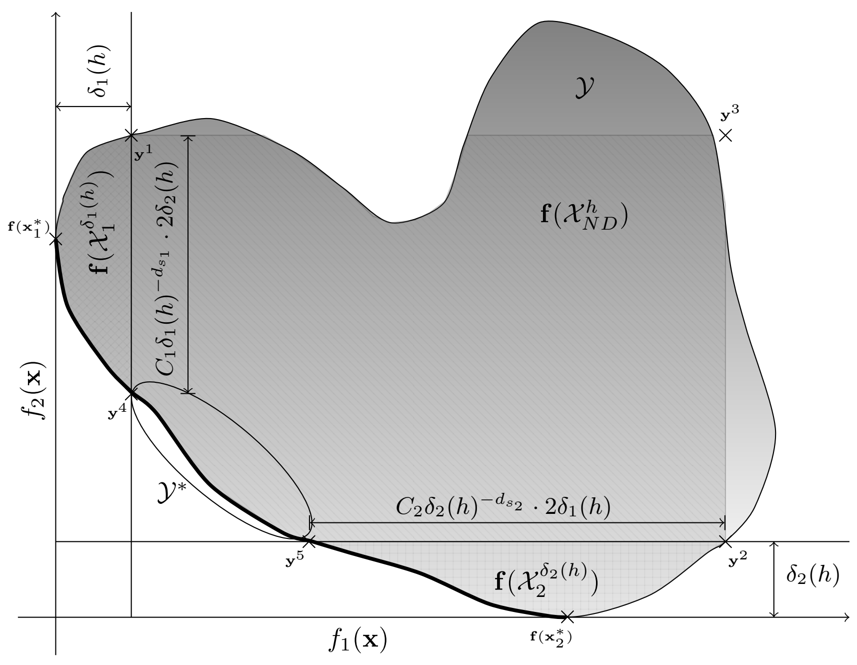

f_j
$\mathcal{X}$
$\mathcal{Y}_j$
$\sup_{\mathcal{X}} f_j(\mathbf{x})$
$\mathcal{X}_j^{\delta_j(h)}$
$\mathbf{x}_j^*$
$\{f_j(\mathbf{x}) : \mathbf{x} \in \mathcal{X}_j^{\delta_j(h)}\}$
$\delta_j(h)$
$f_j(\mathbf{x}_j^*)$

Figure 5.4: The feasible decision space and the corresponding jth objective space ($\mathcal{Y}_j \subseteq \mathbb{R}$). The global optimizer $\mathbf{x}_j^*$ and any solution $\mathbf{x}$ whose image under the jth objective lies within $\{f_j(\mathbf{x}) \leq f_j(\mathbf{x}_j^*) + \delta_j(h)\}$ are denoted by $\mathcal{X}_j^{\delta_j(h)}$.

where inequality (5.6) comes from combining Assumptions A5.1 and A5.2: $f_j(\mathbf{x}_{\acute{h}+1,j^*}) \leq f_j(\mathbf{x}_j^*) + \ell_j(\mathbf{x}_{\acute{h}+1,j^*}, \mathbf{x}_j^*) \leq f_j(\mathbf{x}_j^*) + \delta_j(\acute{h}+1)$. As defined earlier, $\mathcal{X}_j^{\delta_j(h)}$ satisfies Eq. (5.6) (depicted in Figure 5.5, for $m = 2$). Thus, from the definition of $\mathcal{I}_j^h$ and since all the objectives are considered simultaneously, we are certain that $\{(\acute{h}+1, j^*)\}_{1\leq j\leq m}$ get expanded after $\sum_{j=1}^m |\mathcal{I}_j^{\acute{h}+1}|$ node expansions at depth $\acute{h}+1$ in the worst-case scenario. Nevertheless, such definition of the $\texttt{ND}_{min}(\cdot)$ operator favors exploring $\{\mathcal{X}_j^{\delta_j(h)}\}_{1\leq j\leq m}$ over other regions, which delays the search for other Pareto points outside these regions (see, for instance, the circled region in Figure 5.5). Using the $\texttt{ND}(\cdot)$ operator from Definition 20 rectifies this behavior: by expanding non-dominated nodes, MO-SOO explores as well the region $\{\mathbf{x} : \mathbf{f}(\mathbf{x}) \prec \mathbf{y}^{nadir}(\mathbf{f}(\cup_{j=1,2}\mathcal{X}_j^{\delta_j(h)}))\} - \cup_{j=1,2}\mathcal{X}_j^{\delta_j(h)}$ denoted by $\mathcal{X}_{ND}^h$ (see Figure 5.5). While we are able to quantify—based on the near-optimality dimension—the number of nodes within $\{\mathcal{X}_j^{\delta_j(h)}\}_{1\leq j\leq m}$, similar analysis gets unnecessarily complicated for $\mathcal{X}_{ND}^h$. However, since $\texttt{ND}(\cdot)$ expands—besides other nodes—the same set of nodes that would have been selected by $\texttt{ND}_{min}(\cdot)$, we know that at most $|\mathcal{I}_j^{\acute{h}+1}|$ iterations at depth $\acute{h}+1$ are needed to expand the optimal node $(\acute{h}+1, j^*)$. From this observation, the following lemma is deduced.

Lemma 2. *In MO-SOO, after t iterations, for any depth $0 \leq h \leq h_{max}(t)$ whenever*

$$h_{max}(t) \sum_{l=0}^{h} \max_{1\leq j\leq m} |\mathcal{I}_j^l| \leq t \; , \tag{5.7}$$



$\delta_1(h)$
$\mathcal{Y}$
$\mathbf{y}^3$
$\mathbf{y}^1$
$\mathbf{f}(\mathbf{x}_1^*)$
$\mathbf{f}(\mathcal{X}_1^{\delta_1(h)})$
$C_1\delta_1(h)^{-d_{s_1}} \cdot 2\delta_2(h)$
$\mathbf{f}(\mathcal{X}_{ND}^h)$
$f_2(\mathbf{x})$
$\mathbf{y}^4$
$\mathcal{Y}^*$
$C_2\delta_2(h)^{-d_{s_2}} \cdot 2\delta_1(h)$
$\mathbf{y}^5$
$\mathbf{y}^2$
$\delta_2(h)$
$\mathbf{f}(\mathcal{X}_2^{\delta_2(h)})$
$f_1(\mathbf{x})$
$\mathbf{f}(\mathbf{x}_2^*)$

Figure 5.5: The objective space $\mathcal{Y}$ for a multi-objective optimization problem ($m = 2$). The solid curve marks the Pareto front $\mathcal{Y}^*$. At depth h, assuming the $\{j\}_{j=1,2}$-optimal nodes are not expanded yet, one can use the $\texttt{ND}_{min}(\cdot)$ operator, which causes nodes whose representative states lie in the decision space portion $\cup_{j=1,2}\mathcal{X}_j^{\delta_j(h)}$ to be expanded before others. However, this may hold up discovering other parts of the Pareto front (the circled region). This is not the case with the $\texttt{ND}(\cdot)$ operator, where another region $\mathcal{X}_{ND}^h$—whose image in the objective space is bounded by the Pareto front and the points $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3 = \mathbf{y}^{nadir}(\mathbf{f}(\cup_{j=1,2}\mathcal{X}_j^{\delta_j(h)}))$—is as well considered. Let the considered depth at iteration t be h and the depth of the deepest $\{j\}_{j=1,2}$-optimal nodes be $h-1$. Then, prior to expanding the $\{j\}_{j=1,2}$-optimal nodes at depth h, the set $\mathcal{Q}_t$ (of Algorithm 9) comprises of at most 3 types of nodes whose representative states lie in $\mathcal{X}_1^{\delta_1(h)}, \mathcal{X}_2^{\delta_2(h)}$, and $\mathcal{X}_{ND}^h$, respectively. Furthermore, one can note from Eq. (5.5) as well as Assumptions A5.1 and A5.2 that the point $\mathbf{y}^1$ is greater than or equal $\mathbf{y}^4$, and hence $\mathbf{f}(\mathbf{x}_1^*)$, along f_2 by at most $C_1\delta_1(h)^{-d_{s_1}} \cdot 2\delta_2(h)$, that is to say $y_2^1 - f_2(\mathbf{x}_1^*) \leq C_1\delta_1(h)^{-d_{s_1}} \cdot 2\delta_2(h)$; similar argument can be made between the points $\mathbf{y}^2$ and $\mathbf{y}^5$ along f_1. This observation is the main ingredient in the proof of Theorem 11.

we have $\{h_{j,t}^*\}_{1\leq j\leq m} \geq h$.

Proof. We know that $\{h_{j,t}^*\}_{1\leq j\leq m} \geq 0$ and hence the above statement holds for $h = 0$. For $0 < h \leq h_{max}(t)$, we are going to prove it by induction.

Assume that the statement holds for $0 \leq h \leq \hat{h} < h_{max}(t)$. Let us then prove it for $h \geq \hat{h}+1$. Let $h_{max}(t)\sum_{l=0}^{\hat{h}+1}\max_{1\leq j\leq m}|\mathcal{I}_j^l| \leq t$, and hence, $h_{max}(t)\sum_{l=0}^{\hat{h}}\max_{1\leq j\leq m}|\mathcal{I}_j^l| \leq$

t for which we know by our assumption that $\{h^*_{j,t}\}_{1\leq j\leq m} \geq \hat{h}$. Here, we have two cases: (i) $\{h^*_{j,t}\}_{1\leq j\leq m} \geq \hat{h}+1$, for which the proof is done; (ii) $\{h^*_{j,t}\}_{1\leq j\leq m} = \hat{h}$, for this case, the set of nodes expanded at depth $\hat{h}+1$ at each iteration, before the $\{j\}_{1\leq j\leq m}$-optimal nodes at the same depth, belong to $m+1$ sets (possibly overlapped) of nodes. Among theses sets, m sets are from $\{\mathcal{I}^{\hat{h}+1}_j\}_{1\leq j\leq m}$, respectively; while the remaining set of nodes have their representative states in $\mathcal{X}^{\hat{h}+1}_{ND} - \cup_{j=1}^{m}\mathcal{X}^{\delta_j(\hat{h}+1)}_j$. As a result, at each iteration, there could be at least one node to be expanded from $\{\mathcal{I}^{\hat{h}+1}_j\}_{1\leq j\leq m}$, respectively. Since expanding all of these nodes takes at most $\max_{1\leq j\leq m}|\mathcal{I}^{\hat{h}+1}_j|$ iterations at depth $\hat{h}+1$; with a tree of depth $h_{max}(t)$, we are certain that the $\{j\}_{1\leq j\leq m}$-optimal node at depth $\hat{h}+1$ are expanded after at most $h_{max}(t)\max_{1\leq j\leq m}|\mathcal{I}^{\hat{h}+1}_j|$ iterations. Therefore, we have $\{h^*_{j,t}\}_{1\leq j\leq m} \geq h$. □

In other words, the size of $\mathcal{I}^h_j$ gives a measure of how much exploration is needed, provided that the j-optimal node at depth $h-1$ has been expanded; and this exploration is quantified by the near-optimality dimension. The next theorem builds on Lemma 2 to present a finite-time analysis of `MO-SOO` in terms of a bound on the loss of Eq. (2.15) as a function of the number of iterations t.

Theorem 10 ($\mathbf{r}(t)$ for `MO-SOO`). *Let us define $h(t)$ as the smallest $h\geq 0$ such that:*

$$h_{max}(t)\sum_{l=0}^{h(t)}\max_{1\leq j\leq m} C_j\delta_j(l)^{-d_{s_j}} \geq t \tag{5.8}$$

where t is the number of iterations. Then the loss of `MO-SOO` is bounded as:

$$r_j(t) \leq \delta_j(\min(h(t), h_{max}(t)+1))\ ,\ j=1,\ldots,m\ . \tag{5.9}$$

Proof. Since $|\mathcal{I}^h_j| \leq C_j\delta_j(h)^{-d_{s_j}}$ from Eq. (5.5); from the definition of $h(t)$ (5.8), we have:

$$h_{max}(t)\sum_{l=0}^{h(t)-1}|\mathcal{I}^l_j| \leq h_{max}(t)\sum_{l=0}^{h(t)-1}\max_{1\leq j\leq m} C_j\delta_j(l)^{-d_{s_j}} < t$$

Thus, from Lemma 2 and since $h_{max}(t)$ is the maximum depth at which nodes can be expanded, we have $\{h^*_{j,t}\}_{1\leq j\leq m} \geq \min(h(t)-1, h_{max}(t))$. Now, let $(h^*_{j,t}+1, j^*)$ be the deepest non-expanded j-optimal node (which is a child node of the deepest expanded j-optimal node at depth $h^*_{j,t}$ and its representative state $\mathbf{x}_{h^*_{j,t}+1,j^*}$ has been evaluated), then the loss with respect to the jth objective is bounded, based on Assumption A5.2, as:

$$r_j(t) \leq f(\mathbf{x}_{h^*_{j,t}+1,j^*}) - f(\mathbf{x}^*_j) \leq \delta_j(h^*_{j,t}+1)\;.$$

Since $\{h^*_{j,t}\}_{1\leq j\leq m} \geq \min(h(t)-1, h_{max}(t))$, we have $r_j(t) \leq \delta_j(\min(h(t), h_{max}(t)+1))$, for $j = 1, \ldots, m$. $\square$

Bounding the Additive Epsilon Indicator. Within the context of multi-objective optimization and after t iterations, the vectorial loss $\mathbf{r}(t)$ of Eq. (2.15) does not explicitly capture the quality of `MO-SOO`'s approximation set $\mathcal{Y}^t_*$ with respect to the whole Pareto front $\mathcal{Y}^*$. Here, we investigate whether there is an implicit connection between the two concepts. Particularly, we study the relationship between $\mathbf{r}(t)$ (as well as its bound of Eq. 5.9) and the Pareto-compliant additive ϵ-indicator of `MO-SOO`'s approximation set $\mathcal{Y}^t_*$ with respect to the Pareto front $\mathcal{Y}^*$ (or the unary additive ϵ-indicator of $\mathcal{Y}^t_*$): $I^1_{\epsilon+}(\mathcal{Y}^t_*)$. In essence, $I^1_{\epsilon+}(\mathcal{Y}^t_*)$ measures the smallest amount ϵ needed to translate each element in the Pareto front $\mathcal{Y}^*$ such that it is *weakly* dominated by at least one element in the approximation set $\mathcal{Y}^t_*$. This notion is put formally in Definition 7.

A negative value of $I_{\epsilon+}(A, B)$ indicates that A strictly dominates B: every element in B is strictly dominated by at least one element in A. Note that $I^1_{\epsilon+}(\mathcal{Y}^t_*) \overset{\text{def}}{=} I_{\epsilon+}(\mathcal{Y}^t_*, \mathcal{Y}^*) \geq 0$ as no element in $\mathcal{Y}^t_*$ strictly dominates any element in $\mathcal{Y}^*$. Thus, the closer $I^1_{\epsilon+}(\mathcal{Y}^t_*)$ to 0, the better the quality of $\mathcal{Y}^t_*$. Figure 5.6 illustrates the two quantities, viz. $\mathbf{r}(t)$ and $I^1_{\epsilon+}(\mathcal{Y}^t_*)$, and highlights their explicit relationship for $m = 2$. From this observation, the following lemma is deduced.

Lemma 3. *For any MOP solver, we have* $I^1_{\epsilon+}(\mathcal{Y}^t_*) \geq \max_{1\leq j\leq m} r_j(t)$.

Proof. From the definition of the vectorial loss measure (2.15), the m closest elements on the approximation set $\mathcal{Y}^t_*$ to the m extrema of the Pareto front $\mathcal{Y}^*$—i.e., $\{\mathbf{f}(\mathbf{x}^*_j)\}_{1\leq j\leq m}$—differ by $\{r_j(t)\}_{1\leq j\leq m}$ along the corresponding jth objective, respectively. Therefore, an objective-wise translation of at least $\max_{1\leq j\leq m} r_j(t)$ is needed so as each of the translated Pareto front extrema is *weakly* dominated by at least one element in the approximation set $\mathcal{Y}^t_*$. Thus, from Definition 7, $I^1_{\epsilon+}(\mathcal{Y}^t_*) \geq \max_{1\leq j\leq m} r_j(t)$. $\square$

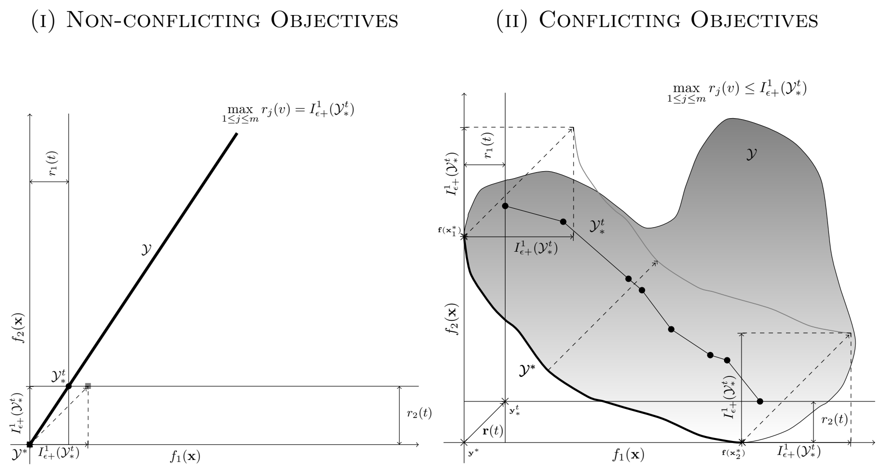

(i) Non-conflicting Objectives
(ii) Conflicting Objectives
$\max_{1\leq j\leq m} r_j(v) = I^1_{\epsilon+}(\mathcal{Y}^t_*)$
$\max_{1\leq j\leq m} r_j(v) \leq I^1_{\epsilon+}(\mathcal{Y}^t_*)$
$\mathcal{Y}$
$\mathcal{Y}^t_*$
$\mathcal{Y}^*$
$r_1(t)$
$r_2(t)$
$\mathbf{r}(t)$
$f_1(\mathbf{x})$
$f_2(\mathbf{x})$
$I^1_{\epsilon+}(\mathcal{Y}^t_*)$
$\mathbf{f}(\mathbf{x}^*_1)$
$\mathbf{f}(\mathbf{x}^*_2)$
y^t_*
y^*

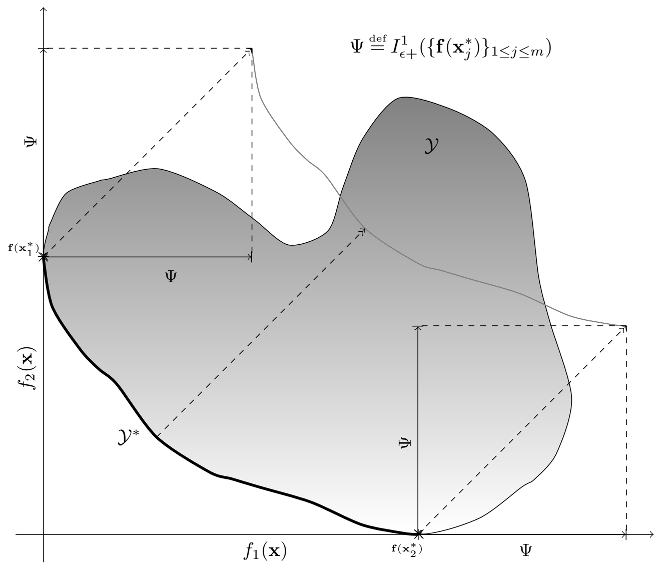
Figure 5.6: Illustration of the vectorial loss $\mathbf{r}(t)$ of Eq. (2.15) and its relation to the unary additive epsilon indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$ (Definition 7) for a multi-objective problem ($m = 2$) whose objective space $\mathcal{Y}$ is shown in two scenarios: (i) non-conflicting objectives and (ii) conflicting objectives. The faded square (resp., curve) in the first (resp., second) scenario represents the least-translated Pareto front so as every translated element is weakly dominated by at least one element in the approximation set $\mathcal{Y}^t_*$, i.e., $\{(y_1 + I^1_{\epsilon+}(\mathcal{Y}^t_*), \ldots, y_m + I^1_{\epsilon+}(\mathcal{Y}^t_*))\}_{\mathbf{y}\in\mathcal{Y}^*}$. Mathematically, $I^1_{\epsilon+}(\mathcal{Y}^t_*) \geq \max_{1\leq j\leq m} r_j(t)$ where equality sufficiently holds when $|\mathcal{Y}^t_*| = 1$ (see Lemma 3).

While Lemma 3 provides a lower bound on the indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$, one is more interested in an upper bound so as to capture the convergence of the approximation set to the whole Pareto front. To this end, we propose a measure of conflict of the Pareto front extrema with respect to the rest of its elements, called *conflict dimension.*

Definition 22. *(conflict dimension) The conflict dimension $\Psi \geq 0$ for an MOP with m objectives and Pareto front $\mathcal{Y}^*$ is the unary additive epsilon indicator of the approximation set that consists of the extrema of $\mathcal{Y}^*$ (m or less elements). Mathematically:*

$$\Psi \overset{\text{def}}{=} I^1_{\epsilon+}(\{\boldsymbol{f}(\boldsymbol{x}^*_j)\}_{1\leq j\leq m}) \overset{\text{def}}{=} I_{\epsilon+}(\{\boldsymbol{f}(\boldsymbol{x}^*_j)\}_{1\leq j\leq m}, \mathcal{Y}^*) \tag{5.10}$$

Figure 5.7 illustrates the proposed measure. Note that Ψ is an intrinsic property of the MOP's Pareto front $\mathcal{Y}^*$. In essence, the conflict dimension Ψ captures the proximity of Pareto front extrema to the rest of its elements, where $\Psi = 0 \iff |\mathcal{Y}^*| \leq m$. We now provide our upper bound on the indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$.



$\Psi \overset{\text{def}}{=} I^1_{\epsilon+}(\{\mathbf{f}(\mathbf{x}^*_j)\}_{1\leq j\leq m})$
Ψ
$\mathcal{Y}$
$\mathbf{f}(\mathbf{x}^*_1)$
Ψ
$f_2(\mathbf{x})$
$\mathcal{Y}^*$
Ψ
$f_1(\mathbf{x})$
$\mathbf{f}(\mathbf{x}^*_2)$
Ψ

Figure 5.7: The objective space $\mathcal{Y}$ for a multi-objective optimization problem ($m = 2$) with conflict dimension $\Psi \geq 0$ (Definition 22). The solid curve marks the Pareto front $\mathcal{Y}^*$. The faded curve represents the Ψ-translated Pareto front, i.e., $\{(y_1 + \Psi, \ldots, y_m + \Psi)\}_{\mathbf{y}\in\mathcal{Y}^*}$. Every element of the Ψ-translated Pareto front is *weakly* dominated by at least one element in the set $\{\mathbf{f}(\mathbf{x}^*_j)\}_{1\leq j\leq m} \subseteq \mathcal{Y}^*$.

Theorem 11 ($I^1_{\epsilon+}(\mathcal{Y}^t_*)$ for `MO-SOO`)**.** *Let us define* $\acute{h}(t) \overset{\text{def}}{=} \min(h(t), h_{max}(t) + 1)$ *where t is the number of iterations and $h(t)$—as in Theorem 10—is the smallest $h \geq 0$ such that Eq. (5.8) holds. Then for an MOP with conflict dimension Ψ, the indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$ of `MO-SOO` is bounded as:*

$$I^1_{\epsilon+}(\mathcal{Y}^t_*) < \Psi + \max_{1\leq k,l\leq m} (1 + 2C_k\delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t)) \;. \tag{5.11}$$

Proof. From the loss bound (5.9) established in Theorem 10, `MO-SOO`'s approximation set after t iterations $\mathcal{Y}^t_*$ lies in a portion of the function space, viz. $\{\mathbf{f}(\mathcal{X}_1^{\delta_1(\acute{h}(t))}), \ldots, \mathbf{f}(\mathcal{X}_m^{\delta_m(\acute{h}(t))})$ and possibly $\mathbf{f}(\mathcal{X}^{\acute{h}(t)}_{ND})$ (defined before Lemma 2 in Section 5.3.2.2 and depicted in Figure 5.5, for $m = 2$). Therefore, in the worst-case scenario, $\mathcal{Y}^t_*$ consists of m (or less) elements that contribute to the nadir point of $\mathbf{f}(\mathcal{X}^{\acute{h}(t)}_{ND})$ (e.g., $\mathcal{Y}^t_* = \{\mathbf{y}^1, \mathbf{y}^2\}$ in Figure 5.5, where their objective-wise values constitute $\mathbf{y}^3$). For brevity, let us denote this worst-case approximation set and the set of the Pareto front extrema $\{\mathbf{f}(\mathbf{x}^*_j)\}_{1\leq j\leq m}$ by $\mathcal{Y}^{t,w}_*$ and $\mathcal{Y}^{*,e}$, respectively.

Now, for all $j \in \{1, \ldots, m\}$, the maximum objective-wise translation between the element $\mathcal{Y}_*^{t,w} \cap \mathbf{f}(\mathcal{X}_j^{\delta_j(\acute{h}(t))})$ (e.g., $\mathbf{y}^1$ in Figure 5.5 for $j = 1$) and $\mathbf{f}(\mathbf{x}_j^*) \in \mathcal{Y}^{*,e}$ is upper bounded as follows (see Figure 5.5 for illustration).

$$\begin{aligned}
\max_{\substack{1 \leq \bar{j} \leq m \\ \mathbf{y}^1 \in \mathcal{Y}_*^{t,w} \cap \mathbf{f}(\mathcal{X}_j^{\delta_j(\acute{h}(t))})}} y_{\bar{j}}^1 - f_{\bar{j}}(\mathbf{x}_{\bar{j}}^*) &\leq \max\left(\max_{\substack{1 \leq l \leq m \\ l \neq j}} \overbrace{C_j \delta_j(\acute{h}(t))^{-d_{s_j}}}^{\text{from Eq. (5.5)}} \cdot \underbrace{2\delta_l(\acute{h}(t))}_{\text{from A5.1 and A5.2}} , \overbrace{\delta_j(\acute{h}(t))}^{\text{from Eq. (5.9)}} \right) \\
&\leq \max_{1 \leq k \leq m} \left(\max\left(\max_{\substack{1 \leq l \leq m \\ l \neq k}} 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}} \delta_l(\acute{h}(t)), \delta_k(\acute{h}(t)) \right) \right) \\
&< \max_{1 \leq k, l \leq m} (1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t)) \ . \qquad (5.12)
\end{aligned}$$

Put it differently, elements in $\mathcal{Y}_*^{t,w}$ differ objective-wise by a value less than the right-hand side of (5.12) with respect to their corresponding closest elements in $\mathcal{Y}_*^e$. On the other hand, Definition 22 implies that there exists at least one element $\mathbf{y}^2 \in \mathcal{Y}^{*,e}$ for every element $\mathbf{y}^3 \in \mathcal{Y}^*$ such that $\mathbf{y}^2 \preceq_\Psi \mathbf{y}^3$. Consequently, we have

$$y_j^2 + \max_{1 \leq k, l \leq m} (1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t)) \leq \Psi + \max_{1 \leq k, l \leq m} (1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t)) + y_j^3 \ , \qquad (5.13)$$

for all $j \in \{1, \ldots, m\}$. Combining (5.12) and (5.13) indicates that for every element $\mathbf{y}^3 \in \mathcal{Y}^*$, there exists at least one element $\mathbf{y}^1 \in \mathcal{Y}_*^{t,w}$ such that

$$y_j^1 < \Psi + \max_{1 \leq k, l \leq m} (1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t)) + y_j^3 \ , \qquad (5.14)$$

for all $j \in \{1, \ldots, m\}$. i.e., $I_{\epsilon+}^1(\mathcal{Y}_*^{t,w}) \leq \Psi + \max_{1 \leq k, l \leq m} (1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t))$. Since $\max_{\mathbf{y}^4 \in \mathcal{Y}_*^t} y_j^4 \leq \max_{\mathbf{y}^1 \in \mathcal{Y}_*^{t,w}} y_j^1$ for all $j \in \{1, \ldots, m\}$, we have

$$I_{\epsilon+}^1(\mathcal{Y}_*^t) < \Psi + \max_{1 \leq k, l \leq m} (1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t)) \ .$$

□

Theorem 11 characterizes the bound on $I_{\epsilon+}^1(\mathcal{Y}_*^t)$ by a non-increasing function $\max_{1 \leq k, l \leq m}$ $(1 + 2C_k \delta_k(\acute{h}(t))^{-d_{s_k}}) \cdot \delta_l(\acute{h}(t))$ reflecting the objectives smoothness with an offset dependent on the structure of the Pareto front with respect to its extrema $\{\mathbf{f}(\mathbf{x}_j^*)\}_{1 \leq j \leq m}$—i.e., the conflict dimension Ψ. Towards the end of this section, some illustrative examples are given about the characteristics of the non-increasing function in relation to the theoretical bounds presented. Furthermore, these theoretical bounds are calculated via *symbolic computation* and validated on a set of synthetic problems.

5.3.2.3 Asymptotic Convergence (Consistency Property)

Theorem 10 addressed the finite-time performance of `MO-SOO` with respect to m points on the Pareto front, whereas Theorem 11 established it with respect to the additive ϵ-indicator as the number of iterations t grows. Here, we consider the asymptotic convergence towards the Pareto front. An algorithm is consistent if it asymptotically converges to the Pareto front. `MO-SOO` guarantees that no portion of $\mathcal{X}$ is disregarded $\iff$ $h_{max}(t) \rightarrow \infty$ as $t \rightarrow \infty$. Accordingly, if a Pareto optimal node happens to be a leaf node at iteration $\acute{t}$, then it will definitely get expanded in one of the next iterations $\geq \acute{t}+1$. As the number of iterations t grows bigger, the base points sampled by `MO-SOO` form a dense subset of $\mathcal{X}$ such that for an arbitrary small $\epsilon \geq 0$: $\forall \acute{\mathbf{x}} \in \mathcal{X}^*, \exists$ a base point $\mathbf{x}$ such that $|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\acute{\mathbf{x}})| \leq \epsilon$. The next theorem establishes formally our proposition about the consistency property of `MO-SOO`.

Theorem 12 (`MO-SOO` Consistency)**.** *`MO-SOO` is consistent, provided that $h_{max}(t) \rightarrow \infty$ as $t \rightarrow \infty$, where t is the number of iterations.*

Proof. Let us denote the deepest Pareto optimal node that has the Pareto optimal solution $\acute{\mathbf{x}} \in \mathcal{X}^*$ by $(h_{\acute{\mathbf{x}}}(t), i_{\acute{\mathbf{x}}})$. i.e., $\acute{\mathbf{x}} \in \mathcal{X}_{h_{\acute{\mathbf{x}}}(t),i_{\acute{\mathbf{x}}}}$. From Assumption A5.2 and the definition of the semi-metric ℓ_j,

$$0 \leq \ell_j(\mathbf{x}_{h_{\acute{\mathbf{x}}}(t),i_{\acute{\mathbf{x}}}}, \acute{\mathbf{x}}) \leq \quad \delta_j(h_{\acute{\mathbf{x}}}(t)) \quad, \forall \acute{\mathbf{x}} \in \mathcal{X}^*, j = 1, \ldots, m\ .$$

Since $h_{max}(t) \rightarrow \infty$ as $t \rightarrow \infty$, the depths of all the Pareto optimal nodes tends to ∞, mathematically:

$$0 \leq \lim_{t\rightarrow\infty} \ell_j(\mathbf{x}_{h_{\acute{\mathbf{x}}}(t),i_{\acute{\mathbf{x}}}}, \acute{\mathbf{x}}) \leq \lim_{h_{\acute{\mathbf{x}}}(t)\rightarrow\infty} \delta_j(h_{\acute{\mathbf{x}}}(t)) \quad, \forall \acute{\mathbf{x}} \in \mathcal{X}^*, j = 1, \ldots, m\ .$$

Then, with Assumption A5.2:

$$\lim_{t\rightarrow\infty} \ell_j(\mathbf{x}_{h_{\acute{\mathbf{x}}}(t),i_{\acute{\mathbf{x}}}}, \acute{\mathbf{x}}) = \quad 0 \qquad\qquad , \forall \acute{\mathbf{x}} \in \mathcal{X}^*, j = 1, \ldots, m\ ,$$

and from the coincidence axiom satisfied by ℓ_j as a semi-metric:

$$\lim_{t\rightarrow\infty} \mathbf{x}_{h_{\acute{\mathbf{x}}}(t),i_{\acute{\mathbf{x}}}} = \quad \acute{\mathbf{x}} \qquad\qquad , \forall \acute{\mathbf{x}} \in \mathcal{X}^*\ .$$

Thus, as the number of iterations t grows bigger, `MO-SOO` asymptotically converges to the Pareto front. □

5.3.2.4 Illustration

In this section, insights on the loss bound (5.9) is presented and illustrated through some examples.[3] For $j = 1, \dots, m$; let $\delta_j(h) = c_j \gamma_j^h$ for some constants $c_j > 0$ and $0 < \gamma_j < 1$; $h_{max}(t) = t^p$ for $p \in (0, 1)$. Putting this in (5.9), two interesting cases can be noted:

- Consider the case where $\{d_{s_j}\}_{0 \leq j \leq m} = 0$, denote $\max_{1 \leq j \leq m} C_j$ by $\hat{C}_j$. From Theorem 10:

  $$t \leq h_{max}(t) \sum_{l=0}^{h(t)} \max_{1 \leq j \leq m} C_j \delta_j(l)^{-d_{s_j}} = h_{max}(t) \cdot \hat{C}_j (h(t) + 1) \ .$$

  Thus, for $j = 1, \dots, m$:

  $$r_j(t) \leq O(\gamma_j^{\min(t^{1-p},\, t^p)}) \ , \tag{5.15}$$

  i.e., the loss is a stretched-exponential function of the number of iterations t.

- Consider the case where $\exists k \in \{1, \dots, m\}$ such that $\forall l$, $C_k \delta_k(l)^{-d_{s_k}} = \max_{1 \leq j \leq m} C_j \delta_j(l)^{-d_{s_j}}$ and $d_{s_k} > 0$, then from Theorem 10, we have:

  $$\begin{aligned} t \leq h_{max}(t) \sum_{l=0}^{h(t)} C_k \delta_k(l)^{-d_{s_k}} &= C_k \cdot c_k^{-d_{s_k}} \cdot t^p \cdot \frac{\gamma_k^{-d_{s_k}(h(t)+1)} - 1}{\gamma_k^{-d_{s_k}} - 1} \ , \\ \frac{(1 - \gamma_k^{d_{s_k}})}{C_k} \cdot t^{1-p} &\leq c_k^{-d_{s_k}} \gamma_k^{-d_{s_k} h(t)} \ , \\ \left(\frac{C_k}{1 - \gamma^{d_{s_k}}} \right)^{1/d_{s_k}} \cdot t^{-\frac{1-p}{d_{s_k}}} &\geq c_k \gamma_k^{h(t)} \ . \end{aligned}$$

  Hence, $h(t)$ is of a logarithmic order in t, making $h(t) < h_{max}(t) + 1$ as t grows bigger. Thus, with $\delta_j(h) = \Theta(\delta_k(h))$ for $j = 1, \dots, m$;

  $$r_j(t) \leq O(t^{-\frac{1-p}{d_{s_k}}}) \ , \tag{5.16}$$

  i.e., the loss is a polynomially-decreasing function of the number of iterations t.

[3] As the indicator bound (5.11) is dependent on the loss bound (5.9), similar analysis holds true for the indicator bound as well.

One can deduce that the performance (in terms of the loss (2.15)) is influenced by two main factors, viz. the near-optimality dimension of the objectives $\{d_{s_j}\}_{0\leq j\leq m}$, and the maximal depth function $h_{max}(t)$.

The Maximal Depth Function $h_{max}(t)$. From Theorem 10, the maximal depth function $h_{max}(t)$ acts as a multiplicative factor in the definition of $h(t)$ (Eq. 5.8) as well as a limiting factor on the loss bound (Eq. 5.9). This effect of $h_{max}(t)$ elegantly captures the exploration-vs.-exploitation trade-off. Larger $h_{max}(t)$ makes the algorithm more exploitative (deeper tree) and $h(t)$ smaller, while smaller $h_{max}(t)$ makes the algorithm more exploratory (broader tree) and $h(t)$ larger; the inverse proportionality between $h_{max}(t)$ and $h(t)$ evens out the loss bound in both situations.

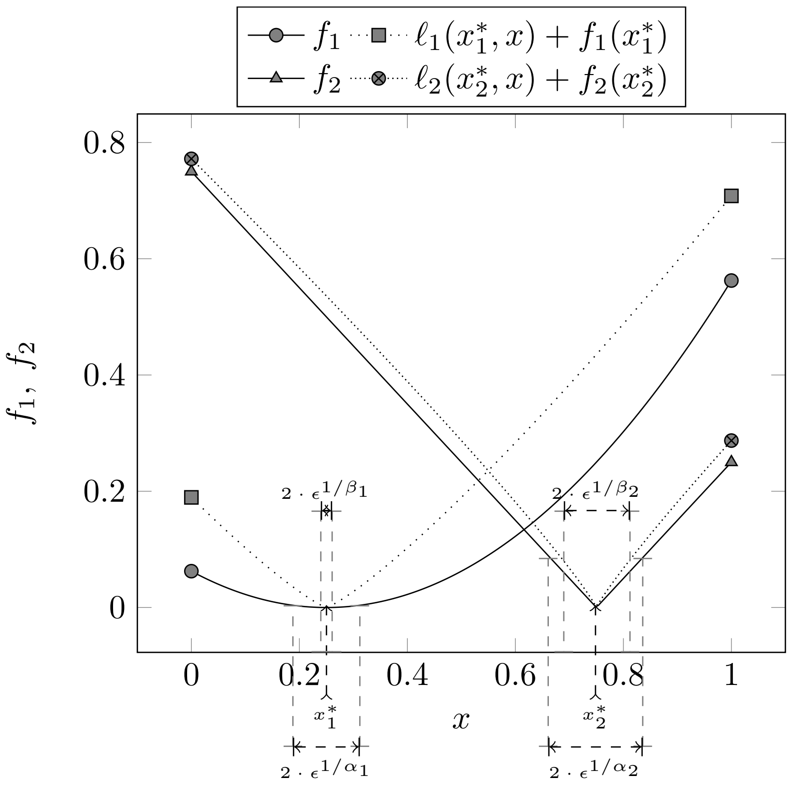


Figure 5.8: Bi-objective problem ($m = 2, n = 1$) over $\mathcal{X} = [0, 1]$ with $f_1(\mathbf{x}) = ||\mathbf{x} - 0.25||_\infty^{\alpha_1}$, $f_2 = ||\mathbf{x} - 0.75||_\infty^{\alpha_2}$, $\ell_1(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_\infty^{\beta_1}$, $\ell_2(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_\infty^{\beta_2}$ where $\beta_1 \leq \alpha_1$, $\beta_2 \leq \alpha_2$. The region $\mathcal{X}_1^\epsilon$ (resp., $\mathcal{X}_2^\epsilon$) is the interval centered around x_1^* (resp., x_*^2) of length $2 \cdot \epsilon^{1/\alpha_1}$ (resp., $2 \cdot \epsilon^{1/\alpha_2}$). They can be packed with $\epsilon^{1/\alpha_1 - 1/\beta_1}$ (resp., $\epsilon^{1/\alpha_2 - 1/\beta_2}$) intervals of length $2 \cdot \epsilon^{1/\beta_1}$ (resp., $2 \cdot \epsilon^{1/\beta_2}$).

The Near-Optimality Dimensions $\{d_{s_j}\}_{0\leq j\leq m}$. While $h_{max}(t)$ is a parameter of the algorithm, $\{d_{s_j}\}_{0\leq j\leq m}$ are dependent on the multi-objective problem at hand

and are related to the algorithm's partitioning strategy through the scaling factors $\{s_j\}_{1\leq j \leq m}$. Consider the near-optimality dimensions for the bi-objective problem (depicted in Figure 5.8 for $n=1$) where $\mathcal{X}=[0,1]^n$, $f_1(\mathbf{x}) = ||\mathbf{x}-0.25||^{\alpha_1}_\infty$, and $f_2 = ||\mathbf{x}-0.75||^{\alpha_2}_\infty$ for $\alpha_1 \geq 1$, $\alpha_2 \geq 1$; and let `MO-SOO` have a partition factor of $K=3^n$. Furthermore, assume the semi-metrics to be $\ell_1(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||^{\beta_1}_\infty$, $\ell_2(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||^{\beta_2}_\infty$ where $\beta_1 \leq \alpha_1$, $\beta_2 \leq \alpha_2$ in line with Assumption A5.1. In the light of Assumption A5.2, $\delta_1(h)$ and $\delta_2(h)$ may be written as $2^{-\beta_1}\cdot 3^{-h\beta_1}$ and $2^{-\beta_2}\cdot 3^{-h\beta_2}$, respectively; and from Assumption A5.3, we have $s_1=1$ and $s_2=1$. The region $\mathcal{X}_1^{\delta_1(h)}$ (resp., $\mathcal{X}_2^{\delta_2(h)}$) is the L_∞-ball of radius $\delta_1(h)^{1/\alpha_1}$ (resp., $\delta_2(h)^{1/\alpha_2}$) centered in $\mathbf{0.25}$ (resp., $\mathbf{0.75}$). In line of Definition 21, these regions can be packed by $\left(\frac{\delta_1(h)^{1/\alpha_1}}{\delta_1(h)^{1/\beta_1}}\right)^n$ (resp., $\left(\frac{\delta_2(h)^{1/\alpha_2}}{\delta_2(h)^{1/\beta_2}}\right)^n$) L_∞-balls of radius $\delta_1(h)^{1/\beta_1}$ (resp., $\delta_2(h)^{1/\beta_2}$). Thus the near-optimality dimensions are $d_{s_1} = n(1/\beta_1 - 1/\alpha_1)$ and $d_{s_2} = n(1/\beta_2 - 1/\alpha_2)$. *Without loss of generality*, three scenarios are present with respect to the first objective:

1. $\alpha_1 = \beta_1 \implies d_{s_1} = 0$; the cardinality of the set $\mathcal{I}_1^h$ is a constant regardless of the depth h and the decision space dimensionality n. This presents a balanced trade-off between exploration and exploitation as the semi-metric ℓ_1 is capturing the function f_1 behavior precisely.

2. $\alpha_1 > \beta_1 \implies d_{s_1} > 0$; the cardinality of the set $\mathcal{I}_1^h$ becomes an increasing function of the depth h and the decision space dimensionality n. This presents a bias towards exploration as the semi-metric ℓ_1 underestimates the behavior of the function f_1.

3. $\alpha_1 < \beta_1$; this violates Assumption A5.1. With this regards, the algorithm becomes more exploitative falling for local optimal solutions as the semi-metric ℓ_1 is overestimating f_1's smoothness.

The first two scenarios coincide with the two cases discussed earlier in this section. As t grows larger and the near-optimality dimensions are zero (reflecting a balance in the exploration-vs.-exploitation dilemma), setting $p=0.5$ in $h_{max}(t) = t^p$ results in a faster decay of the loss bound (Eq. 5.15). On the other hand, when more exploration is needed, setting $p \to 0$ (broader tree) gives a faster loss bound decay (Eq. 5.16).

Remark 8. *It is important to reiterate here that* `MO-SOO` *does not need the knowledge of the functions smoothness and the corresponding near-optimality dimensions, it only requires the* existence *of such smoothness. These measures help only in quantifying the algorithm's performance.*

Remark 9. *The case of zero near-optimality dimension covers a large class of functions. In fact, it has been shown by [160] that the near-optimality dimension is zero for any function defined over a finite-dimensional and bounded space, and whose upper- and lower-envelopes around the global optimizer are of the same order.*

5.3.2.5 Empirical Validation of Theoretical Bounds

In this section, the loss $\mathbf{r}(t)$ and the indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$ bounds of (5.9) and (5.11), respectively, are validated empirically for the bi-objective problem defined in Section 5.3.2.4 and depicted in Figure 5.8. We compute these quantities using *Symbolic Math Toolbox* from The MathWorks, Inc. and compare them with respect to the numerical loss and indicator values obtained by running `MO-SOO` with an evaluation budget of $v = 10^4$ function evaluations.

With a partition factor of $K = 3$, the decreasing sequence $\delta_1(h)$ (resp., $\delta_2(h)$) can be defined as $2^{-\alpha_1} \cdot K^{-3\alpha_1\lfloor h/n \rfloor}$ (resp., $2^{-\alpha_2} \cdot K^{-3\alpha_2\lfloor h/n \rfloor}$) as the search space is partitioned coordinate-wise per depth. Moreover, from Assumption A5.3, we have $s_1 = 1$ (resp., $s_2 = 1$). C_1 and C_2 of Definition 21 are set to 2 as the cell centers may lie on the boundary of $\mathcal{X}_1^{\delta_1(h)}$ and $\mathcal{X}_2^{\delta_2(h)}$, respectively.

To assess the effect of the conflict dimension Ψ (defined in Definition 22), eight instances of the problem are tested, where $n \in \{1, 2\}$, and the j-optimal solutions (x_1^*, x_2^*) are set in one of four configurations—reflecting among others the maximum and minimum Ψ values. The Pareto front $\mathcal{Y}^*$ and the conflict dimension Ψ of the problem are estimated numerically from 10^6 uniformly-sampled points. While the maximal depth function $h_{max}(t)$ acts as a very conservative multiplicative factor in (5.8) for the number of depths visited in each iteration. In our experiments, we have recorded the number of depths visited in each iteration and used the recorded values as the multiplicative factor in computing the theoretical bounds of (5.9) and (5.11).

The numerical and theoretical measures are presented in Figure 5.9. First, one can easily verify Lemma 3. Second, whilst having the same evaluation budget v, the conflict and decision space dimensions have a clear impact on the corresponding number of iterations t. Recall that one iteration represents executing the lines 4–8 of Algorithm 9, once. Though with some offset, one can note how the theoretical measures upper bound the numerical measures with a similar trend. The code for generating the data presented in this section is available at `https://www.dropbox.com/s/ssiq1m52hczuj7a/mosoo-theory-validation.rar?dl=0`.

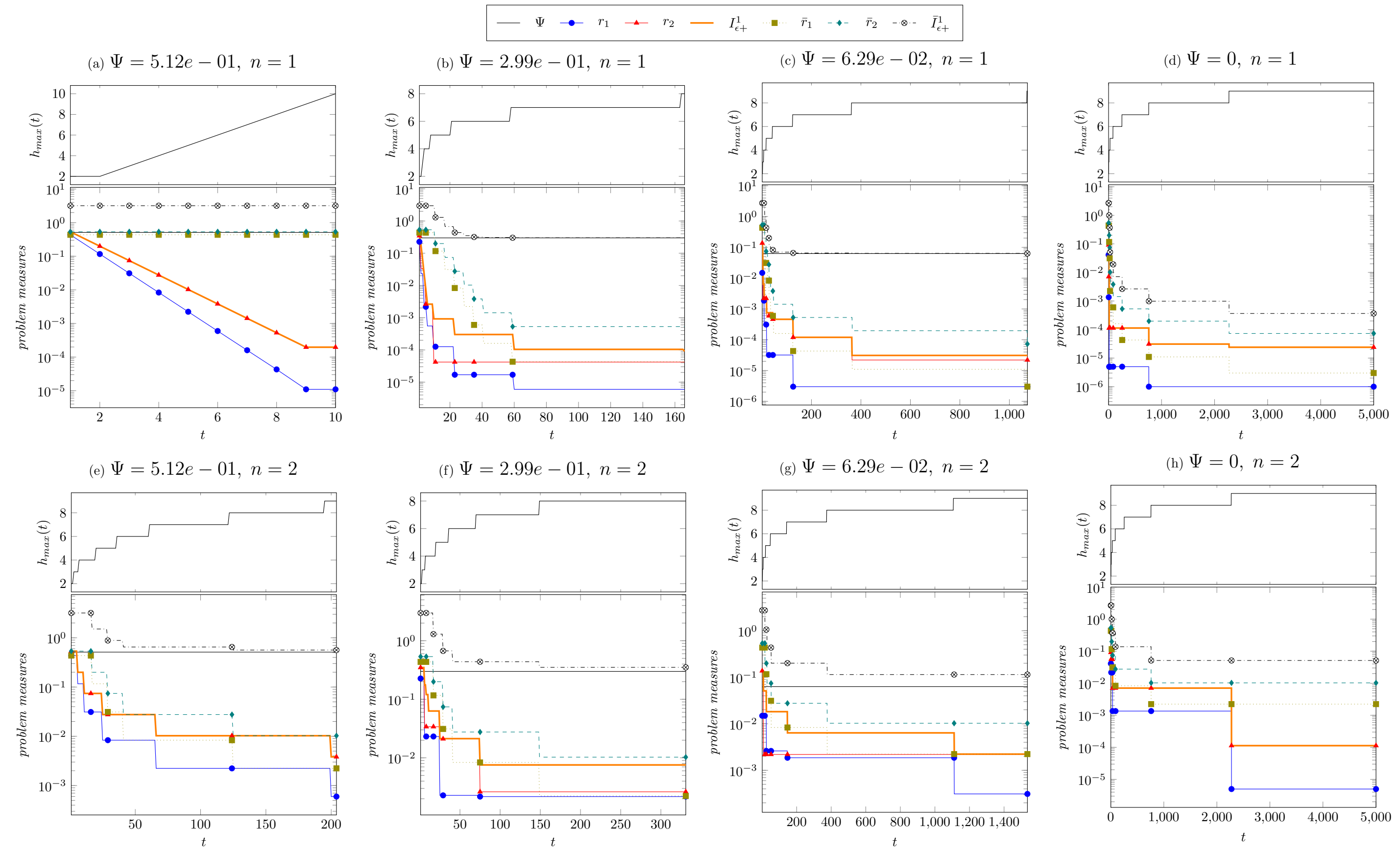


Figure 5.9: Empirical validation of `MO-SOO`'s finite-time analysis for eight instances $\{a, \ldots, h\}$ of the bi-objective problem of Figure 5.8. Each plot shows the problem measures, namely the loss measures $r_1(t)$, $r_2(t)$ and the indicator $I^1_{\epsilon+}(\mathcal{Y}^t_*)$, as well as their upper bounds (denoted by $\bar{r}_1$, $\bar{r}_2$, and $\bar{I}^1_{\epsilon+}$, respectively) as a function of the number of iterations t with a computational budget of $v = 10^4$ function evaluations. The upper bounds are obtained via symbolic computation of the (5.9) and (5.11) equations using MATLAB's Symbolic Math Toolbox. The header of each instance's plot reports the decision space dimension n and the conflict dimension Ψ. The j-optimal solutions $(x^*_1,\ x^*_2)$ are fixed as follows: $(\mathbf{0}, \mathbf{1})$ for (a) and (e), $(\mathbf{0.21}, \mathbf{0.81})$ for (b) and (f), $(\mathbf{0.47}, \mathbf{0.61})$ for (c) and (g), and $(\mathbf{0.57}, \mathbf{0.57})$ for (d) and (h).

5.4 Conclusion

This chapter proposed two multi-objective algorithmic instances within the MSO framework, namely `MO-DIRECT` and `MO-SOO` inspired by the single-objective `DIRECT` [56] and `SOO` [34] algorithms. With the aim of recognizing Pareto optimal solutions, `MO-DIRECT` creates iteratively a set of shrinking hyperrectangles over the search space, seeking an exploration-vs.-exploitation trade-off. Likewise, `MO-SOO` encodes the feasible decision space in a tree of bandits and expands it using the non-dominated Pareto relation among its tree nodes.

While both algorithms are optimal in the limit—that is, they enjoy the consistency property, `MO-DIRECT`'s finite-time analysis is complicated by the inclusion of the hyperrectangle size as a criterion in its search for Pareto-optimal solutions. On the other hand, `MO-SOO` performance in terms of finite-time rate has been established, based on three basic assumptions about the function smoothness and hierarchical partitioning. The theoretical analysis of `MO-SOO` establishes a deterministic upper bound on the Pareto-compliant ϵ-indicator for continuous MOPs, whereas existing theoretical analysis of multi-objective solvers either considers finite-set/discrete problems or provides probabilistic guarantees.

Chapter 6

Benchmarking Multi-Objective Multi-Scale Search

"Do not wait for someone else to validate your existence; it is your own responsibility"

- Jasz Gill

In general, optimization algorithms are assessed in one of two ways, viz. theoretical and empirical analysis. In theoretical analysis (as carried out in Chapter 5), a principled methodology is carried out to derive an analytical bound of the (run-time) solution quality or guarantee its optimality in the limit. Alternatively, empirical analysis employs experimental simulations of the algorithm on complex problems, gaining an insight on the algorithm's practicality/applicability on real-world problems. With this regard, the present chapter complements the theoretical perspective of the proposed multi-objective algorithms, `MO-DIRECT` and `MO-SOO`, presented in Chapter 5 by validating them on a set of multi-objective problems.

In the course of developing efficient good algorithms, the multi-objective optimization community have been testing their techniques on arbitrary sets of problems from the literature (e.g., [64]). In this chapter, we seek to consolidate a testbed for black-box multi-objective optimization in line with the recent efforts in that direction (e.g., Black-Box Optimization Benchmarking (BBOB) [218] and Black-box Optimization Competition (BBComp) [57])

Therefore, our contribution here is of two-fold: i). to validate the empirical performance of `MO-DIRECT` and `MO-SOO` and compare them with the state-of-the-art stochastic as well as deterministic multi-objective algorithms; ii). to build a benchmarking platform around a collection of multi-objective problems reflecting common challenges in real-world problems. To this end, the proposed solvers are evaluated on the recently presented 300 bi-objective problems from the BBOB testbed by [6], which we refer to as BO-BBOB. Furthermore, the Black-box Multi-objective Optimization Benchmarking (BMOBench) is proposed to amalgamate 100 MOPs from the literature.

The rest of this chapter is organized as follows. In Section 6.1, `MO-DIRECT` and `MO-SOO` are validated on the BO-BBOB testbed, whereas Section 6.2 introduces the proposed BMOBench platform and reports the performance of `MO-DIRECT` and `MO-SOO` on the same.

6.1 Bi-Objective Black-Box Optimization Benchmarking (BO-BBOB)

This section presents a numerical assessment and comparison of `MO-DIRECT` and `MO-SOO` with respect to the state-of-the-art multi-objective algorithms based on the BBOB testbed [218]. The rest of the section discusses the compared algorithms, test problems, setup, evaluation procedure, and performance comparison.

6.1.1 Compared Algorithms

Four state-of-the-art algorithms from main multi-objective approaches have been selected to compare against `MO-DIRECT` and `MO-SOO`: the genetic algorithm, `NSGA-II` [177]; the decomposition-based algorithm, `MOEA/D`, by Zhang and Li [184]; the adaptive evolutionary strategy, `MO-CMA-ES`, by Voß et al. [219]; and the hypervolume-based algorithm, `SMS-EMOA`, by Beume et al. [220]. In our experiments, we have used `C/C++` implementations of the algorithms, which can be retrieved from the sources shown in Table 6.1.

6.1.2 Test Problems

We make use of the recently proposed benchmark platform by Brockhoff et al. [6],[1] which we refer to as *BO-BBOB*. It comes with three-hundred bi-objective optimization problems, with simple bound constraints, that is to say, $\mathcal{X} = [\mathbf{l}, \mathbf{u}] \subset$

[1] The test suite is retrieved from `http://coco.gforge.inria.fr/doku.php?id=mo-gecco2015`.

Table 6.1: The compared algorithms and their implementation sources.

Algorithm	Source
NSGA-II [177]	C implementation at http://www.iitk.ac.in/kangal/codes.shtml
MOEA/D [184]	C++ implementation at http://dces.essex.ac.uk/staff/zhang/webofmoead.htm
MO-CMA-ES [219]	Shark library at http://image.diku.dk/shark
SMS-EMOA [220]	Shark library at http://image.diku.dk/shark
MO-SOO	Our C implementation at http://ash-aldujaili.github.io/MOSOO
MO-DIRECT	Our C implementation at https://sites.google.com/site/modirectmops/

$\mathbb{R}^n$, where $\mathbf{u} \succeq \mathbf{l}$. These problems are derived using combinations of 24 well-known single-objective BBOB functions [218], spanning a wide range of attributes that are observed in real-world problems (e.g., separability, conditioning, and modality). Moreover, investigating the scalability of algorithms with increasing n is possible with BO-BBOB problems as they can be tested over different search dimensions. Furthermore, each of these problems has *five* different *instances*, transforming the original functions through such operations as search space rotation and non-linear variable transformations.

6.1.2.1 Experimental Setup

In this section, we discuss the experimental setup, viz. algorithm parameters and evaluation budget settings.

6.1.2.2 Parameters Setting

To improve the exploitation search component as opposed to the exploratory inherent nature of the MO-DIRECT algorithm, the hyperrectangle threshold σ_t—the sole parameter of MO-DIRECT—was set to be the square of the problem dimensionality n^2. The parameters setting for MO-SOO is listed in Table 6.2. For the rest of the algorithms, the default settings of their implementations are used, except for those listed in Table 6.3, which were set as in [6].

6.1.2.3 Evaluation Budget

MO-SOO and MO-DIRECT are deterministic algorithms producing the same approximation set in each run of the algorithm for a given problem, whereas the approx-

Table 6.2: Parameters setting of MO-SOO. In accordance with many single-objective MSO algorithms, we set the partition factor K to be 3 and the cells partitioning to be along one dimension per depth in a sequential fashion. Given a finite evaluation budget $v(t)$ at iteration t, we have simulated the effect of exploration-vs.-exploitation trade-off by offsetting the maximal depth $h_{max}(t)$ from $h_o(t)$, the depth of the shallowest leaf node (i.e., the minimal depth at iteration t), by the depth of the uniform tree that would have been built using double the evaluation budget. Furthermore, as the algorithm splits one dimension per depth, we considered an additive factor proportional to the decision space n. We found a factor of $n^{1.5}$ generalize well over the range of dimensions considered $n \in \{2, 3, 5, 10, 20\}$.

Parameter	**Value**
K	3
Splitting Dimension at depth h	$h \bmod n$
$h_{max}(t)$	$h_o(t) + \log_K(2 \cdot v(t)) + n^{1.5}$

Table 6.3: The modified parameters of MOEA/D, MO-CMA-ES, and SMS-EMOA according to Brockhoff et al. [6].

Algorithm	**Parameter**	**Value**
All	Population Size	200
NSGA-II	Binary Crossover Distribution Index	20
	Binary Crossover rate	1
	Polynomial Mutation Index Parameter	50
	Polynomial Mutation Rate	$1/n$
MOEA/D	Crossover Rate	1
	Mutation Rate	$1/n$
	SBX Crossover Index Parameter	20
	Polynomial Mutation Index Parameter	50
	Differential Evolution Crossover CR	0.8
	Differential Evolution Crossover F	0.9
	Neighborhood Size	20
MO-CMA-ES	Notion of Success	population-based
	Evolutionary Strategy	generational

imation sets produced by the compared stochastic algorithms: NSGA-II, MOEA/D, MO-CMA-ES, and SMS-EMOA can be different every time they are run for a given problem. In practice, stochastic algorithms are run several times per problem. To this end and to ensure a fair comparison, given a computational budget of v function evaluations per run, the stochastic algorithms are allocated 10 runs per problem instance. On the other hand, MO-SOO is run once per problem instance with the cumulated $10 \times v$ function evaluations.

In our experiments, the evaluation budget v is made proportional to the search space dimension n and is set to $10^3 \cdot n$. The overall computational budget used by an algorithm on BO-BBOB is the product of the evaluation budget per run, the number of instances per problem, the number of problems, and the number of runs per instances.

With $n = 2$, for instance, the overall computational budget used by `MO-SOO` on BO-BBOB is $10^4 \cdot 2 \cdot 5 \cdot 300 \cdot 1 = 3 \times 10^7$ function evaluations. Each of the other algorithms uses also a computational budget of $10^3 \cdot 2 \cdot 5 \cdot 300 \cdot 10 = 3 \times 10^7$ function evaluations.

Remark 10. *Comparing deterministic and stochastic algorithms is a general issue and ongoing research that goes beyond the thesis. To this end, the algorithms performance are also compared in a different experimental setup at https://www.dropbox.com/s/uxe4gzcba3lb3al/mosoo-supplement-quantiles-thesis.pdf?dl=0 comparing performance of the deterministic algorithms to quantiles of the stochastic algorithms' under the same evaluation budgets.*

6.1.3 Benchmark Procedure

Similar to [6], a set of targets are defined for each problem in terms of the most popular and recommended quality indicators [169], viz. the hypervolume (I_H^-) and the unary additive ϵ-indicator ($I_{\epsilon+}^1$). A solver (algorithm) is then evaluated based on its runtime with respect to each target: the number of function evaluations used until the target is reached. We present the recorded runtime values in terms of *data profiles* [221]. A data profile can be regarded as an empirical cumulative distribution function of the observed number of function evaluations in which the y-axis tells how many targets—over the set of problems and quality indicators—have been reached by each algorithm for a given evaluation budget (on the x-axis). Mathematically, a data profile for a solver s on a problem class P has the form

$$d_s(\alpha) = \frac{1}{|P|}\left|\left\{p \in P \;\middle|\; \frac{t_{p,s}}{n_p} \leq \alpha\right\}\right| ,$$

where $t_{p,s}$ is the observed runtime of solver s on solving problem p (hitting a target) over a decision space $\mathcal{X} \subseteq \mathbb{R}^{n_p}$. The data profile approach captures several benchmarking aspects, namely the convergence behavior over time rather than a fixed budget, which can as well be aggregated over problems of similar category [see, for more details, 6]. In our experiments, 70 linearly spaced values in the logarithmic scale from $10^{-0.1}$ to 10^{-3} and from $10^{-0.1}$ to 10^{-2} were used as targets for I_H^- and $I_{\epsilon+}^1$, respectively.

The I_H^- and $I_{\epsilon+}^1$values are computed for each algorithm at any point of its run based on the set of all (normalized) non-dominated vectors found so far—i.e., the *archive*—with respect to a (normalized) *reference* set $R \in \Omega$.[2] For I_H^-, we made use of the available hypervolume values of the reference set computed by Brockhoff et al. [6]. Nevertheless, we had to compute the reference set for calculating $I_{\epsilon+}^1$. The reference set R for $I_{\epsilon+}^1$ was computed by running `NSGA-II`, `MOEA/D`, and `MO-CMA-ES` for 10 runs with $10^4 \cdot n$ function evaluations.

As mentioned in the previous section, we aim to provide a fair comparison between `MO-SOO` and the state-of-the-art solvers and accommodate the multiple-run practice for stochastic algorithms, at the same time. This has been reflected in the evaluation budget allocation (see Section 6.1.2.3). Likewise, we need to adapt the data profiles. To this end, given a problem instance and for each one of the stochastic solvers, we consider the *best* reported runtime for each target from the solver's 10 runs, rather than the mean value. With this setting at hand, the data profile of `MO-SOO` at 10^3 function evaluations, for instance, can be compared to that of `SMS-EMOA` at 10^2 function evaluations.

6.1.4 Performance Discussion

Results from the performance experiments are presented—in terms of data profiles aggregated over all the problems for each of the tested search space dimension $n \in \{2, 3, 5, 10, 20\}$ —in Figures 6.1 and 6.2 for HV and $I_{\epsilon+}$, respectively. On the other hand, Figures 6.3 and 6.4 report the aggregated data profiles over problem categories for the same. The online supplement presents the results in detail, reporting data profiles per problem (aggregated over the problem's five instances) and per problem category for each n, for both indicators.[3] As mentioned earlier, the data profiles show the proportion of problems solved (targets hit) as a function of

[2]In line with [169], $I_H^-(\mathcal{Y}_*^v) = I_H(R) - I_H(\mathcal{Y}_*^v)$ and $I_{\epsilon+}^1(\mathcal{Y}_*^v) = I_{\epsilon+}(\mathcal{Y}_*^v, R)$, where the $I_H(A)$ indicator for an approximation set A measures the hypervolume of the objective space portion that is *weakly* dominated by A [222].

[3]The online supplement will be made available at the project's website. Currently, it is available at `https://www.dropbox.com/s/8snmnuozak19scr/mosoo-supplement-thesis.pdf?dl=0`.

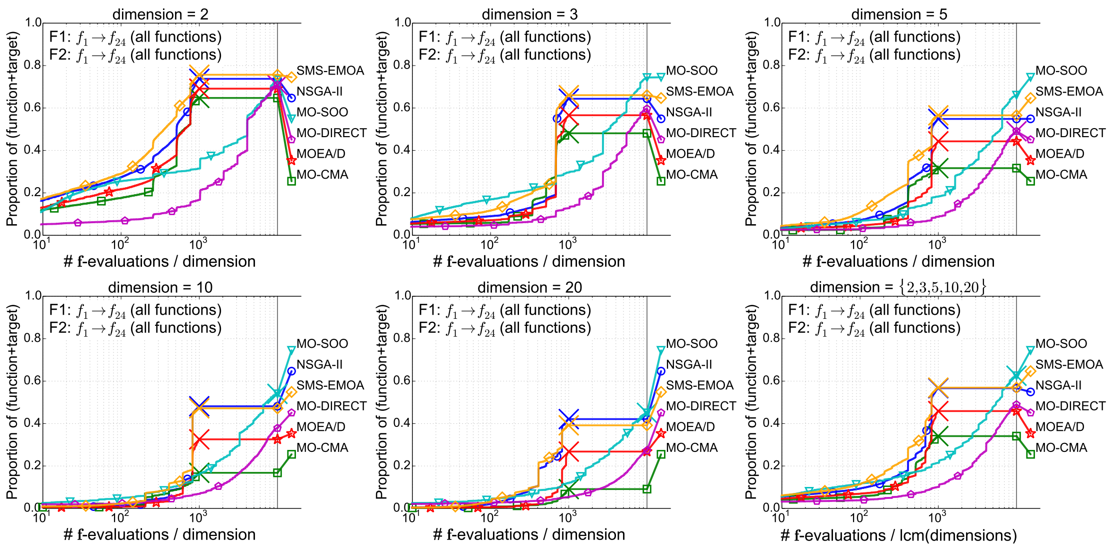


Figure 6.1: I_H^--based data profiles aggregating over all 300 problems for a decision space dimensionality $n \in \{2, 3, 5, 10, 20\}$. The symbol $\times$ indicates the maximum number of function evaluations. The bottom right plot shows the aggregated data profiles over all the decision space dimensions tested.

the number of function evaluations used (denoted by #f-evaluations in the figures). Overall, results show a comparable performance of `MO-SOO` with respect to the compared algorithms. Sometimes (especially, when #f-evaluations $\leq 5 \cdot 10^2 \cdot n$), `MO-SOO` outperforms the compared algorithms (especially, `MO-CMA-ES` and `MOEA/D`) even with a computational budget ten times *less* than that used by the rest of the algorithms. Recall that the data profile of `MO-SOO` at v #f-evaluations represents the performance of the algorithm after v #f-evaluations, while those of the compared algorithms represent their *best* performance among 10 runs after v #f-evaluations each. One can observe such behavior clearly in problems with separable functions (see, e.g., the first two rows of Figures 6.3 and 6.4). On the other hand, `MO-DIRECT`'s exploratory behavior is evident by its data profiles that catch up slowly after the rest of the algorithms.

Problem Dimensionality. As shown in Figures 6.1 and 6.2, the targets become more difficult to solve within the allocated evaluation budge as n increases: `MO-SOO` solves roughly 50% of the problems in 20-n, compared to around 80% in 2-n. The effect of dimensionality is more present on `MO-DIRECT`, which hits around 30% of the targets in 20-n. On the other hand, `MO-CMA-ES`'s performance degrades from hitting 70% of the targets in 2-n to 10% in 20-n. However, given `MO-CMA-ES`'s steep performance towards the end of allocated budget, the algorithm is projected

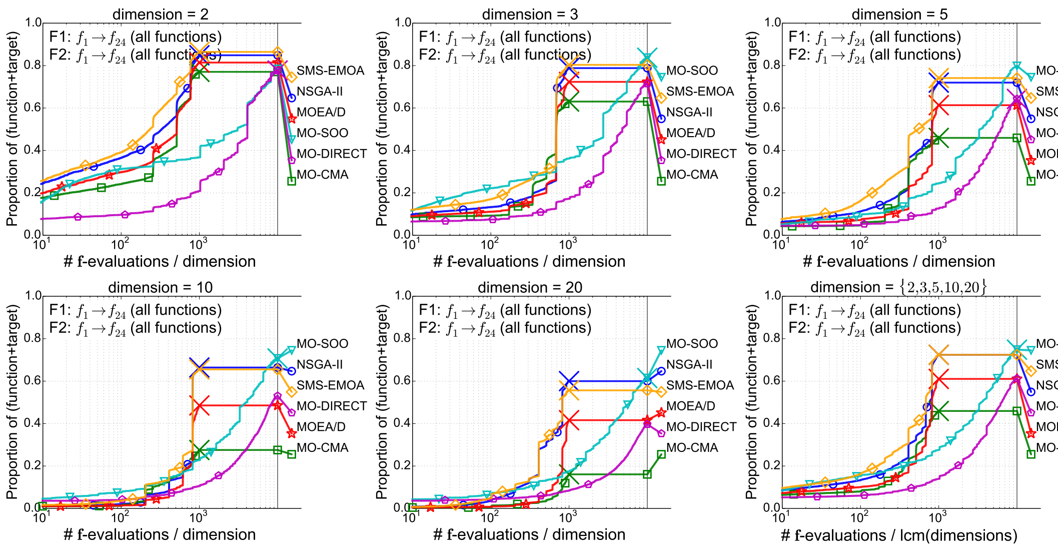


Figure 6.2: $I^1_{\epsilon+}$-based data profiles aggregating over all 300 problems for a decision space dimensionality $n \in \{2, 3, 5, 10, 20\}$. The symbol $\times$ indicates the maximum number of function evaluations. The bottom right plot shows the aggregated data profiles over all the decision space dimensions tested, where the term *lcm* denotes the least common multiple operator.

to solve more targets (up to 70%) with 10% increase in the evaluation budget. In general, the performance gap among the algorithms grows with n as more function evaluations used.

Problem Category. Among the 15 categories of BO-BBOB test suite problems, `MO-SOO`'s performance suffers the most on problems with *weakly-structured multi-modal structures*: hitting around 40% of the targets, which is the lowest among the categories, as shown in Figure 6.3. This is supported by the theoretical basis of the algorithm, since the C_j constant of the near-optimality dimension (Definition 21) is directly proportional to the number of distinct global optima—as shown by Munos [223, page 85]—which is a special case of multi-modal functions. In other words, for a problem whose objectives have k distinct global optima, `MO-SOO` requires an evaluation budget that can be as much as k times the budget used for a problem with uni-modal objectives of similar structure. Similar behavior can be noted on `MO-DIRECT`'s performance but with a slower progress. This slow convergence can be distinctively seen from the algorithm's data profile for separable problems (the first row of Figures 6.3 and 6.4), this is supported by the fact that it is more efficient to perform *independent* n coordinate-wise searches for separable problem, which is not the case for `MO-DIRECT`'s partitioning procedure.

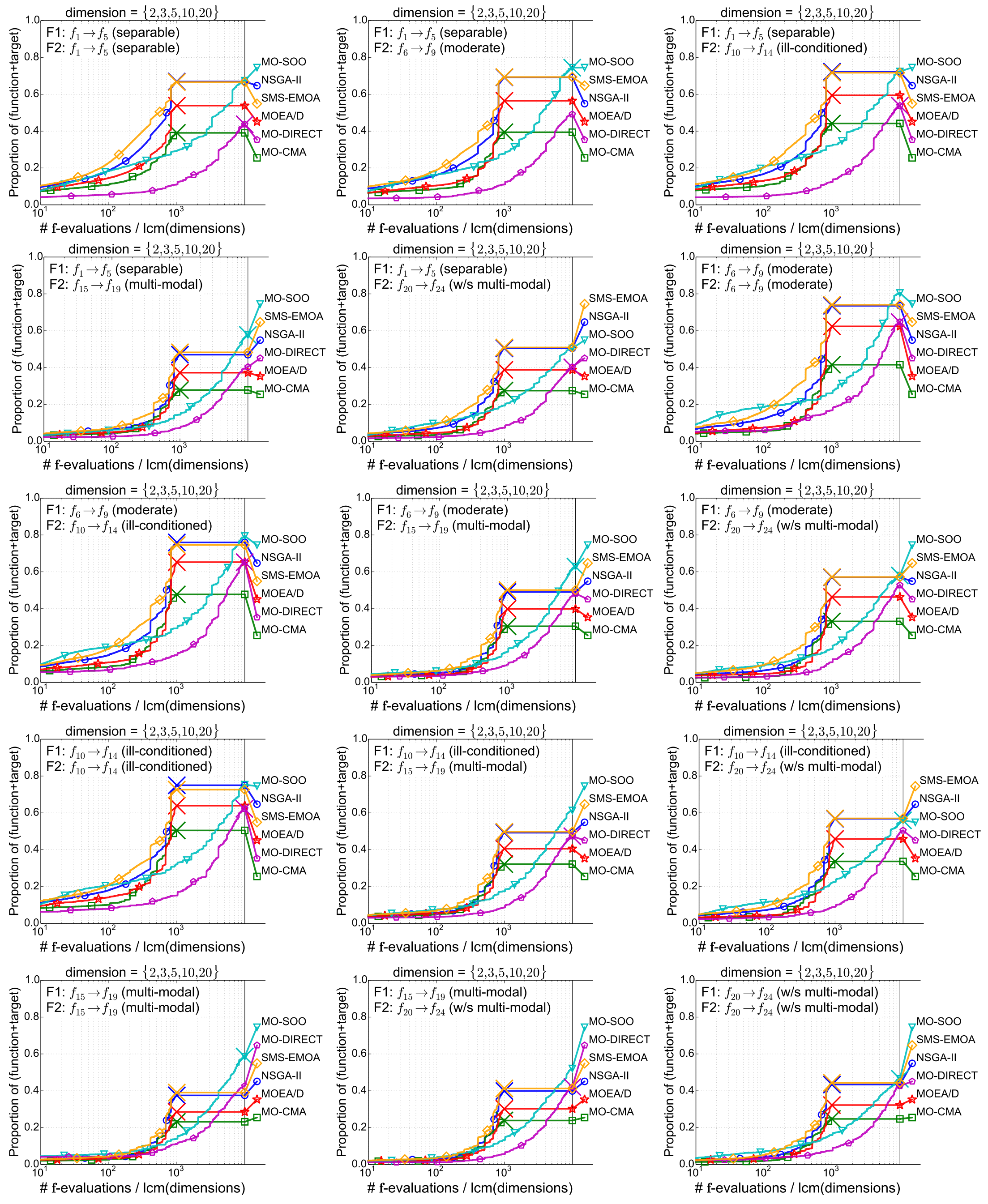


Figure 6.3: I_H^--based data profiles aggregated over problem categories across all the dimensions tested. The symbol $\times$ indicates the maximum number of function evaluations, whereas the term *lcm* denotes the least common multiple operator.

Limitations. When compared with `MO-SOO`, `MO-DIRECT`'s sampling and partitioning along all the coordinates appears to be an ineffective strategy especially in higher dimensions. On the other hand, the sequential partitioning scheme of `MO-SOO` gives equal importance to the coordinates of the search space, which may

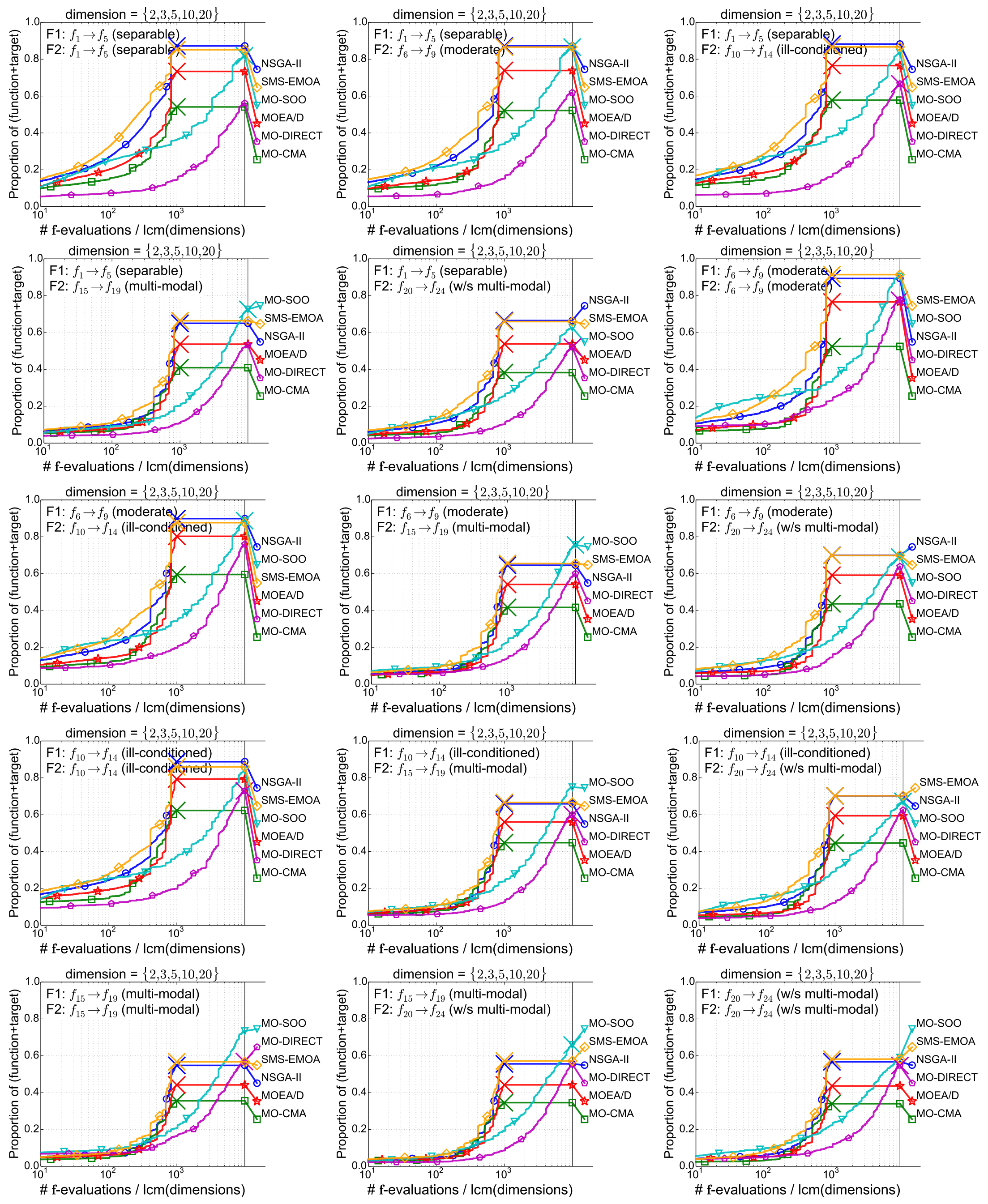


Figure 6.4: $I^1_{\epsilon+}$-indicator-based data profiles aggregated over problem categories across all the dimensions tested. The symbol $\times$ indicates the maximum number of function evaluations, whereas the term *lcm* denotes the least common multiple operator.

affect the performance on weakly-structured objectives. Another factor influencing both algorithms' performance is the nature of the $\texttt{ND}(\cdot)$ operator: selecting all the nodes on the non-dominated front without considering their spread, which may lead to inefficient use of the evaluation budget.

6.2 Black-box Multi-objective Optimization Benchmarking (BMOBench)

Inspired by the BO-BBOB platform [6] and the work of Custódio et al. [65], this section describes the Black-box Multi-objective Optimization Benchmarking (BMOBench) platform, available at https://github.com/ash-aldujaili/BMOBench. It presents the test problems, evaluation procedure, and experimental setup. Towards the end of this section, the BMOBench is demonstrated by comparing recent multi-objective solvers from the literature, namely `SMS-EMOA` [220], `DMS` [65], and `MO-HOO` [133], with the proposed multi-objective MSO algorithms. BMOBench was made possible by making use of the codes developed by Brockhoff et al. [6] and Custódio et al. [65].

6.2.1 Test Problems

One-hundred multi-objective optimization problems from the literature are selected.[4] These problems have simple bound constraints, that is to say, $\mathcal{X} = [\mathbf{l}, \mathbf{u}] \subset \mathbb{R}^n$, where $\mathbf{u} \succeq \mathbf{l}$. Table 6.4 presents a brief list of these problems with number of dimensions/objectives. In order to have a better understanding of the algorithm strength/weakness, the benchmark problems are categorized (wherever possible) according to three key characteristics, namely *dimensionality*: low- or high-dimension decision space, *separability*: separable or non-separable objectives, and *modality*: uni-modal or multi-modal objectives. Each of these attributes imposes a different challenge in solving an MOP [224].

6.2.2 Evaluation Budget

`MO-SOO` and `MO-DIRECT` are deterministic algorithms producing the same approximation set in each run of the algorithm for a given problem, whereas the approximation sets produced by the compared stochastic algorithms: `DMS` and `SMS-EMOA` can be different every time they are run for a given problem. In practice, stochastic algorithms are run several times per problem. To this end and to ensure a fair

[4]retrieved from http://www.mat.uc.pt/dms.

Table 6.4: Test problems definition and properties. Symbols: **D** : dimensionality $\in$ {L : low-dimensionality, H : high-dimensionality}; **S** : separability $\in$ {S : separable, NS : non-separable}; **M** : modality $\in$ {U : uni-modal, M : multi-modal}; $\times$: uncategorized/mixed.

#	Problem Name	n	m	**D**	**S**	**M**	#	Problem Name	n	m	**D**	**S**	**M**
1	BK1 [224]	2	2	L	S	U	51	L3ZDT3 [225]	30	2	H	$\times$	$\times$
2	CL1 [226]	4	2	L	$\times$	$\times$	52	L3ZDT4 [225]	30	2	H	$\times$	$\times$
3	Deb41 [64]	2	2	L	$\times$	$\times$	53	L3ZDT6 [225]	10	2	H	$\times$	$\times$
4	Deb512a [64]	2	2	L	$\times$	$\times$	54	LE1 [224]	2	2	L	S	U
5	Deb512b [64]	2	2	L	$\times$	$\times$	55	lovison1 [227]	2	2	L	$\times$	$\times$
6	Deb512c [64]	2	2	L	$\times$	$\times$	56	lovison2 [227]	2	2	L	$\times$	$\times$
7	Deb513 [64]	2	2	L	$\times$	$\times$	57	lovison3 [227]	2	2	L	$\times$	$\times$
8	Deb521a [64]	2	2	L	$\times$	$\times$	58	lovison4 [227]	2	2	L	$\times$	$\times$
9	Deb521b [64]	2	2	L	$\times$	$\times$	59	lovison5 [227]	3	3	L	$\times$	$\times$
10	Deb53 [64]	2	2	L	$\times$	$\times$	60	lovison6 [227]	3	3	L	$\times$	$\times$
11	DG01 [224]	1	2	L	$\times$	M	61	LRS1 [224]	2	2	L	S	U
12	DPAM1 [224]	10	2	H	NS	$\times$	62	MHHM1 [224]	1	3	L	$\times$	U
13	DTLZ1 [228]	7	3	H	$\times$	M	63	MHHM2 [224]	2	3	L	S	U
14	DTLZ1n2 [228]	2	2	L	$\times$	M	64	MLF1 [224]	1	2	L	$\times$	M
15	DTLZ2 [228]	12	3	H	$\times$	U	65	MLF2 [224]	2	2	L	NS	M
16	DTLZ2n2 [228]	2	2	L	$\times$	U	66	MOP1 [224]	1	2	L	S	U
17	DTLZ3 [228]	12	3	H	$\times$	M	67	MOP2 [224]	4	2	L	S	U
18	DTLZ3n2 [228]	2	2	L	$\times$	M	68	MOP3 [224]	2	2	L	$\times$	$\times$
19	DTLZ4 [228]	12	3	H	$\times$	U	69	MOP4 [224]	3	2	L	S	$\times$
20	DTLZ4n2 [228]	2	2	L	$\times$	U	70	MOP5 [224]	2	3	L	NS	$\times$
21	DTLZ5 [228]	12	3	H	$\times$	U	71	MOP6 [224]	2	2	L	S	$\times$
22	DTLZ5n2 [228]	2	2	L	$\times$	U	72	MOP7 [224]	2	3	L	$\times$	U
23	DTLZ6 [228]	22	3	H	$\times$	U	73	OKA1 [229]	2	2	L	$\times$	$\times$
24	DTLZ6n2 [228]	2	2	L	$\times$	U	74	OKA2 [229]	3	2	L	$\times$	$\times$
25	ex005 [174]	2	2	L	$\times$	U	75	QV1 [224]	10	2	H	S	M
26	Far1 [224]	2	2	L	NS	M	76	Sch1 [224]	1	2	L	$\times$	$\times$
27	FES1 [224]	10	2	H	S	U	77	SK1 [224]	1	2	L	S	M
28	FES2 [224]	10	3	H	S	U	78	SK2 [224]	4	2	L	$\times$	$\times$
29	FES3 [224]	10	4	H	S	U	79	SP1 [224]	2	2	L	NS	U
30	Fonseca [230]	2	2	L	S	U	80	SSFYY1 [224]	2	2	L	S	U
31	I1 [231]	8	3	H	S	U	81	SSFYY2 [224]	1	2	L	$\times$	$\times$
32	I2 [231]	8	3	H	NS	U	82	TKLY1 [224]	4	2	L	$\times$	$\times$
33	I3 [231]	8	3	H	NS	U	83	VFM1 [224]	2	3	L	S	U
34	I4 [231]	8	3	H	NS	U	84	VU1 [224]	2	2	L	S	U
35	I5 [231]	8	3	H	NS	U	85	VU2 [224]	2	2	L	S	U
36	IKK1 [224]	2	3	L	$\times$	U	86	WFG1 [224]	8	3	H	S	U
37	IM1 [224]	2	2	L	$\times$	U	87	WFG2 [224]	8	3	H	NS	$\times$
38	Jin1 [232]	2	2	L	$\times$	U	88	WFG3 [224]	8	3	H	NS	U
39	Jin2 [232]	2	2	L	$\times$	U	89	WFG4 [224]	8	3	H	S	M
40	Jin3 [232]	2	2	L	$\times$	U	90	WFG5 [224]	8	3	H	S	$\times$
41	Jin4 [232]	2	2	L	$\times$	U	91	WFG6 [224]	8	3	H	NS	U
42	Kursawe [233]	3	2	L	$\times$	$\times$	92	WFG7 [224]	8	3	H	S	U
43	L1ZDT4 [225]	10	2	H	$\times$	$\times$	93	WFG8 [224]	8	3	H	NS	U
44	L2ZDT1 [225]	30	2	H	$\times$	$\times$	94	WFG9 [224]	8	3	H	NS	$\times$
45	L2ZDT2 [225]	30	2	H	$\times$	$\times$	95	ZDT1 [234]	30	2	H	S	U
46	L2ZDT3 [225]	30	2	H	$\times$	$\times$	96	ZDT2 [234]	30	2	H	S	U
47	L2ZDT4 [225]	30	2	H	$\times$	$\times$	97	ZDT3 [234]	30	2	H	S	$\times$
48	L2ZDT6 [225]	10	2	H	$\times$	$\times$	98	ZDT4 [234]	10	2	H	S	$\times$
49	L3ZDT1 [225]	30	2	H	$\times$	$\times$	99	ZDT6 [234]	10	2	H	S	M
50	L3ZDT2 [225]	30	2	H	$\times$	$\times$	100	ZLT1 [224]	10	3	H	S	U

comparison, given a computational budget of v function evaluations per run, the stochastic algorithms are allocated 10 runs per problem instance. On the other hand, the deterministic algorithms are run once per problem instance with the cumulated $10 \times v$ function evaluations.

In our experiments, the evaluation budget v is made proportional to the search space dimension n and is set to $10^2 \cdot n$. The overall computational budget used by an algorithm on BMOBench is the product of the evaluation budget per run, the number of problems, and the number of runs per problem.

With $n = 2$, for instance, the overall computational budget used by `MO-SOO` and `MO-DIRECT` on BMOBench is $10^3 \cdot 2 \cdot 100 \cdot 1 = 2 \times 10^5$ function evaluations. Each of the other algorithms uses also a computational budget of $10^2 \cdot 2 \cdot 100 \cdot 10 = 2 \times 10^5$ function evaluations.

6.2.3 Benchmark Procedure

Quality Indicator (I)	Pareto-Compliant	Reference Set Required	Target
Hypervolume Difference (I_H^-)	Yes	Yes	Minimize
Generational Distance (I_{GD})	No	Yes	Minimize
Inverted Generational Distance (I_{IGD})	No	Yes	Minimize
Additive ϵ-Indicator ($I^1_{\epsilon+}$)	Yes	Yes	Minimize

Table 6.5: Employed Quality Indicators. Adapted from Hadka [2] (for more details, see [3, 4]).

In line with the BO-BBOB testbed, data profiles are used. Nevertheless, four—rather than two—popular quality indicators [168, 169] are employed as listed in Table 6.5. In the present experiments, 70 linearly spaced values in the logarithmic scale from $10^{-0.8}$ to 10^{-3} and from $10^{-0.1}$ to 10^{-2} were used as targets for I_H^-, I_{GD}, and I_{IGD}; and $I^1_{\epsilon+}$, respectively.

The I_H^-, I_{GD}, I_{IGD} and $I^1_{\epsilon+}$values are computed for each algorithm at any point of its run based on the set of all (normalized) non-dominated vectors found so far—i.e., the *archive*—with respect to a (normalized) *reference* set $R \in \Omega$. We computed the reference set for calculating the quality indicators by post-processing the union of approximation sets generated by the evolutionary algorithms used in [65].

6.2.4 Compared Algorithms

As the BMOBench is coded in both MATLAB and C, access to a larger pool of multi-objective algorithms was possible. For the sake of demonstration, several algorithms were selected as follows. Besides `SMS-EMOA` [220], the Direct Multi-Search [65] algorithm from the MP community is considered in its deterministic (`DMS`) and stochastic (`sDMS`) settings. The `MO-HOO` algorithm by Van Moffaert et al. [133] is as well selected since it is the only algorithm—to the best of our knowledge—that attempts to solve black-box multi-objective problems via partitioning the decision space, hierarchically. From a theoretical perspective, it would be interesting as well to compare `MO-DIRECT` and `MO-SOO`'s empirical performance with that of a uniform-sampling algorithm whose loss bound can be a polynomially-decreasing function of the number of samples. For instance, a uniform grid of s samples, for the bi-objective problem presented in the previous chapter, has a loss $r_j(s)$ of $\mathcal{O}(s^{-\frac{\alpha_j}{n}})$. Default parameters were used for the aforementioned algorithms in line with Section 6.1.2.2.

6.2.5 Results & Discussion

Figures 6.5 and 6.6 show the data profiles of the compared algorithms as a function of the number of functions evaluations used in terms of I_H^-, $I_{\epsilon+}^1$ and I_{GD}, I_{IGD}, respectively. On the other hand, Fig. 6.7 shows the aggregated performance of the same over all the problems and all the quality indicators used.

The performance of the proposed algorithms is consistent with theirs on the BO-BBOB testbed especially in the initial phase of search. With more function evaluations allocated, `SMS-EMOA` and `DMS` exhibit a better performance over all the quality indicators considered. `DMS` performs a line search, consolidating our hypothesis that independent coordinate-wise search (e.g., in `MO-SOO`) is far more efficient that joint coordinate search. This is evident in `MO-DIRECT`, which exhibits a better performance in low-dimensional rather than high-dimensional problems.

6.3 Conclusion

In this chapter, the empirical performance of `MO-DIRECT` and `MO-SOO` in approximating Pareto fronts has been evaluated using 400 (300: BO-BBOB; 100: BMOBench) multi-objective problems and their results are compared with state-of-the art multi-objective solvers. The performance of `MO-SOO` is comparable with best results of the top performing algorithms. From results, we observe that problems with weakly-structured multi-modal objectives impose a challenge for `MO-SOO`. This can

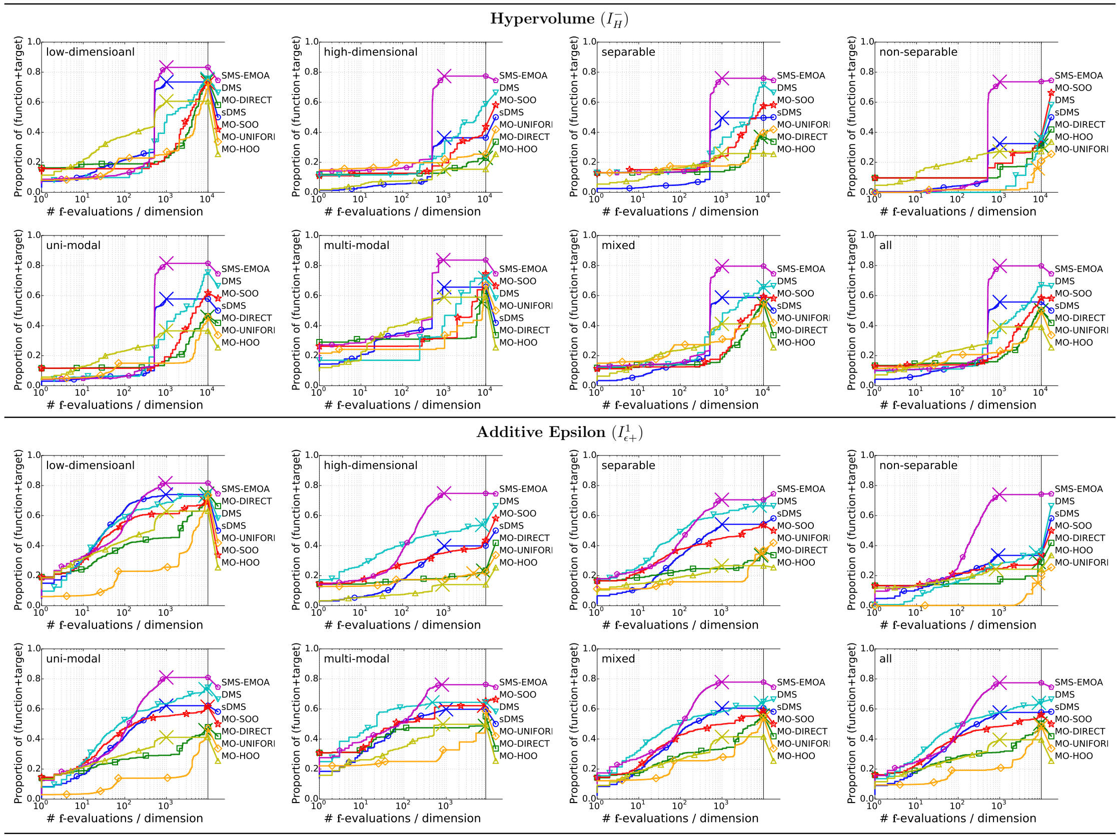


Figure 6.5: Data profiles aggregated over problem categories for the hypervolume and additive epsilon quality indicators computed. The symbol $\times$ indicates the maximum number of function evaluations.

be attributed to two factors: theoretical foundation of the algorithm (the near-optimality dimension) in scaling the exploration proportionally with the number of objective-wise global optima and the fact that sequential partitioning scheme may not adapt well in case of weakly-structured objectives. In addition, the nature of the used `ND(·)` operator overlooks the diversity of the selected nodes for expansion. On the other hand, `MO-DIRECT` exhibits a slow yet consistent performance especially on separable and/or high dimensional problems. This can be attributed to `MO-DIRECT`'s partitioning procedure, which samples on the order of $2n$ points around present solutions, exhausting the search space, inefficiently.

Furthermore, the performance of the indicator-based `SMS-EMOA` validates the efficacy of indicator-based techniques. Incorporating such methods within the MSO framework is straightforward and may be studied as a future investigation.

Besides the numerical assessment, this chapter has presented the Black-box Multi-objective Optimization Benchmarking (BMOBench) platform consolidating

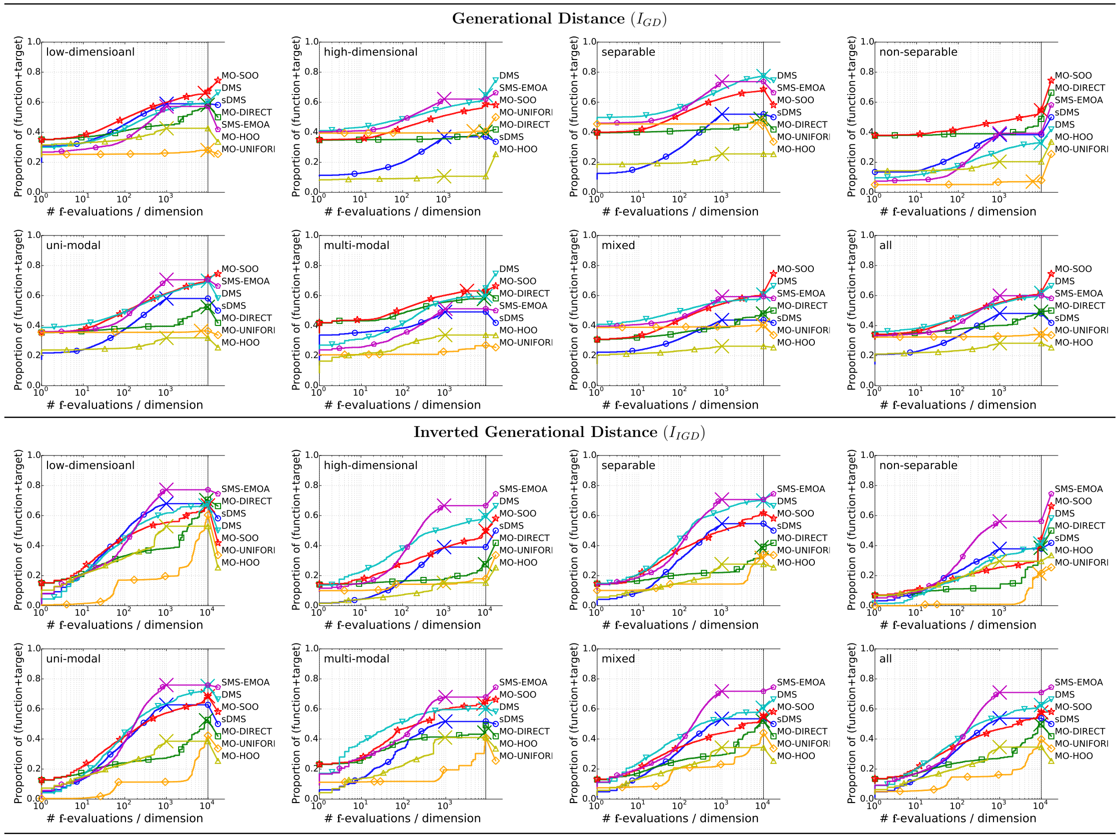


Figure 6.6: Data profiles aggregated over problem categories for the generational and inverted generational distance quality indicators computed. The symbol × indicates the maximum number of function evaluations.

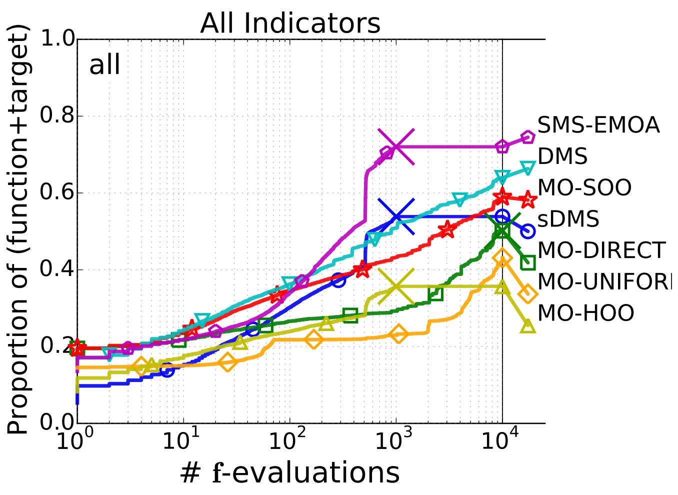


Figure 6.7: Data profiles aggregated over all the problems across all the quality indicators computed for each of the compared algorithms. The symbol × indicates the maximum number of function evaluations.

100 MOPs from the literature. It is hoped that this platform provides the community with a unified environment to test their methods in a reproducible manner.

Chapter 7

Conclusions and Future Works

"People dont like to think, if one thinks, one must reach conclusions. Conclusions are not always pleasant."

- Helen Keller

The present thesis has focused on the development and analysis of a special family of continuous black-box optimization methods. Throughout the research conducted, several lessons have been learned and promising future directions have been identified. In this final chapter, a conclusion to these findings is presented.

7.1 Summary of Contributions

Fueled by the emergence of continuous black-box optimization problems in real-world applications, the numerical optimization community has been actively developing black-box mathematical programming (or so-called derivative-free optimization) techniques through the present and past decades. In line with the community's ongoing effort, the present study is concerned with a class of space-partitioning methods, which we referred to as Multi-scale Search Optimization (MSO) algorithms since they look for the (or one) optimal solution over multiple scales of the decision space. Three key topics within the MSO framework have been particularly addressed, namely i). finite-time convergence; ii). expensive optimization; and iii). multi-objective optimization. In essence, the major contributions of this thesis are:

- Consolidation and validation of a theoretical methodology to study the finite-time convergence of single-objective as well as multi-objective MSO algorithms based on three basic assumptions about the objective smoothness and the decision space partitioning. First time in literature, a deterministic finite-time performance bound is established for multi-objective continuous optimization.

- Development and analysis of an MSO algorithm for expensive continuous black-box single-objective optimization, to which we refer as the Naive Multi-scale Search Optimization (`NMSO`) algorithm.
- Development and analysis of two MSO algorithms for continuous black-box multi-objective optimization, viz. the Multi-Objective DIviding RECTangles (`MO-DIRECT`) and Multi-Objective Simultaneous Optimistic Optimization (`MO-SOO`) algorithms.
- Thorough benchmarking of classical, recent, as well as commercial single-objective MSO algorithms.
- Development of a test suite—which we referred to as the Black-box Multi-objective Optimization Benchmarking (BMOBench)—for continuous black-box multi-objective optimization, consolidating 100 problems from the literature.

Based on the above contributions, the major conclusions of this thesis are summarized as follows.

A Theoretical Framework for Finite-Time Analysis. In Chapter 3, a theoretical methodology to analyze the finite-time behavior of MSO methods was presented. The presented methodology was built on three assumptions with roots in the field of machine learning (multi-armed bandits), to quantify how much search (or how many samples) is required to achieve near-optimal solutions, and was validated empirically using symbolic math. Under the three assumptions of local Hölder continuity of the objective function f, partitions boundedness and partitions sphericity, the worst-case convergence rate of several established MSO algorithms, including `LO` [54, 55], `DOO` [34], `DIRECT` [56], `MCS` [27] and `SOO` [34], is presented/incorporated. While the work of Torczon [118] consolidated the theoretical foundations to analyze and develop direct search methods, the presented theoretical tool in this thesis seeks the same goal with respect to MSO algorithms—i.e., space-partitioning tree search algorithms.

A Finite-Time Pareto-Compliant Indicator Bound. By restricting the smoothness assumption on the objectives (from local to global Hölder continuity assumption A5.1), the theoretical framework described in Chapter 3 is extended in Chapter 5 to establish a finite-time upper bound on the additive ϵ-indicator of the approximation solution set obtained by the proposed `MO-SOO` algorithm as a function of the number of iterations. However, obtaining such a quantity, which can be expressed in terms of the objective-wise loss bounds, is not always an easy task to do for multi-objective MSO algorithms (e.g., `MO-DIRECT`). Furthermore, the presented—first in literature—deterministic finite-time analysis for continuous multi-objective optimization characterizes the performance by the problem in hand, where the bound holds down to the conflict dimension of the problem.

Exploitative Multi-Scale Search Algorithm for Expensive Optimization. Apart from some approaches such as `BB-LS` [52] and `MCS` [27], which couple their space-partitioning search with a local search procedure, MSO methods are biased towards exploration due to their systematic sampling of the decision space. In Chapter 4, the `NMSO` algorithm is presented as an exploitation-biased MSO algorithm whose local search component is integrated within the MSO framework, rather than being a separate procedure. This distinctive feature makes `NMSO` an attractive candidate for optimization problems with limited/expensive evaluation budgets. Besides its asymptotic optimality, `NMSO`'s finite-time analysis provides a generic template to analyze exploitative MSO algorithms with different restart strategies. The efficacy of `NMSO` has been recently validated on the BBComp competition held within the GECCO'16 conference, emerging third out of twenty-eight black-box solvers.

Multi-Scale Search Algorithms for Multi-Objective Optimization. In Chapter 5, the present thesis extended the frontier of MSO framework to multi-objective problems. Two multi-objective algorithmic instances, namely `MO-DIRECT` and `MO-SOO`, were proposed by integrating the non-dominance concept with the notion of the single-objective MSO algorithms, `DIRECT` and `SOO`, respectively. Both algorithms are optimal in the limit, converging to the Pareto front. However, the inclusion of the subspace size as a criterion for further partitioning in `MO-DIRECT` complicates its finite-time analysis compared to MO-SOO's. Furthermore, the performance of MO-DIRECT is adversely affected by its partitioning procedure, whereas the simple coordinate-wise partitioning strategy of MO-SOO appears to be more efficient in using its evaluation budget. Besides its theoretically proven convergence, MO-SOO's performance, in general, is comparable with the state-of-the art algorithms, especially in limited-budget scenarios. Problems with multi-modal objectives impose a challenge on the proposed MSO solvers as the amount of exploration required scales proportionally.

Comprehensive Benchmarking of Multi-Scale Search Algorithms. While the present thesis sought to establish a theoretical tool to analyze MSO algorithms, a thorough empirical performance validation of several MSO algorithms from the literature (BB-LS [52], DIRECT [56], MCS [27], SOO [34], and BaMSOO [5]) has been presented in comparison to the proposed NMSO algorithm using the COCO platform. Overall, NMSO is most suitable in low-budget settings for most of the problems, especially those with separable or multi-modal objectives; BB-LS for ill-conditioned problems, MCS for weakly-structured functions, but it performs poorly on multi-modal problems; BaMSOO's performance improves slowly yet consistently with more function evaluations.

A New Benchmark for Multi-Objective Optimization. The present thesis proposed, in Chapter 6, BMOBench: a benchmarking platform for continuous black-box multi-objective optimization. Coded in MATLAB/C, BMOBench is built around 100 problems from the literature reflecting several real-world difficulties. The platform records the approximation set generated by a multi-objective solver in the course of its run and reports its performance as a function of the number of function evaluations used. The performance report is presented in terms of data profiles of four popular quality indicators (hypervolume, additive epsilon, generational distance, and inverted generational distance) capturing different aspects of the solver's runtime performance. In line with other established testbeds by the community, BMOBench serves the purpose of reproducible research and better understanding of multi-objective algorithms with different evaluation budgets or problem categories. The platform is released as an open source and is flexible enough to incorporate new benchmark problems in the future.

7.2 Future Directions

Based on the present research, we mention here several promising directions for future work with regard to the MSO framework for continuous black-box optimization. One should note that some of these propositions were discussed already in the previous chapters.

Towards Adaptive Partitioning. The bulk of MSO algorithms typically partition the search space along its dimensions/coordinates, giving them equal importance. As evident from the empirical validation (e.g., ill-conditioned problems in Chapter 4), this is not always an efficient strategy to follow. In the literature, there have been some attempts to address such a problem. Proposed by Hansen [192], the adaptive encoding procedure seeks an independence from the problem's coordinates by building a transformed system where the new coordinates are as de-correlated as possible with regard to the objective function. This procedure has been employed by Loshchilov et al. [193] to realize an adaptive variant of the coordinate descent algorithm outperforming the baseline algorithm in solving certain problems. It is therefore interesting to investigate how this and other similar techniques can be incorporated within the MSO framework to break away from boxlike partitions.

Towards Assisted Multi-Scale Search. Given a finite evaluation budget, one seeks to evaluate as few solutions (samples) as possible on the objective function of the problem in hand. To this end, surrogate modeling [66] has been a powerful ingredient for several algorithms tailored towards computationally-expensive optimization problem. With regard to the MSO framework, `BaMSOO` [5] represents a step towards that direction. One can incorporate—besides surrogate models—emerging artificial intelligence (e.g., meta-cognitive [235] and reinforcement learning [179]) techniques to guide the exploratory and exploitative search components of MSO algorithms. Likewise, online parameter tuning can be employed [236]. In the light of recent advances in online learning algorithms, handling general constraints—which has been studied to an extent by Gablonsky [139] for the `DIRECT` [56] algorithm, e.g., the artificial assignment rule—can be re-examined. Furthermore, indicator-based search techniques (e.g., [220]) for multi-objective problems have witnessed an experimental success and it is worthwhile to assess its applicability with respect to MSO.

Towards Large-Scale Problems. On the one hand, real-world problems are increasingly dealing with a high number of decision variables (e.g., training an artificial neural network). On the other hand, the performance of MSO algorithms is adversely affected by the problem dimensionality. Therefore, there is a need for techniques that make the MSO framework scalable. Besides parallelization techniques, methods transforming high-dimensional problems to low-dimensional ones such as space-filling curves [237] and random embedding [194] represent an intriguing topic to explore and apply.

Publications List

Journals

1. A. Al-Dujaili, S. Suresh and N. Sundararajan. MSO: A framework for Bound-Constrained Black-Box Global Optimization Algorithms. *Journal of Global Optimization (JOPT)*, pp:1–35, 2016.

2. A. Al-Dujaili and S. Suresh. Multi-Objective Simultaneous Optimistic Optimization. *Journal of Machine Learning Research (JMLR)*, **Under Review**.

3. A. Al-Dujaili and S. Suresh. A Naïve Multi-Scale Algorithm for Global Optimization Problems. *Information Sciences (Inf. Sci.)*, **Accepted**.

Conference Proceedings

1. A. Al-Dujaili, K. Subramanian and S. Suresh. HumanCog: A cognitive architecture for solving optimization problems. *IEEE Congress onEvolutionary Computation (CEC)*, pp. 3220-3227, May 2015.

2. A. Al-Dujaili and S. Suresh. Analysis of the Bayesian Multi-Scale Optimistic Optimization on the CEC2016 and BBOB Testbeds. *IEEE Congress on Evolutionary Computation (CEC)*, July 2016.

3. A. Al-Dujaili and S. Suresh. Dividing Rectangles Attack Multi-Objective Optimization. *IEEE Congress on Evolutionary Computation (CEC)*, July 2016.

4. A. Al-Dujaili, M. R. Tanweer and S. Suresh. DE vs. PSO: A Performance Assessment for Expensive Problems. *IEEE Symposium Series on Computational Intelligence: IEEE Symposium on Differential Evolution (2015 IEEE SDE)*, December 2015.

5. A. Al-Dujaili, M. R. Tanweer and S. Suresh. On the Performance of Particle Swarm Optimization Algorithms in Solving Cheap Problems. *IEEE Symposium Series on Computational Intelligence: IEEE Swarm Intelligence Symposium (2015 IEEE SIS)*, December 2015.

6. M.R. Tanweer, A. Al-Dujaili and S. Suresh. Empirical Assessment of Human Learning Principles inspired PSO Algorithms on Continuous Black-Box Optimization Testbed. *International Conference on Swarm, Evolutionary and Memetic Computing (SEMCCO 2015)*, 2015.

Workshop Papers

1. A. Al-Dujaili and S. Suresh. A MATLAB Toolbox for Surrogate-Assisted Multi-Objective Optimization: A Preliminary Study. *Genetic and Evolutionary Computation Conference (GECCO 2016)*, Denver, USA, December 2016.

2. C. S. Y. WONG, A. Al-Dujaili and S. Suresh. Hypervolume-based DIRECT for Multi-Objective Optimisation. *Genetic and Evolutionary Computation Conference (GECCO 2016)*, Denver, USA, December 2016.

Technical Reports

1. A. Al-Dujaili and S. Suresh. BMOBench: Black-box multiobjective optimization benchmarking platform. *ArXiv e-prints*, vol. arXiv:1605.07009, 2016.

Bibliography

[1] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, and B. Qu. Problem definition and evaluation criteria for CEC 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization. *Technical Report*, 2015.

[2] D. Hadka. MOEA framework: A free and open source java framework for multiobjective optimization, 2012.

[3] J. Knowles and D. Corne. On metrics for comparing nondominated sets. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 711–716, May 2002. doi: 10.1109/CEC.2002.1007013.

[4] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer, 2002.

[5] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas. Bayesian multi-scale optimistic optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, pages 1005–1014, 2014.

[6] D. Brockhoff, T.-D. Tran, and N. Hansen. Benchmarking Numerical Multiobjective Optimizers Revisited. In *Genetic and Evolutionary Computation Conference (GECCO 2015)*, Madrid, Spain, July 2015. doi: 10.1145/2739480.2754777. URL https://hal.inria.fr/hal-01146741.

[7] G. Gray, T. Kolda, K. Sale, and M. Young. Optimizing an empirical scoring function for transmembrane protein structure determination. *INFORMS Journal on Computing*, 16(4):406–418, 2004. doi: 10.1287/ijoc.1040.0102. URL http://dx.doi.org/10.1287/ijoc.1040.0102.

[8] S. Awasthi. Molecular docking by derivative-free optimization solver. Master's thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, 2008.

[9] J. Han, M. Kokkolaras, and P. Y. Papalambros. Optimal design of hybrid fuel cell vehicles. *Journal of fuel cell science and technology*, 5(4):041014, 2008. doi: 10.1115/1.2890106. URL http://dx.doi.org/10.1115/1.2890106.

[10] B. Depraetere, G. Pinte, and J. Swevers. Iterative optimization of the filling phase of wet clutches. In *Advanced Motion Control, 2010 11th IEEE International Workshop on*, pages 94–99. IEEE, 2010.

[11] J. Hu, Y. Wang, E. Zhou, M. C. Fu, and S. I. Marcus. A survey of some model-based methods for global optimization. In *Optimization, Control, and Applications of Stochastic Systems*, pages 157–179. Springer, 2012.

[12] A. L. Hoffmann, A. Y. Siem, D. den Hertog, J. H. Kaanders, and H. Huizenga. Derivative-free generation and interpolation of convex pareto optimal IMRT plans. *Physics in medicine and biology*, 51(24):6349, 2006.

[13] J. C. Chaput and J. W. Szostak. Evolutionary optimization of a nonbiological ATP binding protein for improved folding stability. *Chemistry & biology*, 11 (6):865–874, 2004.

[14] G. Fursin, A. Cohen, M. OBoyle, and O. Temam. A practical method for quickly evaluating program optimizations. In *High Performance Embedded Architectures and Compilers*, pages 29–46. Springer, 2005.

[15] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani. On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, 10(3):287–299, 2007.

[16] J. Roslund, O. M. Shir, T. Bäck, and H. Rabitz. Accelerated optimization and automated discovery with covariance matrix adaptation for experimental quantum control. *Physical Review A*, 80(4):043415, 2009.

[17] M. Yagoubi. *Optimisation évolutionnaire multi-objectif parallèle: application à la combustion Diesel.* PhD thesis, Université Paris Sud-Paris XI, 2012.

[18] M. Bartholomew-Biggs, S. Parkhurst, and S. Wilson. Using DIRECT to solve an aircraft routing problem. *Computational Optimization and Applications*, 21(3):311–323, 2002. ISSN 1573-2894. doi: 10.1023/A:1013729320435. URL http://dx.doi.org/10.1023/A:1013729320435.

[19] S. Lambot, E. C. Slob, I. Van den Bosch, B. Stockbroeckx, and M. Vanclooster. Modeling of ground-penetrating radar for accurate characterization of subsurface electric properties. *IEEE Transactions on Geoscience and Remote Sensing*, 42(11):2555–2568, 2004.

[20] C. Audet, V. Béchard, and J. Chaouki. Spent potliner treatment process optimization using a MADS algorithm. *Optimization and Engineering*, 9 (2):143–160, 2008. ISSN 1573-2924. doi: 10.1007/s11081-007-9030-2. URL http://dx.doi.org/10.1007/s11081-007-9030-2.

[21] K. Fowler, J. Reese, C. Kees, J. D. Jr., C. Kelley, C. Miller, C. Audet, A. Booker, G. Couture, R. Darwin, M. Farthing, D. Finkel, J. Gablonsky, G. Gray, and T. Kolda. Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743 – 757, 2008. ISSN 0309-1708. doi: http://dx.doi.org/10.1016/j.advwatres.2008.01.010. URL http://www.sciencedirect.com/science/article/pii/S0309170808000110.

[22] T. L. Nguyen and K.-S. Low. A global maximum power point tracking scheme employing direct search algorithm for photovoltaic systems. *IEEE Transactions on Industrial Electronics*, 57(10):3456–3467, 2010.

[23] A. Torn and A. Zilinskas. *Global optimization.* Springer-Verlag New York, Inc., 1989.

[24] A. R. Conn, K. Scheinberg, and P. L. Toint. On the convergence of derivative-free methods for unconstrained optimization. *Approximation theory and optimization: tributes to MJD Powell*, pages 83–108, 1997.

[25] Y. D. Sergeyev. On convergence of "divide the best" global optimization algorithms. *Optimization*, 44(3):303–325, 1998.

[26] C. T. Kelley. *Iterative methods for optimization*, volume 18. SIAM, 1999.

[27] W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14(4):331–355, 1999.

[28] R. M. Lewis and V. Torczon. A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, 2002.

[29] D. E. Finkel and C. Kelley. Convergence analysis of the direct algorithm. Technical report, NCSU Mathematics Department, Raleigh, NC, 2004.

[30] C. Audet and J. E. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[31] A. R. Conn, K. Scheinberg, and L. N. Vicente. Global convergence of general derivative-free trust-region algorithms to first-and second-order critical points. *SIAM Journal on Optimization*, 20(1):387–415, 2009.

[32] Y. D. Sergeyev, M. S. Mukhametzhanov, D. E. Kvasov, and D. Lera. Derivative-free local tuning and local improvement techniques embedded in the univariate global optimization. *Journal of Optimization Theory and Applications*, pages 1–23, 2016. ISSN 1573-2878. doi: 10.1007/s10957-016-0947-5. URL http://dx.doi.org/10.1007/s10957-016-0947-5.

[33] P. Hansen, B. Jaumard, and S.-H. Lu. On the number of iterations of Piyavskii's global optimization algorithm. *Math. Oper. Res.*, 16(2):334–350, April 1991. ISSN 0364-765X. doi: 10.1287/moor.16.2.334. URL http://dx.doi.org/10.1287/moor.16.2.334.

[34] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems 24*, pages 783–791. Curran Associates, Inc., 2011.

[35] A. Auger. *Analysis of Comparison-based Stochastic Continuous Black-Box Optimization Algorithms.* habilitation, Université Paris-Sud, 2016.

[36] D. Finkel and C. Kelley. Additive scaling and the direct algorithm. *Journal of Global Optimization*, 36(4):597–608, 2006.

[37] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45 (3):385–482, 2003.

[38] T.-Y. Lai. Portfolio selection with skewness: A multiple-objective approach. *Review of Quantitative Finance and Accounting*, 1(3):293–305, 1991. ISSN 1573-7179. doi: 10.1007/BF02408382. URL http://dx.doi.org/10.1007/BF02408382.

[39] J. A. Madeira, H. Pina, E. B. Pires, and J. Monteiro. Surgical correction of scoliosis: Numerical analysis and optimization of the procedure. *International Journal for Numerical Methods in Biomedical Engineering*, 26(9): 1087–1098, 2010.

[40] J. Madeira, H. Pina, and H. Rodrigues. GA topology optimization using random keys for tree encoding of structures. *Structural and Multidisciplinary Optimization*, 40(1-6):227–240, 2010. ISSN 1615-147X. doi: 10.1007/s00158-008-0353-1. URL http://dx.doi.org/10.1007/s00158-008-0353-1.

[41] K. Bi, B. Liu, W. Zhang, and R. Nie. Optimization of aircraft flight control system based on an improved mopso algorithm. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 4842–4845, September 2011. doi: 10.1109/ICECENG.2011.6057040.

[42] E. Asadollahi-Yazdi, A. Hassan, A. Siadat, J. Y. Dantan, A. Azadeh, and A. Keramati. Multi-objective optimization for inspection planning using nsga-ii. In *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, pages 1422–1426, December 2015. doi: 10.1109/IEEM.2015.7385882.

[43] Z. Michalewicz. *Advances in Computational Intelligence: IEEE World Congress on Computational Intelligence, WCCI 2012, Brisbane, Australia, June 10-15, 2012. Plenary/Invited Lectures*, chapter Quo Vadis, Evolutionary Computation?, pages 98–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-30687-7. doi: 10.1007/978-3-642-30687-7_6. URL http://dx.doi.org/10.1007/978-3-642-30687-7_6.

[44] I. Loshchilov. *Surrogate-Assisted Evolutionary Algorithms.* Theses, Université Paris Sud - Paris XI ; Institut national de recherche en informatique et en automatique - INRIA, January 2013. URL https://tel.archives-ouvertes.fr/tel-00823882.

[45] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[46] H. Robbins et al. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[47] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Acoustics Speech and Signal ProcessingComputational Intelligence and AI in Games,*, 4(1):1–43, 2012.

[48] G. Chaslot, J.-T. Saito, B. Bouzy, J. Uiterwijk, and H. J. Van Den Herik. Monte-carlo strategies for computer go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*, pages 83–91. Citeseer, 2006.

[49] D. Silver and D. Hassabis. AlphaGo: Mastering the ancient game of Go with machine learning. Google Research Blog, January 2016. URL http://googleresearch.blogspot.com/2016/01/alphago-mastering-ancient-game-of-go.html.

[50] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

[51] P. Rusmevichientong and J. N. Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

[52] J. Pintér. *Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications*, volume 6. Springer Science & Business Media, 1995.

[53] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[54] S. Piyavskii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972.

[55] B. O. Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.

[56] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[57] I. Loshchilov and T. Glasmachers. Black Box Optimization Competition (BBComp). http://bbcomp.ini.rub.de/, 2015.

[58] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012. URL http://coco.gforge.inria.fr/bbob2012-downloads.

[59] M. R. Tanweer, S. Suresh, and N. Sundararajan. Self regulating particle swarm optimization algorithm. *Information Sciences*, 294:182 – 202, 2014.

[60] M. Tanweer, S. Suresh, and N. Sundararajan. Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems. *Information Sciences*, 326: 1 – 24, 2016. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2015.07.035. URL http://www.sciencedirect.com/science/article/pii/S0020025515005319.

[61] R. Tanabe and A. Fukunaga. Tuning differential evolution for cheap, medium, and expensive computational budgets. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 2018–2025, May 2015. doi: 10.1109/CEC.2015.7257133.

[62] L. Bajer, Z. Pitra, and M. Holeňa. Benchmarking gaussian processes and random forests surrogate models on the bbob noiseless testbed. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 1143–1150. ACM, 2015.

[63] P. Baudiš and P. Pošík. Global line search algorithm hybridized with quadratic interpolation and its extension to separable functions. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 257–264. ACM, 2015.

[64] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary computation*, 7(3):205–230, 1999.

[65] A. L. Custódio, J. A. Madeira, A. I. F. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3):1109–1140, 2011.

[66] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[67] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[68] R. Bellman. Dynamic programming princeton university press. *Princeton University Press, NJ*, 1957.

[69] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. PSO facing non-separable and ill-conditioned problems. Technical report, INRIA, 2008.

[70] M. Helbig and A. P. Engelbrecht. *Dynamic Multi-Objective Optimization Using PSO*, pages 147–188. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-30665-5. doi: 10.1007/978-3-642-30665-5_8. URL http://dx.doi.org/10.1007/978-3-642-30665-5_8.

[71] E. J. Anderson and M. C. Ferris. A direct search algorithm for optimization with noisy function evaluations. *SIAM Journal on optimization*, 11(3):837–857, 2001.

[72] Y. G. Evtushenko. Numerical methods for finding global extrema (case of a non-uniform mesh). *USSR Computational Mathematics and Mathematical Physics*, 11(6):38–54, 1971.

[73] V. Ivanov. On optimal algorithms of minimization in the class of functions with the lipschitz condition. *Information Processing*, 71:1324–1327, 1972.

[74] A. G. Sukharev. Optimal strategies of the search for an extremum. *USSR Computational Mathematics and Mathematical Physics*, 11(4):119–137, 1971.

[75] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[76] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[77] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[78] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[79] M. Srinivas and L. M. Patnaik. Genetic algorithms: A survey. *Computer*, 27 (6):17–26, 1994.

[80] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996.

[81] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*, 22(3):52–67, 2002.

[82] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[83] S. Z. Li Xiaolei and Q. Jixin. An optimizing method based on autonomous animates: fish-swarm algorithm. *Systems Engineering Theory & Practice*, 22:32–38, 2004.

[84] O. Syed. *Applying genetic algorithms to recurrent neural networks for learning network parameters and architecture*. PhD thesis, Case Western Reserve University, 1995.

[85] Y. Li, K. H. Ang, G. C. Chong, W. Feng, K. C. Tan, and H. Kashiwagi. Cautocsd-evolutionary search and optimisation enabled computer automated control system design. *International Journal of Automation and Computing*, 1(1):76–88, 2004.

[86] A. George, B. Rajakumar, and D. Binu. Genetic algorithm based airlines booking terminal open/close decision system. In *proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 174–179. ACM, 2012.

[87] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[88] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics (TOG)*, 32(6):206, 2013.

[89] G. E. Box and K. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 13(1):1–45, 1951.

[90] E. Fermi and N. Metropolis. Technical Report Los Alamos Unclassified Report LA-1492, Los Alamos National Laboratory, Los Alamos, NM, 1952.

[91] G. E. Box. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, pages 81–101, 1957.

[92] R. Hooke and T. A. Jeeves. ”direct search” solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

[93] W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.

[94] J. Nelder and R. Mead. The downhill simplex method. *Computer Journal*, 7:308–313, 1965.

[95] R. G. Strongin. *Numerical Methods in Multi-Extremal Problems: Information-Statistical Algorithms*. Nauka, Moscow, Nauka, Moscow, 1978.

[96] R. G. Strongin. On the convergence of an algorithm for finding a global extremum. *Engineering Cybernetics*, 11:549–555, 1973.

[97] R. Fisher. *The design of experiments*. Hafner Press, New York, 1935.

[98] C. Audet and J. E. Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003.

[99] D. H. Winfield. *Function and functional optimization by interpolation in data tables*. PhD thesis, Harvard University, 1970.

[100] T. Glad and A. Goldstein. Optimization of functions whose values are subject to small errors. *BIT Numerical Mathematics*, 17(2):160–169, 1977. ISSN 1572-9125. doi: 10.1007/BF01932287. URL http://dx.doi.org/10.1007/BF01932287.

[101] A. R. Conn, K. Scheinberg, and P. L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical programming*, 79 (1-3):397–414, 1997.

[102] M. J. Powell. Uobyqa: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582, 2002.

[103] T. Winslow, R. Trew, P. Gilmore, and C. T. Kelley. Doping profiles for optimum class B performance of GaAs mesfet amplifiers. In *Proceedings IEEE/Cornell Conference on Advanced Concepts in High Speed Semiconductor Devices and Circuits*, pages 188–197. IEEE, 1991.

[104] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-free Optimization*. MPS-SIAM series on optimization, (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[105] R. Mifflin. A superlinearly convergent algorithm for minimization without evaluating derivatives. *Mathematical Programming*, 9(1):100–117, 1975. ISSN 1436-4646. doi: 10.1007/BF01681333. URL http://dx.doi.org/10.1007/BF01681333.

[106] D. E. Stoneking, G. L. Bilbro, P. A. Gilmore, R. J. Trew, and C. T. Kelley. Yield optimization using a gaas process simulator coupled to a physical device model. *IEEE Transactions on Microwave Theory and Techniques*, 40(7): 1353–1363, 1992.

[107] A. L. Custódio, M. Emmerich, and J. F. A. Maderia. Recent developments in derivative-free multiobjective optimization. *Computational Technology Reviews*, 5:1–30, 2012.

[108] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Athena Scientific, Belmont, MA, 1996.

[109] I. M. The Morgridge Institute for Research. Bound constrained optimization. http://www.neos-guide.org/content/bound-constrained-optimization, December 2013.

[110] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

[111] C. T. Kelley. Detection and remediation of stagnation in the nelder–mead algorithm using a sufficient decrease condition. *SIAM J. on Optimization*, 10(1):43–55, May 1999. ISSN 1052-6234. doi: 10.1137/S1052623497315203. URL http://dx.doi.org/10.1137/S1052623497315203.

[112] M. J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. *Large Scale Nonlinear Optimization*, pages 255–297, 2006.

[113] A. L. Custódio and L. N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18(2):537–555, 2007.

[114] M. A. Abramson, C. Audet, J. E. Dennis Jr, and S. L. Digabel. Orthomads: A deterministic mads instance with orthogonal directions. *SIAM Journal on Optimization*, 20(2):948–966, 2009.

[115] G. Berman. Minimization by successive approximation. *SIAM Journal on Numerical Analysis*, 3(1):123–133, 1966.

[116] E. Polak. *Computational methods in optimization: a unified approach*, volume 77. Academic press, New York, 1971.

[117] W. H. Swann. Direct search methods. In W. Murray, editor, *Numerical Methods for Unconstrained Optimization*, pages 13–28. Academic Press, London and New York, 1972.

[118] V. J. Torczon. *Multidirectional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Houston, TX, USA, 1989. AAI9012878.

[119] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM journal on Optimization*, 1(1):123–145, 1991.

[120] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on optimization*, 7(1):1–25, 1997.

[121] F. H. Clarke. Optimization and nonsmooth analysis, vol. 5 of classics in applied mathematics, 1990.

[122] P. Tseng. Fortified-descent simplicial search method: A general approach. *SIAM Journal on Optimization*, 10(1):269–288, 1999.

[123] L. Nazareth and P. Tseng. Gilding the lily: A variant of the nelder-mead algorithm based on golden-section search. *Computational Optimization and Applications*, 22(1):133–144, 2002. ISSN 1573-2894. doi: 10.1023/A:1014842520519. URL http://dx.doi.org/10.1023/A:1014842520519.

[124] D. Bortz and C. Kelley. The simplex gradient and noisy optimization problems. In *Computational Methods for Optimal Design and Control*, pages 77–90. Springer, 1998.

[125] C. Kelley. Detection and remediation of stagnation in the nelder–mead algorithm using a sufficient decrease condition. *SIAM journal on optimization*, 10(1):43–55, 1999.

[126] Y. M. Danilin. Estimation of the efficiency of an absolute-minimum-finding algorithm. *USSR Computational Mathematics and Mathematical Physics*, 11 (4):261–267, 1971.

[127] C. Elster and A. Neumaier. A grid algorithm for bound constrained optimization of noisy functions. *IMA Journal of Numerical Analysis*, 15(4):585–608, 1995.

[128] T. Choi and C. T. Kelley. Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4):1149–1162, 2000.

[129] C. Audet and M. Kokkolaras. Blackbox and derivative-free optimization: theory, algorithms and applications. *Optimization and Engineering*, 17(1): 1–2, 2016. ISSN 1573-2924. doi: 10.1007/s11081-016-9307-4. URL http://dx.doi.org/10.1007/s11081-016-9307-4.

[130] C. Audet, G. Savard, and W. Zghal. A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research*, 204(3):545–556, 2010.

[131] L. Wang, H. Ishida, T. Hiroyasu, and M. Miki. Examination of multi-objective optimization method for global search using direct and ga. In *IEEE World Congress on Computational Intelligence*, pages 2446–2451. IEEE, 2008.

[132] W. Wang and M. Sebag. Multi-objective Monte-Carlo Tree Search. In S. C. Hoi and W. Buntine, editors, *Asian Conference on Machine Learning*, volume 25, pages 507–522, Singapour, Singapore, November 2012. URL https://hal.inria.fr/hal-00758379.

[133] K. Van Moffaert, K. Van Vaerenbergh, P. Vrancx, and A. Nowé. Multi-objective χ-armed bandits. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 2331–2338. IEEE, 2014.

[134] F. Archetti and B. Betrò. A priori analysis of deterministic strategies for global optimization problems. *Towards Global Optimization*, 2:31, 1978.

[135] D. Mayne and E. Polak. Outer approximation algorithm for nondifferentiable optimization problems. *Journal of Optimization Theory and Applications*, 42(1):19–30, 1984. ISSN 0022-3239. doi: 10.1007/BF00934131. URL http://dx.doi.org/10.1007/BF00934131.

[136] R. Horst and H. Tuy. On the convergence of global methods in multiextremal optimization. *Journal of Optimization Theory and Applications*, 54(2):253–271, 1987.

[137] J. Pintér. Globally convergent methods for n-dimensional multiextremal optimization. *Optimization*, 17(2):187–202, 1986.

[138] R. J. Vanderbei. Extension of piyavskii's algorithm to continuous global optimization. *Journal of Global Optimization*, 14(2):205–216, 1999.

[139] J. Gablonsky. *Modifications of the Direct Algorithm.* PhD thesis, North Carolina State University, Raleigh, North Carolina, 2001.

[140] R. Carter, J. Gablonsky, A. Patrick, C. Kelley, and O. Eslinger. Algorithms for noisy problems in gas transmission pipeline optimization. *Optimization and engineering*, 2(2):139–157, 2001.

[141] F. H. Mladineo. An algorithm for finding the global maximum of a multimodal, multivariate function. *Math. Program.*, 34(2):188–200, March 1986. ISSN 0025-5610. doi: 10.1007/BF01580583. URL http://dx.doi.org/10.1007/BF01580583.

[142] G. L. N. Laurence A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, New York, 1988.

[143] J. M. Fowkes, N. I. Gould, and C. L. Farmer. A branch and bound algorithm for the global optimization of hessian lipschitz continuous functions. *Journal of Global Optimization*, 56(4):1791–1815, 2013.

[144] Y. Evtushenko and M. Posypkin. A deterministic approach to global box-constrained optimization. *Optimization Letters*, 7(4):819–829, 2013.

[145] Y. D. Sergeyev. A one-dimensional deterministic global minimization algorithm. *Computational mathematics and mathematical physics*, 35(5):705–717, 1995.

[146] Y. D. Sergeyev and R. G. Strongin. A global minimization algorithm with parallel iterations. *USSR Computational Mathematics and Mathematical Physics*, 29(2):7–15, 1990.

[147] R. G. Strongin and Y. D. Sergeyev. Global multidimensional optimization on parallel computer. *Parallel Computing*, 18(11):1259–1273, 1992.

[148] Y. G. Evtushenko, V. Malkova, and A. Stanevichyus. Parallel global optimization of functions of several variables. *Computational Mathematics and Mathematical Physics*, 49(2):246–260, 2009.

[149] J. Gablonsky. An implementation of the direct algorithm. Technical Report CRSC-TR98-29, North Carolina State University, Centre for Research in Scientific Computing, August 1998.

[150] J. M. Gablonsky and C. T. Kelley. A locally-biased form of the direct algorithm. *Journal of Global Optimization*, 21(1):27–37, 2001.

[151] Y. D. Sergeyev and D. E. Kvasov. Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM Journal on Optimization*, 16(3):910–937, 2006.

[152] D. E. Kvasov and Y. D. Sergeyev. Lipschitz gradients for global optimization in a one-point-based partitioning scheme. *Journal of Computational and Applied Mathematics*, 236(16):4042–4054, 2012.

[153] R. Paulavičius and J. Žilinskas. *Simplicial global optimization*. Springer, New York, 2014.

[154] R. Paulavičius, Y. D. Sergeyev, D. E. Kvasov, and J. Žilinskas. Globally-biased DISIMPL algorithm for expensive global optimization. *Journal of Global Optimization*, 59(2-3):545–567, 2014.

[155] Y. D. Sergeyev and D. E. Kvasov. A deterministic global optimization using smooth diagonal auxiliary functions. *Communications in Nonlinear Science and Numerical Simulation*, 21(1):99–111, 2015.

[156] D. E. Kvasov and Y. D. Sergeyev. Deterministic approaches for solving practical black-box global optimization problems. *Advances in Engineering Software*, 80:58–66, 2015.

[157] P. Pošík, W. Huyer, and L. Pál. A comparison of global search algorithms for continuous black box optimization. *Evolutionary computation*, 20(4):509–541, 2012.

[158] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

[159] S. Bubeck, G. Stoltz, C. Szepesvári, and R. Munos. Online optimization in x-armed bandits. In *Advances in Neural Information Processing Systems*, pages 201–208, 2009.

[160] M. Valko, A. Carpentier, and R. Munos. Stochastic simultaneous optimistic optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 19–27, 2013.

[161] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *The Journal of Machine Learning Research*, 12:1655–1695, 2011.

[162] R. M. Lewis, V. Torczon, and M. W. Trosset. Direct search methods: then and now. *Journal of computational and Applied Mathematics*, 124(1):191–207, 2000.

[163] C. T. Kelley. *Implicit filtering*, volume 23. SIAM, 2011.

[164] D. Ratz and T. Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization*, 7(2):183–207, 1995. ISSN 0925-5001. doi: 10.1007/BF01097060. URL http://dx.doi.org/10.1007/BF01097060.

[165] T. Csendes and D. Ratz. Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, 34(3):922–938, 1997.

[166] D. E. Kvasov, C. Pizzuti, and Y. D. Sergeyev. Local tuning and partition strategies for diagonal go methods. *Numerische Mathematik*, 94(1):93–106, 2003.

[167] V. Pareto. *Manual of political economy*. Augustus M. Kelley Publishers, New York, 1971.

[168] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[169] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multi-objective optimizers. TIK-Report 214, Computer Engineering and Networks Laboratory, ETH Zurich, Gloriastrasse 35, ETH-Zentrum, 8092 Zurich, Switzerland, February 2006.

[170] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *evolutionary computation, IEEE transactions on*, 3(4):257–271, 1999.

[171] D. A. V. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University, June 1999.

[172] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM symposium on Applied computing*, pages 351–357. ACM, 1999.

[173] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[174] C.-L. Hwang and A. S. M. Masud. Multiple objective decision making methods and applications: A state-of-the-art survey. *Lecture Notes in Econom. Math. Systems*, 164, 1979. Springer-Verlag, Berlin.

[175] K. Miettinen. *Nonlinear multiobjective optimization.* Kluwer, Boston, MA, USA, 1999.

[176] S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *The Journal of Machine Learning Research*, 5:325–360, 2004.

[177] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation (CEC)*, 6(2):182–197, 2002.

[178] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. G. et al., editor, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[179] Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *ICML*, volume 98, pages 197–205, 1998.

[180] S. Natarajan and P. Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 601–608. ACM, 2005.

[181] L. Barrett and S. Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on Machine learning*, pages 41–47. ACM, 2008.

[182] H. Liao, Q. Wu, and L. Jiang. Multi-objective optimization by reinforcement learning for power system dispatch and voltage stability. In *Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES*, pages 1–8. IEEE, 2010.

[183] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 2013.

[184] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007.

[185] J. Cousty, L. Najman, and B. Perret. Constructive links between some morphological hierarchies on edge-weighted graphs. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 86–97. Springer, 2013.

[186] P. Preux, R. Munos, and M. Valko. Bandits attack function optimization. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2245–2252. IEEE, 2014.

[187] C. Stover and E. W. Weisstein. Hölder condition. MathWorld–A Wolfram Web Resource. URL http://mathworld.wolfram.com/HoelderCondition.html.

[188] F. H. Clarke. Nonsmooth analysis and optimization. In *Proceedings of the International Congress of Mathematicians (Helsinki, 1978)*, pages 847–853, 1983.

[189] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *arXiv preprint arXiv:1603.08785*, 2016.

[190] J. Pintér, K. Holmström, A. Göran, and M. Edvall. *User's Guide for TOMLAB/LGO*. Tomlab Optimization, 2006. URL `http://tomopt.com/docs/TOMLAB_LGO.pdf`.

[191] K. Price. Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157, 1997.

[192] N. Hansen. Adaptive encoding: How to render search coordinate system invariant. In *International Conference on Parallel Problem Solving from Nature*, pages 205–214. Springer, 2008.

[193] I. Loshchilov, M. Schoenauer, and M. Sebag. Adaptive coordinate descent. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 885–892. ACM, 2011.

[194] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, N. de Freitas, et al. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, pages 1778–1784. AAAI Press/International Joint Conferences on Artificial Intelligence, 2013.

[195] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[196] I. Erlich, G. Venayagamoorthy, and N. Worawat. A mean-variance optimization algorithm. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6, July 2010. doi: 10.1109/CEC.2010.5586027.

[197] A. Jiménez. Sparse hessian factorization in curved trajectories for unconstrained minimization. *Optimization Methods and Software*, 29(1):1–9, 2014. doi: 10.1080/10556788.2012.707652.

[198] A. Costa and G. Nannicini. RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Optimization Online*, (4538), 2014.

[199] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785–1791. IEEE, 2005.

[200] J. Tvrdík. Adaptation in differential evolution: A numerical comparison. *Applied Soft Computing*, 9(3):1149–1155, 2009.

[201] I. Křivỳ and J. Tvrdík. The controlled random search algorithm in optimizing regression models. *Computational statistics & data analysis*, 20(2):229–234, 1995.

[202] P. Conejo, E. Karas, and L. Pedroso. A trust-region derivative-free algorithm for constrained optimization. *Optimization Methods and Software*, (ahead-of-print):1–20, 2014.

[203] K. Makukhin. Evolution strategies with an rbm-based meta-model. In Y. Kim, B. Kang, and D. Richards, editors, *Knowledge Management and Acquisition for Smart Systems and Services*, volume 8863 of *Lecture Notes in Computer Science*, pages 246–259. Springer International Publishing, 2014. ISBN 978-3-319-13331-7. doi: 10.1007/978-3-319-13332-4_20. URL http://dx.doi.org/10.1007/978-3-319-13332-4_20.

[204] B. A. Tolson and C. A. Shoemaker. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. *Water Resources Research*, 43(1), 2007.

[205] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[206] P. Lopez-Garcia, E. Onieva, E. Osaba, A. D. Masegosa, and A. Perallos. Hybridizing genetic algorithm with cross entropy for solving continuous functions. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 763–764. ACM, 2015.

[207] W. Huang and H. Li. On the differential evolution schemes in moea/d. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, volume 6, pages 2788–2792. IEEE, 2010.

[208] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by moea/d with gaussian process model. *Evolutionary Computation, IEEE Transactions on*, 14(3):456–474, 2010.

[209] N. Al Moubayed, A. Petrovski, and J. McCall. A novel smart multi-objective particle swarm optimisation using decomposition. In *Parallel Problem Solving from Nature, PPSN XI*, pages 1–10. Springer, 2010.

[210] A. Waldock and D. Corne. Multiple objective optimisation applied to route planning. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1827–1834. ACM, 2011.

[211] M. M. Drugan and A. Nowe. Designing multi-objective multi-armed bandits algorithms: A study. In *The International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[212] K. Li, Á. Fialho, S. Kwong, and Q. Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014.

[213] P. Vamplew, R. Dazeley, A. Berry, E. Dekker, and R. Issabekov. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1-2):51–80, 2011.

[214] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler. Do additional objectives make a problem harder? In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 765–772. ACM, 2007.

[215] G. Rudolph. Evolutionary search for minimal elements in partially ordered finite sets. In *Evolutionary Programming VII*, pages 345–353. Springer, 1998.

[216] R. Kumar and N. Banerjee. Running time analysis of a multiobjective evolutionary algorithm on simple and hard problems. In *Foundations of Genetic Algorithms*, pages 112–131. Springer, 2005.

[217] T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, 117(3):553–564, 1999.

[218] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions,. Technical Report RR-6829, INRIA, 2009, updated February 2010. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads.

[219] T. Voß, N. Hansen, and C. Igel. Improved step size adaptation for the mo-cma-es. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 487–494. ACM, 2010.

[220] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[221] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[222] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), 1999.

[223] R. Munos. From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–130, 2014. URL `http://hal.archives-ouvertes.fr/hal-00747575`.

[224] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, 10(5):477–506, Oct 2006. ISSN 1089-778X. doi: 10.1109/TEVC.2005.861417.

[225] K. Deb, A. Sinha, and S. Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1141–1148. ACM, 2006.

[226] F. Cheng and X. Li. Generalized center method for multiobjective engineering optimization. *Engineering Optimization*, 31(5):641–661, 1999.

[227] G. Liuzzi, S. Lucidi, F. Parasiliti, and M. Villani. Multiobjective optimization techniques for the design of induction motors. *IEEE Transactions on Magnetics*, 39(3):1261–1264, 2003.

[228] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA)*, pages 825–830. Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA), 2002.

[229] J. Nocedal and S. J. Wright. *Numerical optimization*, volume 2. Springer-Verlag, Berlin, 2 edition, 2006.

[230] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(1):26–37, 1998.

[231] S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In *Evolutionary multi-criterion optimization*, pages 280–295. Springer, 2005.

[232] L. N. Vicente and A. Custódio. Analysis of direct searches for discontinuous functions. *Mathematical programming*, 133(1-2):299–325, 2012.

[233] F. Kursawe. A variant of evolution strategies for vector optimization. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, PPSN I, pages 193–197, London, UK, UK, 1991. Springer-Verlag. ISBN 3-540-54148-9. URL http://dl.acm.org/citation.cfm?id=645821.670214.

[234] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

[235] K. Subramanian and S. Suresh. A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system. *Applied Soft Computing*, 12(11): 3603–3614, 2012.

[236] R. Tanabe and A. S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 1658–1665, July 2014. doi: 10.1109/CEC.2014.6900380.

[237] D. Lera and Y. Sergeyev. Lipschitz and hlder global optimization using space-filling curves. *Applied Numerical Mathematics*, 60(1):115 – 129, 2010. ISSN 0168-9274. doi: http://dx.doi.org/10.1016/j.apnum.2009.10.004. URL http://www.sciencedirect.com/science/article/pii/S0168927409001688.