Hybrid Algorithm for Multi-objective Mixed-integer Non-convex Mechanical Design Optimization Problems

Ahmed Jaber

▶ **To cite this version:**

Ahmed Jaber. Hybrid Algorithm for Multi-objective Mixed-integer Non-convex Mechanical Design Optimization Problems. Mechanical engineering [physics.class-ph]. Université de Technologie de Troyes; Université Libanaise, 2021. English. NNT : 2021TROY0034 . tel-03810704

HAL Id: tel-03810704
https://theses.hal.science/tel-03810704

Submitted on 11 Oct 2022

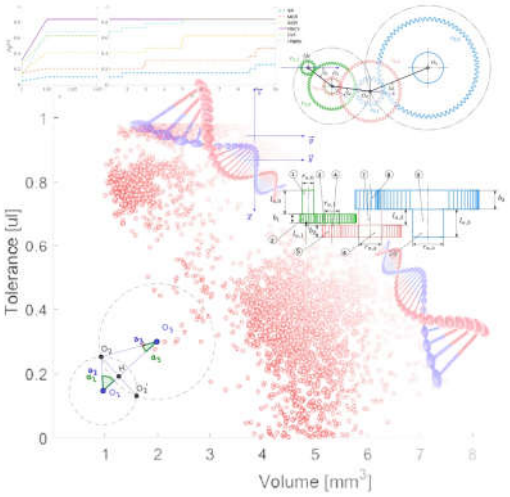
HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse
de doctorat
de l'UTT

Ahmed JABER

Hybrid Algorithm for Multi-objective Mixed-integer Non-convex Mechanical Design Optimization Problems



Champ disciplinaire :
Sciences pour l'Ingénieur

2021TROY0034 **Année 2021**

Thèse en cotutelle avec l'Université Libanaise - Beyrouth - Liban

Université Libanaise

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

en SCIENCES POUR L'INGENIEUR

Spécialité : MATERIAUX, MECANIQUE, OPTIQUE, NANOTECHNOLOGIE

présentée et soutenue par

Ahmed JABER

le 9 novembre 2021

Hybrid Algorithm for Multi-objective Mixed-integer Non-convex Mechanical Design Optimization Problems

JURY

M. Rodolphe LE RICHE	DIRECTEUR DE RECHERCHE CNRS	Président
M. Fouad BENNIS	PROFESSEUR DES UNIVERSITES	Rapporteur
M. Manuel PAREDES	PROFESSEUR DES UNIVERSITES	Rapporteur
Mme Laurence MOREAU	MAITRE DE CONFERENCES - HDR	Examinatrice
M. Ahmed SAMROUT	DOCTEUR	Examinateur
M. Pascal LAFON	PROFESSEUR DES UNIVERSITES	Directeur de thèse
M. Rafic YOUNES	PROFESSEUR	Directeur de thèse

Dedicated to the memory of my brother Hassan

Acknowledgments

Completing my Ph.D. has been one of the most significant academic challenges that I ever had to face, but provided me a remarkable experience. Accomplishing this academic challenge would not have been possible without many people, not only for their contributions to scientific knowledge, but also for my personal benefit from the privilege of working alongside them during this journey. These lines are dedicated to all of them.

First and foremost, I would like to thank my advisor **Pascal Lafon**. He does not deserve only my thanks, but also direct credit for some of the work this thesis represents. In fact, much of the intellectual work of this thesis took place in conversations with him. Pascal has not just been a great advisor, but a great friend as well. I am very grateful for her patience, motivation, and immense knowledge in the field, which made this thesis as successful as it is. I would also like to thank my second advisor **Rafic Younes**, who has always been guiding me in the right direction. His excellent research skills are one of a kind, allowing me to work on a high professional level. I simply could not wish for better advisors.

I would like to thank the reviewers **Manuel Paredes** and **Fouad Bennis**, and the committe members **Roldolphe Leriche**, **Laurence Moreau** and **Ahmad Samrout** for their acceptance to read my thesis and share their reviews and thoughts with me.

The monitoring committee that has followed me during these years, has had an important positive impact on the quality of the work achieved in this thesis. The constructive remarks and the detailed follow-up provided by its members have allowed me to continuously enhance my work to meet their high set standards. I thank the monitoring committee members in particular the external member **Manuel Paredes** who gave me continuous and important advices that helped to improve our work significantly.

I also thank the doctoral schools of the University of Technology of Troyes and the Lebanese University, in particular the director **Khemais Saanouni**, and the secretaries **Pascale Denis**, **Isabelle Leclercq** and **Thérèse Kazarian**.

I would also like to take the opportunity to thank my friends and colleagues for their support during the last three years. Special thanks go to my friends Daniel, Abed, Fadel, Samer, Hala, Rasha, Khadija, Hussein, Ali, Melissa, Sania, Ziad, Hawraa, Dalia, and Noor .

My biggest thank go, without question, to my family, for my Parents Mohamad and Mariam, my brothers and sisters Doaa, Ali, Ayat, and Jihad, and my fiancee Batool for their support during these years.

Abstract

Multi-objective mixed-integer non-convex non-linear constrained optimization problems that appears in several fields especially in mechanical applications. This thesis aims to develop a new method to solve such problems. Our proposal is a hybridization of the Multi-Criteria Branch-and-Bound (MCBB) algorithm with the Non-dominated Sorting Genetic Algorithm 2 (NSGAII). The proposed approach is furthermore enhanced by new branching strategies designed for MCBB. The constraints are handled using a new proposed constraint handling technique for evolutionary algorithms. Numerical experiments based on statistical assessment are done in this thesis to examine the performance of the new proposed approach. Results show the competitive performance of our algorithm among NSGAII. We propose two applications of our proposed approach: "Search Feasibility" and "Seek Optimality" applications. Both are applied on a real-world state of art 3 stages reducer problem which is formulated in this thesis to a bi-objective problem to meet the requirement of ISO standards on calculation of load capacity of gears.

Keywords:

- Mathematical optimization
- Metaheuristic
- Evolutionary programming (computer science)
- Multiple Criteria Decision Making
- Mechanics
- Engineering Design

Resumè

Les problèmes d'optimisation sous contraintes non linéaires non convexes multi-objectifs en variables mixtes (discrètes et continues) apparaissent dans de nombreux domaines de l'ingénierie et notamment dans les applications de conception en mécaniques. Cette thèse vise à développer une nouvelle méthode pour résoudre ces problèmes d'optimisation. Notre proposition est une hybridation de l'algorithme multicritère ≪ Branch-and-Bound ≫ (MCBB) avec l'algorithme évolutionnaire de type NSGAII. L'approche proposée est en outre renforcée par de nouvelles stratégies de branchement conçues pour l'algorithme MCBB. Les contraintes du problème d'optimisation sont gérées à l'aide d'une nouvelle technique dédiées aux algorithmes évolutionnaires. Les performances de cette nouvelle approche sont évaluées et comparées à l'existant par une étude statistique sur un ensemble de problèmes tests. Les résultats montrent que les performances de notre algorithme sont compétitives face à l'algorithme NSGAII seul. Nous proposons deux applications de notre algorithme : les applications "Recherche de solutions faisables" et "Recherche de solutions optimales". Celles-ci sont appliquées sur un problème industriel réel d'un réducteur à engrenages à 3 étages formulé comme un problème bi-objectif. Dans ce problème des contraintes sont incluses pour satisfaire aux exigences de normes ISO sur le calcul de la capacité de charge des engrenages.

Mots-clés :

- Optimisation Mathématique
- Métaheuristiques
- Programmation évolutionnaire
- Décision Multicritère
- Mécanique
- Conception Technique

Table of Contents

List of Figures xv

List of Tables xix

I English version 1

1 General Introduction 3

1.1 Introduction . . . 4

1.2 Thesis Organization . . . 7

2 State of Art 9

2.1 Overview . . . 10

2.1.1 Taxonomy . . . 10

2.1.1.1 Exact Methods (E) . . . 10

2.1.1.2 Approximate Methods (A) . . . 12

2.1.1.3 Problems Characteristics (P) . . . 12

2.1.1.4 Transformation Techniques (T) . . . 15

2.1.2 Scope . . . 16

2.2 Exact Methods for MO-MINLPs . . . 17

2.2.1 Multi-Objective . . . 18

2.2.1.1 Scalarization Methods (T1) . . . 18

2.2.1.2 Multi-Criteria Branch and Bound (E2.1) . . . 19

2.2.2 Mono-Objective MINLP . . . 20

2.2.2.1 Convex Problems (P4.2.1) . . . 21

2.2.2.2 Non-Convex Problems (P4.2.2) . . . 23

2.2.3 Drawbacks and Conclusions . . . 24

2.3 Approximate Methods . . . 26

2.3.1 Multi-Objective Metaheuristics . . . 27

2.3.1.1 Fitness Assignment . . . 28
2.3.1.2 Diversity . . . 28
2.3.1.3 Elitism . . . 28
2.3.2 Evolutionary Algorithms . . . 29
2.3.2.1 Genetic Algorithm . . . 29
2.3.2.2 NSGAII . . . 30
2.3.3 Drawbacks and Conclusions . . . 31
2.4 Hybrid Methods . . . 32
2.4.1 Hybrids Classifications . . . 33
2.4.1.1 Metaheuristics with Metaheuristics (A2+A2) . . . 33
2.4.1.2 Exact with Approximate (E+A) . . . 34
2.4.1.3 Multi-objective Hybrids . . . 35
2.4.2 Combining Approximate Methods with Tree Search (A + E2.1) . . . 36
2.4.2.1 Combining mono-objective branch and bound with metaheuristics (A2+E2.1) . . . 36
2.4.2.2 Combining multi-objective branch and bound with heuristics (A1 + E2.1) . . . 37
2.4.2.3 Discussion and Conclusion . . . 38
2.5 Conclusion and Propositions . . . 38

3 Hybrid Branch and Bound Based Genetic Algorithm 41
3.1 Introduction . . . 42
3.2 MCBB . . . 43
3.3 BnB-NSGAII . . . 44
3.3.1 Parameters Tuning . . . 45
3.3.2 BnB-NSGAII Algorithm . . . 47
3.3.3 BnB-NSGAII Legacy Feature . . . 49
3.4 Branching Strategy . . . 50
3.4.1 Branching by Anchor Points . . . 52
3.4.1.1 Branching Algorithm . . . 52
3.4.1.2 Constructing the Combinatorial Tree . . . 53
3.4.2 Branching by Pareto Elements . . . 55
3.5 Constraint Handling Technique . . . 56
3.5.1 Literature Review . . . 57
3.5.1.1 Preserving feasibility methods . . . 57
3.5.1.2 Search for feasibility methods . . . 58
3.5.1.3 Penalty Methods . . . 59

3.5.1.4 Ranking-Based Methods . . . 61
3.5.2 NSCV . . . 63
3.5.2.1 Non-Dominated Sorting . . . 64
3.5.2.2 NSCV Concept . . . 65
3.6 Conclusion . . . 67

4 Numerical Experiment **71**
4.1 Introduction . . . 72
4.2 BnB-NSGAII . . . 73
4.2.1 Test Problems . . . 73
4.2.2 Evaluation Criteria . . . 76
4.2.2.1 Cardinality Indicators . . . 76
4.2.2.2 Convergence Metrics . . . 77
4.2.2.3 Distribution and spread indicators . . . 77
4.2.2.4 Computational Effort . . . 78
4.2.2.5 Robustness . . . 78
4.2.2.6 Investment Ratio . . . 78
4.2.3 Benchmark Experiment . . . 79
4.2.4 Results and discussion . . . 80
4.3 Branching Strategies . . . 89
4.3.1 Results and Discussion . . . 93
4.4 NSCV . . . 96
4.4.1 Profile Performance . . . 97
4.4.2 Results and Discussion . . . 99
4.5 Conclusion . . . 102

5 Applications **105**
5.1 Introduction . . . 106
5.2 3 stages Reducer Problem . . . 107
5.2.1 Closure condition . . . 108
5.2.2 Mechanical constraint for one stage of the mechanism . . . 110
5.2.2.1 Constraints related to the gear pair . . . 110
5.2.2.2 Constraint related to shaft's reducer . . . 112
5.2.3 3SR Complexity . . . 112
5.3 An Application of BnB-NSGAII: Search Feasibility . . . 114
5.3.1 Numerical Experiment . . . 114
5.3.1.1 Results and Discussion . . . 114
5.4 NSCV Case Study . . . 116

5.5 An Application of BnB-NSGAII: Seek Optimality . . . 119
5.6 Conclusion . . . 121

6 Conclusion **123**
6.1 Perspectives . . . 124
6.2 Publications . . . 125

II French version **127**

Résumé étendu en Français **129**
1 Introduction . . . 130
2 État de l'art . . . 131
2.1 Techniques de manipulation des variables mixtes . . . 132
2.1.1 Méthodes exactes . . . 132
2.1.2 Inconvénients . . . 133
2.1.3 Méthodes Approximatives . . . 133
2.1.4 Inconvénients . . . 135
2.1.5 Hybridation . . . 136
2.2 Techniques pour manipuler les contraintes . . . 137
3 Hybridation "Branch and Bound" et algo. génétique . . . 139
3.1 Principes de l'algorithme BnB-NSGAII . . . 139
3.2 Stratégies de Branchement . . . 140
3.3 NSCV . . . 142
4 Expériences numériques . . . 142
4.1 Algorithme BnB-NSGAII . . . 143
4.2 Stratégies de branchement . . . 145
4.3 Technique NSCV . . . 146
5 Applications : réducteur à engrenages à 3 étages . . . 147
5.1 Introduction . . . 147
5.2 Recherche de solutions faisables . . . 148
5.3 Recherche de solutions optimales . . . 149
5.4 Technique de gestion de contraintes NSCV . . . 150
6 Conclusion . . . 150

Bibliography **153**

A Benchmark Problems **165**
1 Problems Formulation . . . 166
1.1 Gear Problem . . . 166
1.2 Bearing Problem . . . 166
1.3 Coupling Problem . . . 168
1.4 Disk Brake Problem . . . 171
1.5 Truss Problem . . . 172
1.6 Mela Mathematical Problem . . . 172
1.7 Tong Mathematical Problem . . . 173

B 3 Stages Reducer Problem **175**
1 Variables . . . 176
2 Reducer Data . . . 177
3 Gear Data . . . 178
4 Standard Modules . . . 179

C Numerical Experiment Data **181**
1 BnB-NSGAII Tuning Parameters . . . 181
2 NSGAII Parameters Tuning . . . 184

List of Figures

1.1 A 3-D view of the 3 stages reducer. 4
1.2 Example of mechanical design problem: 3 stages reducer. 5
1.3 Thesis Outline. 7

2.1 Taxonomy of exact algorithms. 11
2.2 Implicit enumeration of the domain space by branch and bound algorithm. . . . 11
2.3 Taxonomy of approximate algorithms. 12
2.4 Taxonomy of optimization problems based on their characteristics. 13
2.5 Taxonomy of transformation techniques of the optimization problems. 15
2.6 MCBB components review chart. 21
2.7 Exact methods review chart (*color-coded to meet the presented taxonomy*). 25
2.8 Classification of metaheuristics [Dréo & Candan, 2007]. 27
2.9 Combining approximate methods with tree search. 37
2.10 Hybrid tree search with approximate methods review chart. 39
2.11 An overview of the proposed approach. 40

3.1 Illustrative example of the branching tree of MCBB on the problem $\mathcal{P}_{\text{MO-MINLP}}$. 43
3.2 Flowchart of MCBB algorithm. 44
3.3 Illustrative example of the branching tree of BnB-NSGAII on the problem $\mathcal{P}_{\text{MO-MINLP}}$ 46
3.4 Example of updating the incumbent list in BnB-NSGAII. 49
3.5 Concept of the legacy feature in BnB-NSGAII approach. 50
3.6 Example of branching by integer strategy. 51
3.7 Example of BA strategy for a bi-objective problem. 52
3.8 Flow chart of MCBB with BA strategy. 54
3.9 Example of BP strategy for bi-objective problem. 55
3.10 Graphical view of a general decoder [Mezura-Montes & Coello Coello, 2011]. . . . 58
3.11 Non-dominated sorting of the population of a bi-objective problem 64

3.12 Illustrative example plotted on the projected constraint domain 66
3.13 Benchmarking and applications of the proposed approach. 68

4.1 The attainable set of solutions of the test problems. 74
4.1 The attainable set of solutions of the test problems. 75
4.2 Distribution of metrics for gear problem. 81
4.3 Distribution of metrics for bearing problem. 82
4.4 Distribution of metrics for coupling problem. 83
4.5 Distribution of metrics for disk brake problem. 84
4.6 Distribution of metrics for truss problem. 85
4.7 Distribution of metrics for Mela problem. 86
4.8 Distribution of metrics for Tong problem. 87
4.9 Pareto front obtained by BnB-NSGAII and NSGAII vs the true Pareto for each problem. 90
4.9 Pareto front obtained by BnB-NSGAII and NSGAII vs the true Pareto for each problem. 91
4.10 Distribution of total nodes exploration with respect to the separation orders on each problem. 93
4.11 Winning ratio of each strategy on each problem based on total nodes exploration ratio. 94
4.12 Distribution of leaf nodes exploration with respect to the separation orders on each problem. 95
4.13 Winning ratio of each strategy on each problem based on leaf nodes exploration ratio. 96
4.14 Example of profile performance 100
4.15 Profile performance of CHTs on G-Problems set 100
4.16 Ratio of failure of each CHT on G-Problems set 101
4.17 Overall mean feasible density on G-Problems set 101
4.18 Profile performance of each CHT on RW-Problems set 101
4.19 Ratio of failure of each CHT on RW-Problems set 101
4.20 Overall mean feasible density on RW-Problems set 101

5.1 Benchmarking of BnB-NSGAII on the 3 stages reducer problem. 107
5.2 Front and back view of a 3 stages reducer with closure. 108
5.3 Detailed view of the 3 stages reducer. 109
5.4 Gear mesh for each stage. 109
5.5 Results of 3SR problem solved by NSGAII with (blue) and without (red) initial feasible seed. 113

5.6 Explored portion of the domain, showing the 3SR problem complexity. 114
5.7 Flowchart of BnB-NSGAII "Search Feasibility" application. 115
5.8 Number of converged runs 115
5.9 Explored domain by (5.9a) NSGAII, (5.9b) BnB-NSGAII and (5.9c) BnB-NSGAII legacy methods. Feasible and infeasible individuals are plotted in green and red respectively. 116
5.10 Number of successful runs of each CHT on 3SR problem 117
5.11 Results of the 6 CHTs on the mono-objective version of the 3SR problem. 118
5.12 Distribution of the quality metric over the 10 runs. 119
5.13 Best to Best comparison between the Pareto fronts obtained by BnB-NSGAII (red) and NSGAII (blue). 120

6.1 Example that show the need for new fathoming rule. 125

A.1 Illustration of gear problem 166
A.2 Illustration of bearing problem 168
A.3 Illustration of coupling problem 171
A.4 Illustration of truss problem 172

List of Tables

4.1 Test problems properties 76
4.2 Parameters used for NSGAII and BnB-NSGAII algorithms 79
4.3 Selected parameters for NSGAII 80
4.4 Selected parameters for BnB-NSGAII 88
4.5 Mean values of GD, relative spread (dΔ), number of solutions ($\mid S \mid$) and number of evaluations in addition to IR for the selected combinations ID. (Better results are **emphasized**) 89
4.6 Benchmark problems sizes 92
4.7 Properties of G-Problems set. 97
4.8 Properties of RW-Problems set. 98
4.9 Parameters used in genetic algorithm. 99

5.1 Parameters used for NSGAII algorithm 113
5.2 Anchor points of the best Pareto front obtained by each solver for the 3SR problem. 121

B.1 List of the 20 variables of the optimization problem 176
B.2 Input data for the 3SR problem used in this thesis. 177
B.3 Gears Data. 178
B.4 List of the 41 standard values of normal module in [mm], $s = \{1, 2, 3\}$ 179

C.1 Population size, allowed generations and maximum stall generations parameters for BnB-NSGAII. 181
C.1 Population size, allowed generations and maximum stall generations parameters for BnB-NSGAII. 182
C.1 Population size, allowed generations and maximum stall generations parameters for BnB-NSGAII. 183
C.2 Population size, allowed generations and maximum stall generations parameters for NSGAII. 184

C.2 Population size, allowed generations and maximum stall generations parameters for NSGAII. 185

Part I

English version

Chapter 1

General Introduction

Contents

1.1 Introduction 4
1.2 Thesis Organization 7

1.1 Introduction

Optimization problems arising from the mechanical and materials engineering field are often difficult to solve. This is due to the characteristics these optimization problems have. For instance, mechanical optimization problems are usually nonlinear and sometimes non-convex. They are characterized generally by several conflicting objectives, and by a set of constraints that take the attention of an optimization algorithm away from optimization into focusing instead on feasibility.

Another feature that set these problems apart from other optimization problems, is the presence of both continuous and discrete optimization variables. The optimization problem is then called a mixed variables problem. Mixed variable may make the solution space disjoint, and thus the optimization problem more difficult to solve. A common way to simplify such problems is to relax the integer constraint, meaning that to consider the discrete variables as continuous ones. Then, the problem is solved as a continuous problem with the hope that the solution obtained may be easily repaired to become a good solution to the original problem. The relaxation is particularly easy if the discrete variables are numerical. The situation is different in the case of categorical discrete variables. An example of such problem is the choice of material to be used for manufacturing certain parts. Hence, relaxing the mixed variable problem cannot be always applicable.

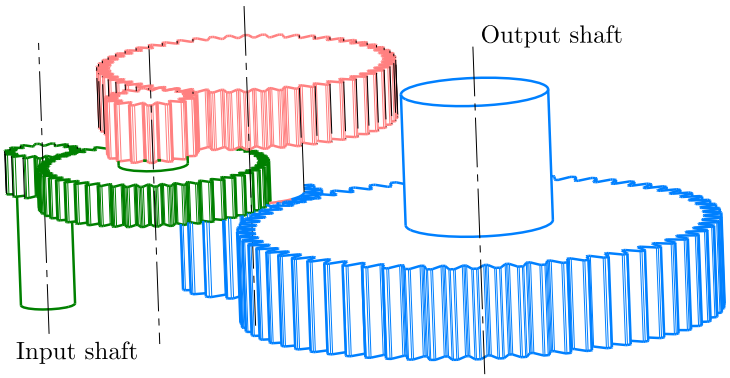


Figure 1.1: A 3-D view of the 3 stages reducer.

Several methods are proposed in literature to solve such problems, i.e. multi-objective mixed variables non-linear problems. These methods could be classified into exact, approximate or even hybrid methods. Exact methods guarantee to find the real optimum of the problem, however, the computation cost increases prohibitively when the combinatorial space is large. Mechanical engineering optimization problems are usually mid-sized problems, they are hand-formulated which may contain tens of variables (usually not more) and few objective functions. Figures 1.2 and 1.1 show a 3 stages reducer problem as an example of mechanical design optimization

problem. In this example, the design problem consists in minimizing the volume and the gear ratio tolerance of the gear reducer instantaneously. The problem is formulated with 2 objective functions, 41 constraints, 3 categorical variables (gears modules) which are indexed as integers that cannot be relaxed, 6 integer variables (number of teeth), and 11 continuous variables. Gears modules have 41 possibilities, pinion number of teeth ranges from 14 to 30 and wheel number of teeth ranges from 14 to 150. Hence, the combinatorial space of this problem consists in $41^3 * (30-13)^3 * (150-13)^3 \simeq 8.7 \times 10^{14}$. Thus, although the problem is considered a mid-sized problem concerning the number of variables and constraints, it consists in a huge combinatorial space. For this reason, exact methods are not commonly used to solve such problems.

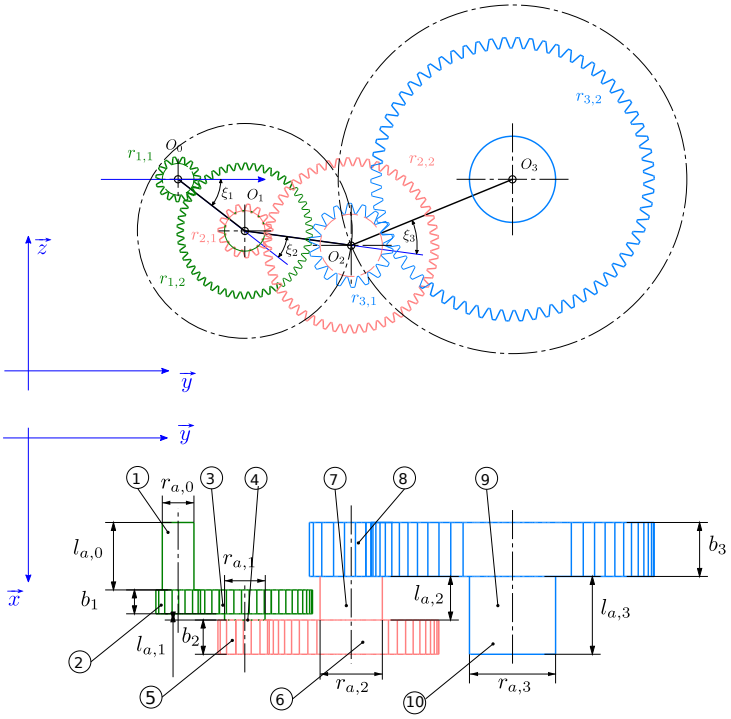


Figure 1.2: Example of mechanical design problem: 3 stages reducer.

Approximate methods are stochastic in nature, they can solve complex optimization problems resulting in - without any guarantee - a good quality solution(s) in a relatively short time. Approximate methods, in particular metaheuristics, are popular in the field of mechanical optimization. However, the literature - as will be shown in the literature review in the next chapter - does not provide a clear conclusion of which metaheuristic algorithm is the best for such problems. It will be shown that although metaheuristics are a form of optimization that is proven to be suitable and robust in many industrial applications from robotics to engineering design optimization, there is no guarantee that these algorithms will fair any better in real life problems.

In fact, a benchmark is needed to measure the performance of the metaheuristic since there is no theoretical proof about the convergence speed nor about the global convergence (i.e a proof of the property of on algorithm to converge toward a solution that satisfy an optimality condition for any initial solution). Thus, no single metaheuristics is the ultimate best for all kind of optimization problems. Instead, an algorithm can be considered more efficient only for *a set of test problems*. Therefore, the only way of testing and evaluating the efficiency and the performance of an algorithm is based on the numerical experiment on a given benchmark problem. Hence, the results are "benchmark" dependent, accordingly, when applying this algorithm on a "real-world" problem, the algorithm is assumed to have the same performance since the "real world" problem is in the same class of - or very near to - the benchmark problem. In this context, El Samrout [2019] has studied the performance of various popular metaheuristics on different multi-objective mixed-integer non-linear real-world mechanical engineering problems. The author concluded that the results obtained by these metaheuristics on these problems were unsatisfying. The author reads into "the need to propose new and adequate optimization algorithms to these problems seems more pressing than ever". Hence, the author recommends to hybridize the metaheuristics with other algorithms to enhance their performance on this kind of problems.

In fact, hybrid metaheuristics has become a very active field, where researchers are more focused nowadays on finding the most adequate hybrid algorithm to the problem at hand. Hybrid metaheuristics are a combination of a metaheuristic with another (or several others) optimization algorithm(s). These optimization algorithms can be stochastic (like metaheuristics) or exact methods. Hybrids are designed based on whether the goal is to obtain faster algorithms or more accurate ones. In any case, hybrids - ideally - will always try to make the most of its components while minimizing their disadvantages. Blum *et al.* [2008] said that "the main motivation behind the hybridization of different algorithms is to exploit the complementary character of different optimization strategies". El Samrout [2019] said that "the huge number of efficient hybrids proves that they are actually the most efficient algorithms for many classical and real-life difficult problems".

Despite their great practicality in real-world mono-objective optimization problems, our literature review will show that only few works has been done on hybrids to solve Multi-Objective Mixed-Integer Non-linear Programming (MO-MINLP) problems that appears in several fields especially in mechanical applications. This thesis aims to develop a new method to solve such problems. Our proposal is a hybridization of the Multi-Criteria Branch-and-Bound (MCBB) algorithm with the Non-dominated Sorting Genetic Algorithm 2 (NSGAII), we called BnB-NSGAII. The proposed approach is furthermore enhanced by new branching strategies designed for MCBB. The constraints are handled using a new proposed constraint handling technique

for evolutionary algorithms that is based on the Non-dominated Sorting of Constraints Violation (NSCV). Numerical experiments based on statistical assessment are done in this thesis to examine the performance of the new proposed approach.

1.2 Thesis Organization

This thesis is organized as shown in Figure 1.3. The literature review on the state of art methods to solve mix variable problems is presented in Chapter 2. In this chapter, our main goal is to provide enough descriptive information for the newcomers in this area about the research that has been done and that is currently under way. The methods are classified in three groups: exact, approximate and hybrid methods. For each group, the state of art methods are addressed showing the advantages and disadvantages for each type of methods.

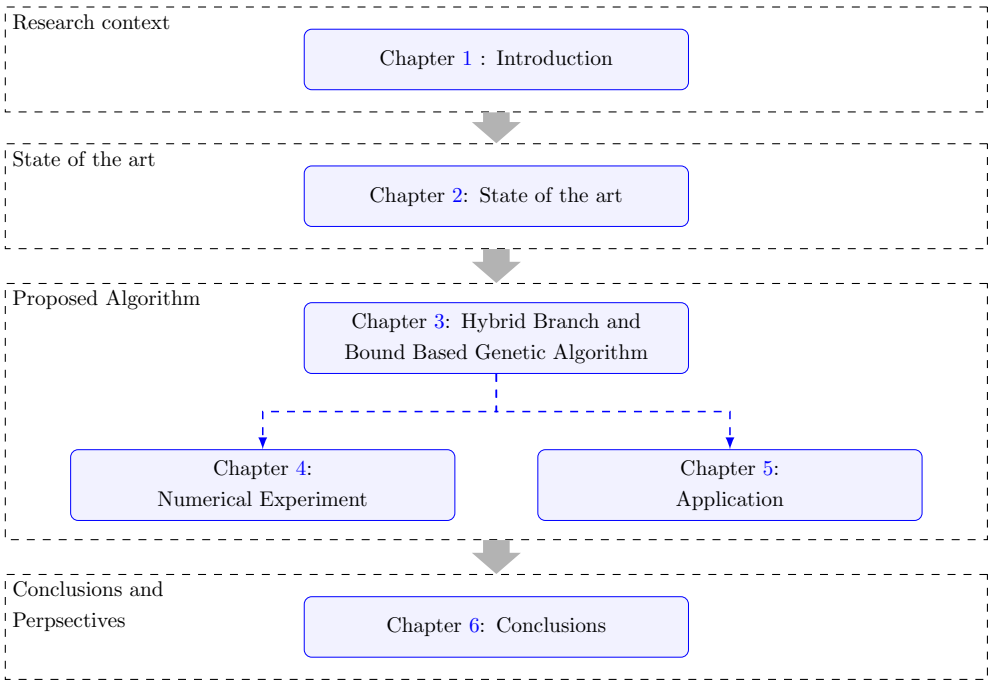


Figure 1.3: Thesis Outline.

In Chapter 3, our proposed algorithm is introduced explaining each portion of it. First, BnB-NSGAII - the general framework of our algorithm - is explained. Then, the legacy feature is demonstrated as an enhancement feature for BnB-NSGAII. After that, our proposed branching strategies are presented. Finally, CHTs are reviewed then our proposed CHT is introduced.

Each proposed feature is tested independently in Chapter 4 showing the impact of each feature on the solution. For each feature, an independent experiment is designed having its own evaluation metrics and benchmark scenario.

In Chapter 5, the 3 stages reducer (3SR) problem is introduced showing its complexity. Then, BnB-NSGAII is used to search for feasible individuals to initialize NSGAII with them. This application of BnB-NSGAII is called "Search Feasibility" application. After that, BnB-NSGAII with the proposed branching strategy and NSCV as CHT is tested on the 3SR problem to examine the overall performance of it. This application is called "Seek Optimality" application.

Finally, an overall conclusion is drawn in Chapter 6 summarizing the key ideas in this thesis and showing the conclusions and perspectives for future work.

Chapter 2

State of Art

Contents

2.1 Overview 10
2.1.1 Taxonomy 10
2.1.2 Scope 16
2.2 Exact Methods for MO-MINLPs 17
2.2.1 Multi-Objective 18
2.2.2 Mono-Objective MINLP 20
2.2.3 Drawbacks and Conclusions 24
2.3 Approximate Methods 26
2.3.1 Multi-Objective Metaheuristics 27
2.3.2 Evolutionary Algorithms 29
2.3.3 Drawbacks and Conclusions 31
2.4 Hybrid Methods 32
2.4.1 Hybrids Classifications 33
2.4.2 Combining Approximate Methods with Tree Search (A + E2.1) 36
2.5 Conclusion and Propositions 38

2.1 Overview

Applications of optimization are countless. Every process has a potential to be optimized. In fact, Talbi [2009] said that "there is no company that is not involved in solving optimization problems". Breitkopf & Coelho [2013] said that "any system design is an optimization process". Indeed, many challenging applications in science and industry can be formulated as optimization problems. Optimization occurs in the minimization of time, cost, and risk or the maximization of profit, quality, and efficiency.

2.1.1 Taxonomy

Optimization methods could be classified based on the type of the problems that they attend to solve, or they could be classified based on the type of the algorithm. In this section, we present a taxonomy that addresses the classifications of optimization problems based on their characteristics, the classification of optimization methods and the classification of transformation techniques of the optimization problems.

2.1.1.1 Exact Methods (E)

Exact methods are deterministic algorithms that converge towards a solution which satisfy some optimality conditions. These algorithms have been successfully applied to many engineering design problems [Rao & Savsani, 2012]. Exact methods are either enumerative or iterative methods (Figure 2.1). Iterative methods are sequence of solution displacements or mathematical procedures that use an initial value to generate a sequence of improving solutions for a class of problems, in which the n-th solution is derived from the previous ones. The cutting-plane method iteratively refines a feasible set by means of linear inequalities, termed cuts. It is commonly used to solve mixed-integer linear programming (MILP) problems, as well as to solve general, not necessarily differentiable, optimization problems. The use of cutting planes to solve MILP was introduced by Gomory [1960].

Enumerative methods consists in enumerating the domain space explicitly (all the solutions in the domain) or implicitly as in the branch and bound algorithm. Figure 2.2 shows an example of how the branch and bound implicitly enumerates the domain space. Branch and bound is a well-known generic method for computing an optimal solution of a single objective optimization problem. Based on the "divide to conquer" idea, it consists in an implicit enumeration principle, viewed as a tree search. The feasible set of the problem is iteratively partitioned to form subproblems of the original one. Generally in branch and bound, the main problem is divided to subproblems (nodes). Each subproblem is evaluated to obtain a lower bound on the subproblem objective value. The lower bounds on subproblem objective values are used to construct a proof of optimality without exhaustive search. Uninteresting and infeasible subproblems are

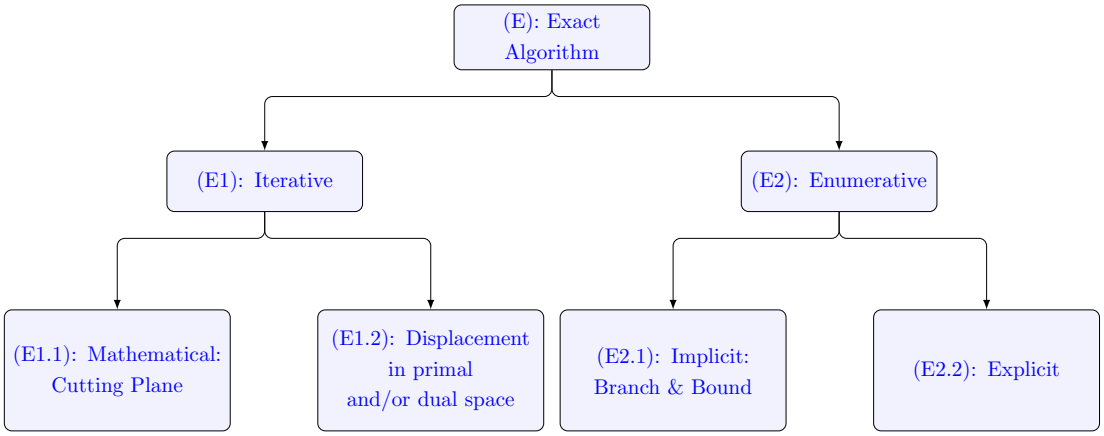


Figure 2.1: Taxonomy of exact algorithms.

pruned (fathomed) as the red subproblem in Figure 2.2, promising subproblems are selected and instantiated as the blue ones.

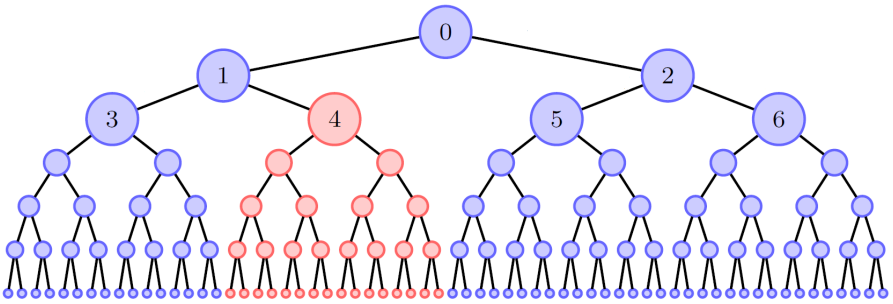


Figure 2.2: Implicit enumeration of the domain space by branch and bound algorithm.

Branch and bound is considered exact method as long as (1) the branching condition - the union of the subsets remains equal to the complete set - (2) the bounding condition - the lower bound is proved to be lower than the global optimal solution - are attained. Otherwise, branch and bound cannot guarantee to provide the optimal solution(s). If the branching condition is not verified, then there might be a part of the feasible domain, that may contain the global optimum solution(s), is discarded. Consequently, the solver will converge to a solution that cannot be verified as a global optimum or even a local one, since the true global optimum is lost. On the bounding side, any fathoming rule depends on the obtained lower bound of the node. If the obtained lower bound cannot be proven that it is lower than the global minimum, the node that contains the optimal solution(s) can be mistakenly fathomed.

Generally, exact methods guarantee to find the real optimum of the problem, however, the computation cost increases prohibitively when the combinatorial space is large.

2.1.1.2 Approximate Methods (A)

On the other hand, approximate methods are stochastic in nature with a probabilistic transition rules, these techniques were basically developed to overcome the shortcomings of exact methods, as they can solve complex optimization problems resulting a good quality solutions in a relatively short time. However, they cannot guarantee to find an optimum solution nor a local optimal one.

Approximate methods could be generally heuristics or metaheuristics as shown in Figure 2.3. Metaheuristics represent a family of approximate optimization techniques that gained a lot of popularity in the past two decades. They are among the most promising and successful techniques. Metaheuristics provide "acceptable" solutions in a reasonable time for solving hard and complex problems in science and engineering. This explains the significant growth of interest in metaheuristic domain. Unlike exact optimization algorithms, metaheuristics do not guarantee the optimality of the obtained solutions. They do not define how close are the obtained solutions from the optimal ones.

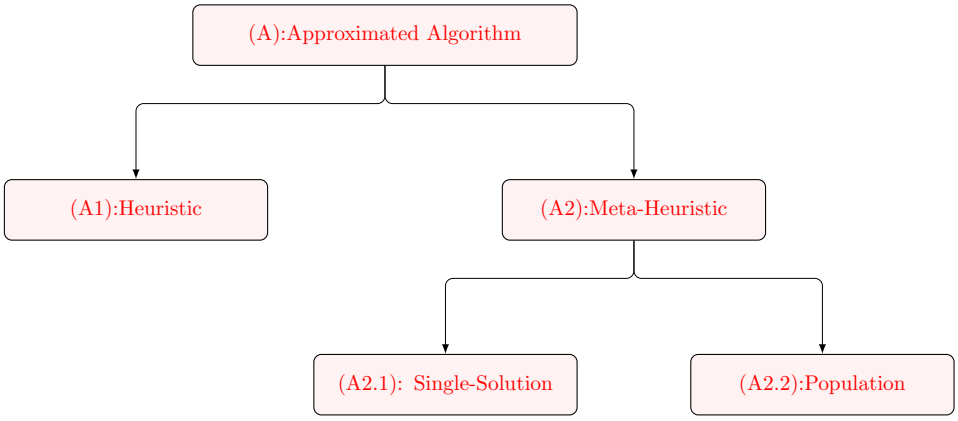


Figure 2.3: Taxonomy of approximate algorithms.

Metaheuristics could be classified into: (A2.1) single-solution based algorithms (e.g., local search, simulated annealing) manipulate and transform a single solution during the search; (A2.2) population-based algorithms (e.g., particle swarm, evolutionary algorithms) a whole population of solutions is evolved. Population-based metaheuristics are exploration oriented; they allow a better diversification in the whole search space, moreover, they are more suited to multi-objective applications [Talbi, 2009].

2.1.1.3 Problems Characteristics (P)

In another manner, optimization methods are classified based on the problem type, mainly combinatorial or continuous problems, constrained or unconstrained, linear or non-linear and mono

or multi-objective. Figure 2.4 shows the taxonomy of optimization problems based on their characteristics. Optimization problems could be classified into two categories: mono-objective having one objective function and multi-objective that have more than one contradictive objective function simultaneously that have a set of solutions called Pareto Front. Many practically important problems can be described by a multi-criteria optimization problem [Ehrgott *et al.*, 2002]. Optimization problems could be subjected to some constraints. A problem is said to be non-linear (or non-convex) if one of the objective(s) or constraints functions is non-linear (non-convex).

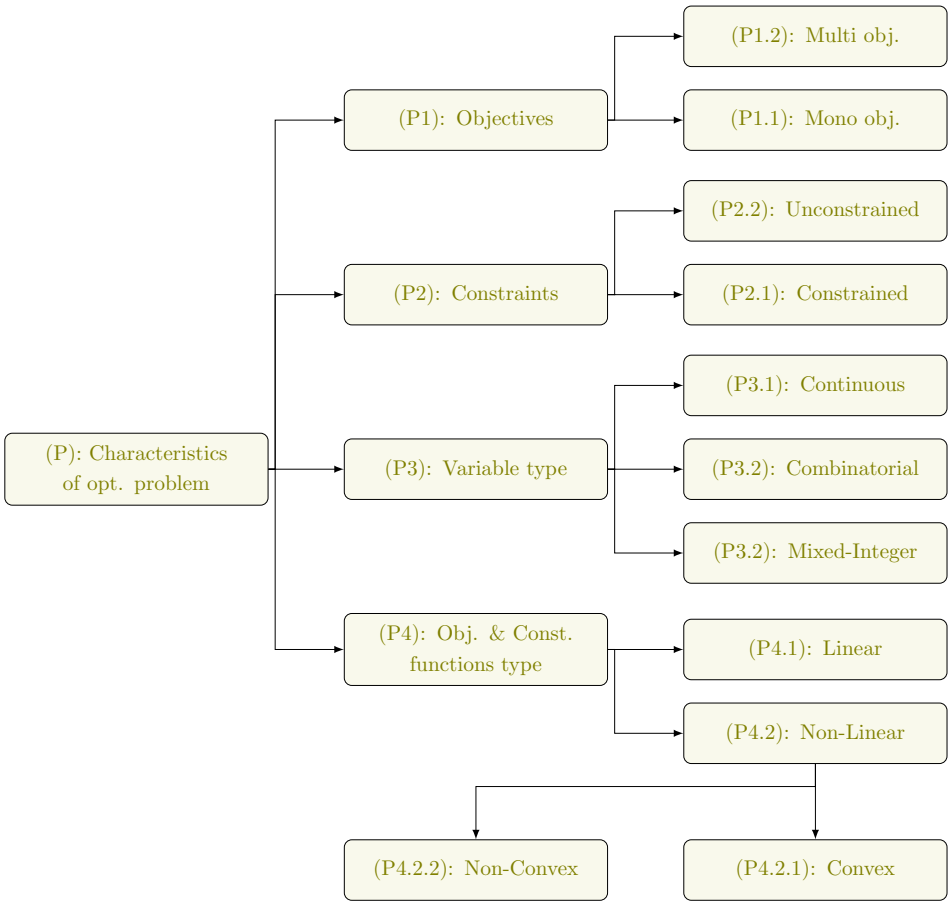


Figure 2.4: Taxonomy of optimization problems based on their characteristics.

Combinatorial optimization is a field extensively studied by many researchers [Ehrgott & Gandibleux, 2000] which consists in solving optimization problems with only discrete variables. Due to its potential for application in real world problems, it has prospered over the last few decades. Combinatorial optimization problems (P3.2) are characterized by a *finite set of possible solutions*. Such problems occur in almost all fields of management as well as in many engineering

disciplines [Socha, 2009]. The combinatorial optimization model's adaptability arises from the fact that many practical problems include indivisible activity and resources, such as machinery, airplanes, and people. Furthermore, many problems have a finite number of possible options and, as a result, can be stated as combinatorial optimization problems. It can be challenging to solve combinatorial optimization problems. The difficulty derives from the fact that, unlike linear programming, which has a convex feasible region, combinatorial problems require the search of a lattice of feasible solutions. Unlike linear programming, where we can take advantage of the fact that any locally optimal solution is a global optimum due to the problem's convexity, combinatorial optimization problems have many local optima, and finding a global optimum to the problem necessitates enumerating all feasible solutions. The only way to solve this kind of problems exactly is by enumerating all possible solutions (E2) which is among the bounds of possibility for large combinatorial space problems. Hence, here is the mean of interest in approximate methods (A).

The main difference between the combinatorial and continuous optimization problems (P3.1), is the fact that *the search space is not finite*. Each of the continuous decision variables may assume an infinite number of values. Of course, solving such problems using computers imposes certain limitations, as computers—being digital in nature—can only represent a finite number of values. One could think that this reduces the problem to a combinatorial optimization one. This is however not really the case. In the case of combinatorial optimization, the set of available values is predefined before starting the optimization. The size of this set has a significant influence on the difficulty of finding an optimal or near-optimal solution. The area of applications for continuous optimization is very wide. Many real-world problems and processes may be presented in the form of a continuous optimization problem. If the optimality can not be proved by calculus-based derivative approaches, exact methods solve this type of problems iteratively using an iterative step in the search space. The quality of the solution is highly influenced by the size of this step which is a major concern in such methods.

Discrete decisions and non-linear system dynamics affect the quality of the final design or plan in many optimal decision problems in scientific, engineering, and public sector applications. Mixed-integer difficulties arise as a result of these decision problems. (P3.3) that are hence a combination of combinatorial and continuous optimization problems. They are particularly difficult to tackle, as they pose both type of difficulties—those of combinatorial problems (e.g., the necessity to check essentially all the solutions to be certain that the optimal one has been found), and those of continuous problems (e.g., the fact that the search space is infinite and may be unbounded). For this reason, there are not many dedicated algorithms to tackle mixed-variable problems directly.

Applications of mixed-variable optimization problems concern mostly mechanical applications. Many industrial process and design problems contain parameters of which some are dis-

crete and some continuous. Popular examples include the truss design problem, the coil spring design problem, designing a pressure vessel, welded beam design, or design of thermal insulation systems [Socha, 2009].

2.1.1.4 Transformation Techniques (T)

Many transformation techniques have been proposed to transform the optimization problem to an easier one. The taxonomy of the transformation techniques is illustrated in Figure 2.5. In this section, scalarization and relaxation techniques - T1 and T3 respectively - are highlighted. Some of the linearization techniques (T4) are addressed in Section 2.2. While penalization techniques (T2) are reviewed in Chapter 3 with other constraints handling techniques.

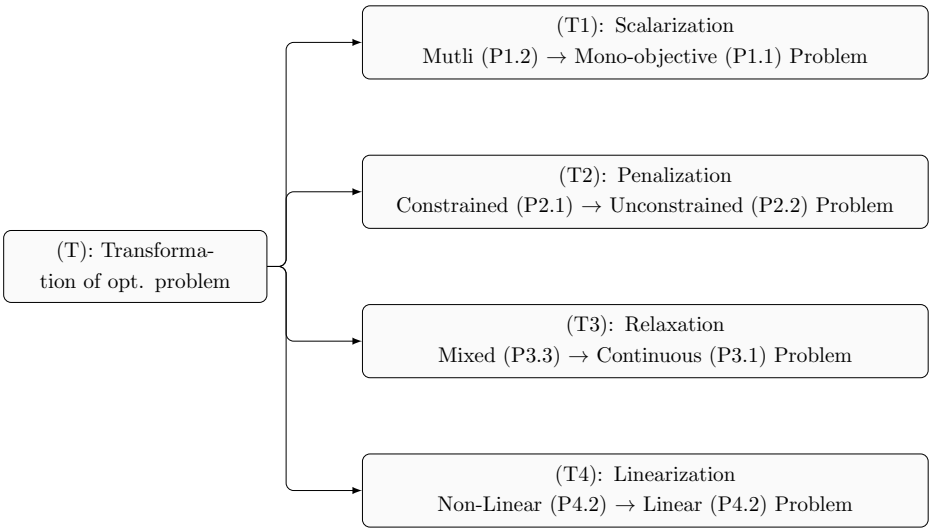


Figure 2.5: Taxonomy of transformation techniques of the optimization problems.

2.1.1.4.1 Scalarization Scalarization of a given multi-objective problem (P1.2) means converting that problem into an optimization problem (or a family of problems) with a mono-objective function (P1.1) to be minimized. This fact is reflected in the usual mathematical modeling of an optimization problem as a *scalar optimization problem*, that is, a minimization of a function over a set specified by some constraints, where the function and/or part of the constraints are derived from the conflicting goals (see Section 2.2.1.1).

2.1.1.4.2 Relaxation Most of the approaches that are in use to solve mixed variables problems relax some of the constraints of the problem and then try to solve the transformed problem with the hope that the solution obtained may be easily repaired to become a good solution to the original problem. The most popular approach is to relax the requirement for the discrete

variables. They are replaced by continuous variables and the problem is solved as a continuous optimization problem, but taking into account during the objective function evaluation only the allowed values for the discrete variables. This type of approach is often called continuous relaxation approach. The relaxation is particularly easy if the discrete variables are numerical, or at least if a certain order among them may be established.

2.1.2 Scope

In the context of optimization in mechanical engineering and more precisely in the context of the design of mechanical systems, possibly including the problems of optimization of mechanical processes, it should be noted that:

- The design problem contains integer variables (P3.3) that can refer either to a number of components or to an input number in a data table, allowing manipulation of "catalogs" of mechanical components. So it is *sometimes impossible to "relax" the integer variables by considering them as continuous.*

- These problems generally involve multi optimization criteria (P1.2) (often 2 or 3 and rarely more), because these criteria - that decision maker seeks to minimize - are related to the performance of the system as, for example, the mass, the rigidity, the cost (when it can be realistically estimated) or etc. Physical models for expressing, mechanical strength, durability, reliability, etc. are normally used to express the constrained functions of the optimization problem (hence P2.2). In this context, the objectives and constraints functions are usually non-linear (P4.2) and not necessary convex (sometimes P4.2.2). Hence, the resulting problem is a Multi-Objective Mixed-Integer Non-Linear Programming (MO-MINLP) problem. For some examples and an explanation of how to formulate an optimization problem from an optimization problem, see Pomrehn & Papalambros [1995] and Simpson & Martins [2011]. For more examples about these problems, see Messac [2015] or Deb [2001].

- Another particularity to this type of optimization problems is that they have at least a mid-sized combinatorial space (couple of integer variables accepting less than a hundred values). Therefore, the exact solution can not be computed using a simple explicit enumeration procedure.

Mixed variables problems require choosing values for discrete and continuous variables. Some of them may be easily transformed into continuous optimization problems (this is often the approach used to solve them). Some may not be easily transformed in such a way. This is due to the fact that there is no obvious ordering that may be imposed on the categorical variables.

Certainly, relaxation of the mixed-variable problem to a continuous problem is not the only possible approach, and there exist also methods performing full mixed-variable optimization. Such methods are referred to as Mixed-Integer Non-Linear Programming (MINLP) native methods.

To demonstrate the need to have a native MINLP optimization method for solving engineering design problems, let us consider a simple cantilever beam design problem. A cantilever beam is a beam that is rigidly supported at one end and carries a certain load on the other end. In trying to design such a beam designers face a number of possibilities. For simplicity the beam can be assumed to have a circular cross-section or a square cross-section. Thus the shape of the beam constitute a binary zero/one design variable. Once the shape is determined the next choice is the size of the shape. If the chosen shape is circular the diameter of the cross-section is a variable. However, if the chosen shape is a square the side of the square cross-section is a design variable. Since circular or square cross-sections are not available in any arbitrary sizes this variable is a discrete variable taking a few permissible values. The length of the beam could be another design variable. Since the beam can be cut to a large precision, we may consider this variable as a continuous variable taking any positive real value. These three design variables fix the size and shape of the beam to their permissible values. Another variable which is very difficult to include in any traditional optimization method is the material of the beam. This variable is again a discrete variable taking only a few chosen materials. One distinct difference between this variable and the second discrete variable representing the size of the cross-section is that no value other than the permissible values of this variable is strictly allowed during the optimization process. For example, if the material variable is represented as steel or cast iron, then no other intermediate value is allowed. Ideally a designer is interested in an optimization algorithm which has the flexibility to handle such a mixed type of variables.

In this chapter we intend to give an overview over the literature in the field of MO-MINLP optimization methods. First, the exact methods are reviewed showing how the exact multi-objective methods converts MO-MINLP to a mono-objective MINLP. Hence, we extend the survey of the literature to cover the mono-objective methods for the newcomers to this area showing the general concepts for solving such problems. Then, the approximate methods are generally reviewed focusing on multi-objective evolutionary algorithms. Finally, different approaches of hybrid methods are presented.

2.2 Exact Methods for MO-MINLPs

The main concept of a multi-objective exact method is to convert the multi-objective problem to a mono-objective one. Each method has its own mechanism to make this conversion. In this section, we show the different types of these mechanisms. Since the problem is converted to a

mono-objective one, we generally address the mono-objective methods from literature showing how these problems are solved.

2.2.1 Multi-Objective

The main methodologies for the exact solution of multi-objective (P1.2) non-linear (P4.2) problems with mixed-integer variables (P3.3) are the scalarization methods (T1) and multi-criteria branch and bound (E2.1) [Przybylski & Gandibleux, 2017].

2.2.1.1 Scalarization Methods (T1)

The most commonly used technique to find efficient solutions — employed in most exact methods and many heuristic techniques — is scalarization [Ehrgott *et al.*, 2002]. Scalarization methods are not dedicated to MO-MINLPs only, in fact they are general techniques that transform the multi-objective problem to a single objective one. Burachik *et al.* [2021] have mentioned five scalarization methods that are namely used to solve MO-MINLP problems. In particular, they transform MO-MINLP into a scalarized MINLP. The scalarized problem is a single objective problem related to MO-MINLP with additional variables and/or parameters, that is usually solved repeatedly in order to find some subset of efficient solutions of MO-MINLP. In this section we summarize some of the common scalarization techniques that have been proposed in the literature.

2.2.1.1.1 The weighted sum method The scalarization of the weighted sum method is a convex combination of the p objectives of MO-MINLP, where the feasible set is unchanged:

$$\min_{x \in X} \sum_{k=1}^{p} w_k f_k(x), \tag{2.1}$$

where w_k is the weight of each objective function $f_k(x)$. The most important positive property of the weighted sum method is that it has the same computational complexity and needs the same computational effort to solve as the single objective version of MO-MINLP [Ehrgott, 2006].

2.2.1.1.2 The ϵ-constraint method The contender for most popular method to find all efficient solutions is the ϵ-constraint method. In the ϵ-constraint method, one of the p objectives (say the j -th) is retained for minimization and the other $p-1$ are turned into constraints:

$$\begin{array}{ll} \underset{x \in X}{\text{minimize}} & f_j(x) \\ \text{subject to} & f_k \leq \epsilon_k,\ k \neq j. \end{array} \tag{2.2}$$

All efficient solutions can be found by appropriately specifying the right hand side values. However, the computational effort is strongly problem dependent [Ehrgott, 2006].

2.2.1.1.3 Pascoletti–Serafini scalarization This approach is introduced by Pascoletti & Serafini [1984] and is also referred to as the goal-attainment method. Given a fixed $w \in W$, the Pascoletti–Serafini (PS) scalarization is posed as the following problem.

$$\begin{array}{ll} \underset{(\alpha, x) \in \mathcal{R} \times X}{\text{minimize}} & \alpha \\ \text{subject to} & w_i(f_i(x) - u_i) \leq \alpha, \forall i = 1, \ldots, l, i \end{array} \tag{2.3}$$

where $\alpha \in \mathcal{R}$ is a new variable and $u = (u_1, \ldots, u_l)$ is a utopia vector. By varying the parameters of the scalar problem, it is possible to find all vector optima from the scalar ones.

2.2.1.2 Multi-Criteria Branch and Bound (E2.1)

Although the branch and bound was first suggested in (1960) [Land & Doig, 1960], the first complete algorithm introduced as a multi-objective branch and bound, that was identified in [Przybylski & Gandibleux, 2017], was proposed in (1983) [Kiziltan & Yucaoğlu, 1983]. The first branch-and-bound algorithm for multi-objective mixed binary linear problems MO-MILP was proposed later in (1998) [Mavrotas & Diakoulaki, 1998]. Multi-Criteria Branch and Bound (MCBB) proposed in [Mavrotas & Diakoulaki, 1998] is a vector optimization approach capable of producing all the efficient extreme solutions of Mixed 0-1 multi-objective linear problems. For this purpose, the conventional branch and bound algorithm was properly modified, to find all the non-dominated solutions of multiple objective problems, after the exhaustive examination of all possible combinations of the binary variables.

MCBB was successfully implemented to solve mixed integer linear problems in Engineering, Operational Research [Cacchiani & D'Ambrosio, 2017] and Chemical [Boix *et al.*, 2010] fields. Moreover, MCBB was used to develop a heuristic method to solve convex MO-MINLP (E2.1 + A1 to solve P1.2+P3.3+P4.2.1) in [Cacchiani & D'Ambrosio, 2017]. Later in [De Santis *et al.*, 2020], the authors proposed the first deterministic MCBB based method to solve convex MINLPs.

2.2.1.2.1 Branching Branching is a separation procedure that splits a problem domain into subsets while ensuring that the union of the subsets remains equal to the complete set. Any branching strategy acts on the integer variables bounds, thus, two types of sub-spaces of combinations might result:

- Those for which all integer variables are fixed, thus a leaf is obtained, so it is associated with a problem of the type multi-objective continuous problems.
- Those for which contain free integer variables are associated with a problem of the type mono-objective mixed-integer problems to obtain their bounds.

2.2.1.2.2 Bounding and fathoming Bounding (solving) a node means to evaluate its the upper and lower bounds. For a single-objective optimization problem, there is at most one optimal value and the definition of upper and lower bound on this value is therefore natural. For a multi-objective optimization problem, upper bounds are considered on the set of non-dominated points. A well-known lower bound is the ideal point of the Pareto front. Integer relaxation is also popular technique to obtain the lower bound. In MCBB, the node is repeatedly solved for each objective function independently. Hence, a mono-objective solver is called repeatedly to obtain an optimal solution for each objective function.

Fathoming is extended in MCBB to deal with multiple objectives: a node can be fathomed if its lower bound vector is dominated by (at least) one of the non-dominated points of the current incumbent set, or if the node is infeasible.

2.2.1.2.3 Discussion In MCBB, there are several components which are left unspecified, but which can have significant impacts on the performance of the algorithm [Morrison *et al.*, 2016]. These components are illustrated in Figure 2.6: the separation order (i.e., the order of integer variables to be branched in which subproblems in the tree are explored), the branching strategy (i.e., how the solution space is partitioned to produce new subproblems in the tree), active node selection strategy such as "deep first" or "breadth first" methods, and fathoming rules (i.e., rules that prevent exploration of suboptimal regions of the tree). Belotti *et al.* [2009]; Przybylski & Gandibleux [2017] discuss the impact of "separation order" and "choosing the active node" on the performance of MCBB. Moreover, the authors review the existing strategies and differences between them. Przybylski & Gandibleux [2017] and Cacchiani & D'Ambrosio [2017] review distinct pruning rules used in MCBB. Morrison *et al.* [2016] mention two general frameworks of "branching strategies", however, these frameworks are dedicated for the single objective branch and bound. To the best of our knowledge, there is a lack in the literature concerning the branching strategies for MCBB.

2.2.2 Mono-Objective MINLP

All exact multi-objective methods converts the multi-objective problem into mono-objective (P1.1 + P3.3) one. So in this section the mono-objective solvers are briefly described to provide enough descriptive information for the newcomers in this area. In this context, we deal with non-linear problems, hence, linear methods are excluded. A non-linear problem might be a convex or a non-convex depending on the type of the objective and constraint functions. In this section we categorize the methods based on the type of the problems that they are designed to solve.

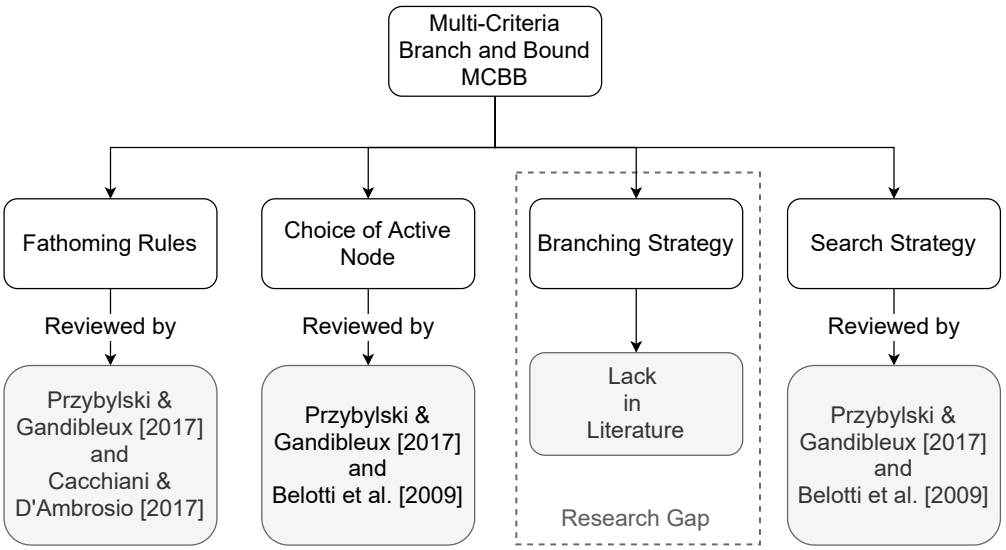


Figure 2.6: MCBB components review chart.

2.2.2.1 Convex Problems (P4.2.1)

Convex MINLP is often considered as a "difficult" class of optimization problems. However, tremendous progress has been made in the field over the last two decades, and there are now numerous good MINLP solvers available [Kronqvist *et al.*, 2019]. This section aims to provide an overview of the most widely used deterministic solvers for convex MINLP problems. This summary is not meant to be a comprehensive examination of the algorithms, but rather to highlight the distinctions between the solvers.

2.2.2.1.1 Extended Cutting Plane (E1.1 + T3 + T4) MILP cutting plane methods solve a non-integer linear problem, which is the linear relaxation of the given integer program. Under modest assumptions (assuming the linear program has an optimal solution and the feasible region does not contain a line), the theory of Linear Programming (LP) states that an optimal extreme point or corner point can always be found. The optimum is checked to see if it is an integer solution. If it isn't, there will always be a linear inequality separating the optimum from the true feasible set's convex hull. The separation problem is finding such an inequality, and such an inequality is a cut. To the relaxed linear program, a cut can be added. The relaxation is then no longer possible with the current non-integer answer. This procedure is done until the best integer solution is discovered. The extended cutting plane (ECP) algorithm was first presented by Westerlund & Pettersson [1995]. In its original form the ECP method is intended for convex MINLP problems by linearizing the nonlinear constraints to construct an iteratively improving polyhedral approximation constructed by the first-order Taylor series expansions.

2.2.2.1.2 Extended supporting hyperplane (E1.1 + T3 + T4) The extended supporting hyperplane (ESH) algorithm was presented by Kronqvist *et al.* [2016] as an algorithm for solving convex MINLP problems. The ESH method obtains trial solutions using the same technique as the ECP algorithm, but generates the polyhedral approximation using a different technique. ESH obtains supporting hyperplanes by using a one dimensional root search (see [Kronqvist *et al.*, 2019]) with LP relaxation as a preprocessing step.

2.2.2.1.3 Outer approximation (E1.1 + E2.1 + T3 + T4) The outer approximation (OA) method was first presented by Duran & Grossmann [1986], with additional properties for convex MINLP problems described in [Fletcher & Leyffer, 1994]. To handle non-convex problems more efficiently, some changes to the OA approach have been presented. OA is a decomposition approach that solves a series of mixed-integer linear programming (MOLP) and nonlinear programming (NLP) subproblems to find the optimal solution to the original problem. OA, like ECP and ESH, builds a polyhedral outer approximation of the nonlinear feasible region that improves iteratively. OA, on the other hand, simply uses the polyhedral approximation to choose integer combinations, while the continuous variables are chosen by solving a convex NLP subproblem. The polyhedral outer approximation is utilized to construct the problem in each iteration, resulting in a new integer combination. After obtaining the integer combination, the problem is solved as NLP. If the problem (NLP-fixed) is feasible, the solution yields a valid upper bound, which is then used to improve the polyhedral approximation. The polyhedral outer approximation is updated by either linearizing all constraints or only the active constraints.

2.2.2.1.4 Solver enhancement techniques Most solvers don't use a single approach to increase their performance; instead, they mix various techniques. In this section, we briefly describe several pre-processing approaches and fundamental heuristics that can be used in conjunction with the other methods to increase practical performance. Pre-processing entails a variety of strategies for transforming the problem into a format that is more suitable to the solver. The pre-processing procedures can result in tighter relaxations or reduce the problem size. Belotti *et al.* [2013] classified MINLP pre-solving techniques into two major categories: housekeeping and reformulations.Reformulations can include strategies such as improving coefficients in constraints and disaggregation of constraints, while housekeeping can include approaches such as bound tightening and removal of unnecessary constraints. Primal heuristics, on the other hand, is a general term for algorithms and techniques that aim to find good feasible solutions with minimal computational work when compared to solving the original problem. Primal heuristics were first used in the field of MILP, with [Fischetti & Lodi, 2011] claiming that primal heuristics were one of the most significant improvements in MILP solvers in the recent decade. In recent years, there has been a surge of interest in primal heuristics for MINLP problems, with several

algorithms suggested. Such algorithms can be found in [Kronqvist *et al.*, 2019], e.g. undercover [Berthold & Gleixner, 2014], feasibility pumps [Bonami *et al.*, 2009], rounding heuristics [Berthold, 2014], and etc.

2.2.2.2 Non-Convex Problems (P4.2.2)

Several simple non-convex MINLP examples are known to be not just NP-hard, but also undecidable, such as the situation where all functions are quadratic, all variables are integer bound, and the number of variables is fixed. [Burer & Letchford, 2012]. All of the proofs that non-convex MINLPs can be undecidable, it turns out, involve cases with an unbounded feasible domain. Fortunately, the feasible domain is typically bounded, either explicitly or implicitly, in practice. Nonetheless, some relatively small non-convex MINLPs, with only a few tens of variables, can lead present approaches to run into serious difficulties. Many important practical problems are naturally modeled as non-convex MINLPs. Several methods were reviewed in [Burer & Letchford, 2012] among them are the following.

2.2.2.2.1 Spatial branch-and-bound (E2.1 + T3) Whether standard or spatial branching, it is commonly implemented recursively, resulting in a hierarchy of subproblems. These subproblems can be considered as being organized in a tree structure that can be searched in many ways, as the branch-and-bound method. A subproblem can be fathomed or pruned (removed from further consideration) if the associated lower bound is no better than the best upper bound found so far, (ii) it is infeasible, or (iii) it is feasible for the original problem and its cost under the relaxed objective equals its true cost (to within a specified tolerance). The fundamental difference between this strategy and the traditional branch and bound is the third condition. This general technique was first introduced in the context of global optimization issues by McCormick [1976]. Later, other authors realized that the method might be used to solve problems with integer variables as well..

2.2.2.2.2 Branch-and-reduce (E2.1 + T3) The development of the branch-and-reduce technique by Ryoo & Sahinidis [1996] was a major step forward in the exact solution of non-convex MINLPs. This is an upgraded variant of spatial branch-and-bound in which the domains of the variables are reduced in addition to the reductions that occur naturally as a result of branching. To be more explicit, the following two operations are added: (i) before solving a subproblem, its constraints are examined to see if the domain of any variables can be reduced without losing any feasible solutions; (ii) after solving the subproblem, sensitivity information is examined to see if the domain of any variables can be reduced without losing any optimal solutions. After performing domain reduction, better convex under-estimators can be generated. As a result, the constraints can be tightened, resulting in better lower bounds. The overall result

is typically a significant reduction in the size of the enumeration tree. Because of two crucial features, branch-and-reduce is commonly conducted using Linear Programming (LP) relaxations (continuous relaxation) rather than more complex convex programming relaxations. First, LPs can be solved more efficiently and with more numerical stability. Second, in the case of LPs, sensitivity information is more readily available (and easier to interpret).

2.2.2.2.3 α-branch-and-bound (E2.1 + T3) Androulakis *et al.* [1995] proposed an exact spatial branch-and-bound algorithm for global optimization of non-convex NLPs in which *all functions involved are twice differentiable.* This method, termed *alpha*-BB, is based on their generic under-estimator construction technique. In comparison to the more conventional spatial branch-and-bound technique, or even branch-and-reduce, one advantage of *alpha*-BB is that it usually requires no new variables. In other words, it's often possible to work with the original objective and constraint functions without resorting to factorization. This is due to the fact that the under-estimators utilized do not factor functions. However, rather of an LP solver, a generic convex programming solver is required to solve the relaxations.

2.2.2.2.4 Conversion to an MILP (T4) Another technique is to factorize the problem, then approximate the resulting separable MINLP with a MILP and solve the resultant MILP with any available MILP solver. The conversion to a MILP results in sets of binary variables with a certain structure. These sets are referred to as special ordered sets of type 2 by Beale & Tomlin [1969], and they suggest a specific branching algorithm. This branching rule is now used by the majority of commercial and academic MILP solvers.

2.2.3 Drawbacks and Conclusions

As a summary, mixed-variable optimization problems are particularly difficult to tackle, because they combine two type of difficulties. The first is that of combinatorial problems, where it is necessary to check all the solutions to be certain that the optimal one has been found. While the second difficulty is that of continuous problems where the search space is infinite and may be unbounded. Figure 2.7 shows the summary of this section. Exact methods can be categorized into mono-objective or multi-objective methods. Multi-objective methods are enumerative methods that convert the MO-MINLP problem to a single objective one. A method is said to be better when it guarantees to find the optimal solution avoiding enumeration of all the combinatorial space. Then, depending on the type of the problem, i.e. convex or non-convex, the mono-objective methods are classified and called to solve the revealed mono-objective problem(s).

The main drawback of MINLP exact methods is first that the computation cost increases prohibitively when the combinatorial space is large. Another aspect related to exact methods that solve MINLPs is that most of them relay on the concept of integer relaxation, which may be not

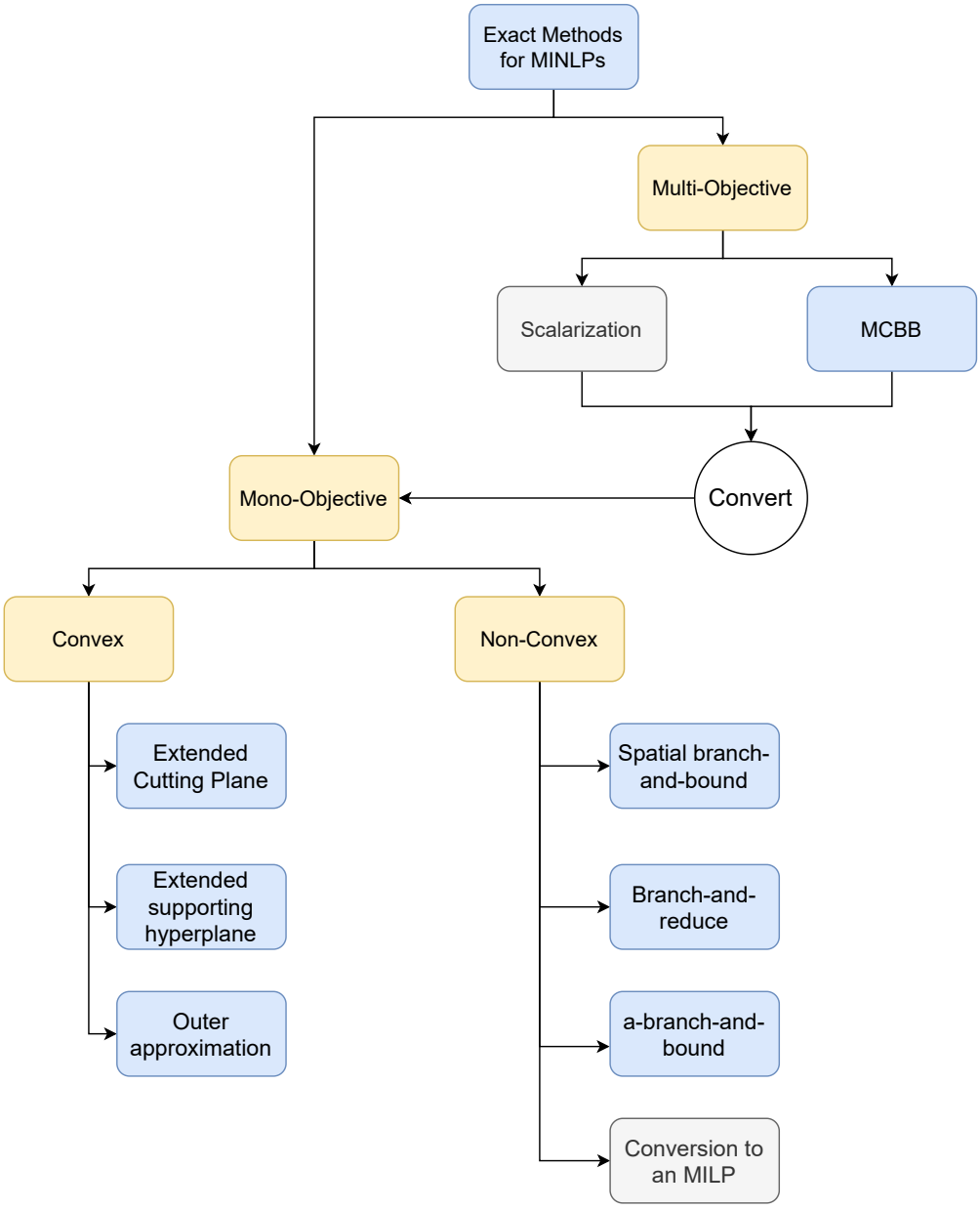


Figure 2.7: Exact methods review chart (*color-coded to meet the presented taxonomy*).

desired in some cases. If the problem cannot be relaxed, the only way to solve it is by enumerating all possible combinations of the integer variables which may lead to the "combinatorial explosion" with the increase of the problem size, only the small combinatorial spaces could be solved by such an approach. Hence, although there exist exact methods for solving mixed-integer optimization problems, they have their disadvantages. They may not be always used either due to the problem

formulation, or to the fact that they would need excessive time to solve the problems. This is why metaheuristics and other approximate methods are enjoying increasing attention.

2.3 Approximate Methods

Unlike exact methods, approximate methods allow to tackle large-size problem instances by delivering satisfactory solutions in a reasonable time. There is no guarantee to find global optimal solutions or even bounded solutions. Due to the computational complexity of hard combinatorial problems, especially in the presence of large scale problem instances, in the last decades there has been an extensive research effort devoted to the development of heuristic algorithms. There is no general framework behind the design of a good heuristic method that for any problem is guaranteed able to find good quality solution. The efficacy of a heuristic method is determined by its capacity to adapt to a specific realization, avoid entrapment at local optima, and exploit the problem's core structure. On the basis of these principles, numerous heuristic search approaches have been created, which have demonstrably enhanced the potential to obtain good solutions to challenging combinatorial optimization problems [Festa, 2014].

Metaheuristics are a type of methods that are widely used to solve computationally hard combinatorial optimization problems. The phrase meta-heuristic is derived from two Greek words: meta and heuriskein. The suffix "meta" signifies "beyond," "on a higher level," and "heuriskein" means "to discover." Indeed, metaheuristics are a class of algorithms that attempt to combine basic heuristic methods in higher-level frameworks with the goal of rapidly investigating the set of feasible solutions to a given combinatorial problem. Metaheuristics have received more and more popularity in the past decades. Their use in many applications shows their efficiency and effectiveness to solve large and complex problems. Application of metaheuristics falls into a large number of areas [Talbi, 2009]. Metaheuristics are classified into population-based and single-solution based algorithms in Figure 2.3, however, in fact, several classifications are proposed in literature as shown in Figure 2.8 by Dréo & Candan [2007].

In designing a population-based metaheuristic (A2.2), two contradictory behavior must be taken into account: exploration of the search space (diversification) and exploitation of the best solutions found (intensification). Promising regions are determined by the obtained "good" solutions. In exploitation, the promising regions are explored more thoroughly in the hope to find better solutions. In exploration, non-explored regions must be visited to be sure that all regions of the search space are evenly explored and that the search is not confined to only a reduced number of regions. In this design space, the extreme search algorithms in terms of the exploration are random search. While, the exploitation is an iterative improvement local search. In random search, at each iteration, one generates a random solution in the search space. No

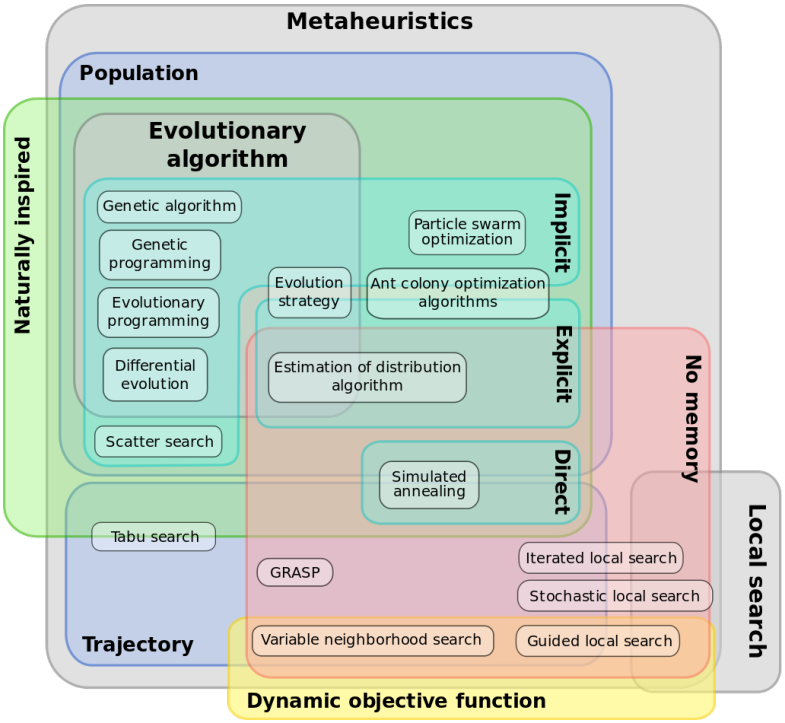


Figure 2.8: Classification of metaheuristics [Dréo & Candan, 2007].

search memory is used. In the basic steepest local search algorithm, at each iteration one selects the best neighbor solution that improves the current solution [Črepinšek *et al.*, 2013].

2.3.1 Multi-Objective Metaheuristics

In addition to the common concepts of mono-objective metaheuristics, a multi-objective metaheuristic contains three main search components:

- Fitness assignment: The main role of this procedure is to guide the search algorithm toward Pareto optimal solutions for a better convergence. It assigns a scalar-valued fitness to a vector objective function.

- Diversity preserving: The emphasis here is to generate a diverse set of Pareto solutions in the objective and/or the decision space.

- Elitism: The preservation and use of elite solutions (e.g., Pareto optimal solutions) allows a robust, fast, and a monotonically improving performance of a metaheuristic.

2.3.1.1 Fitness Assignment

For a given solution, a fitness assignment procedure maps a fitness vector to a single value. The fitness scalar value measures the quality of the solution. According to the fitness assignment strategy, multi-objective metaheuristics can be classified into four main categories:

- Scalar approaches: These approaches are similar to the exact scalarization methods discussed in Section 2.2.1.1, however, after transforming the multi-objective problem to a mono-objective one, the problem is solved using a metaheuristic method.
- Criterion-based approaches: In criterion-based approaches, the search is performed by treating the various non-commensurable objectives separately.
- Dominance-based approaches: The dominance-based approaches use the concept of dominance and Pareto optimality to guide the search process.
- Indicator-based approaches: In indicator-based approaches, the metaheuristics use performance quality indicators to drive the search toward the Pareto front.

2.3.1.2 Diversity

Diversity preserving is to generate a diverse set of Pareto solutions in the objective and/or the decision space. Population-based metaheuristics are reputed to be very sensitive to the choice of the initial population and the biased sampling during the search. Diversity loss is then observable in many of them. This phenomenon is called genetic drift in evolutionary algorithms. To face this drawback related to the stagnation of a population, diversity must be maintained in the population. Thus, diversity preservation strategies must be incorporated into multi-objective metaheuristics. In general, diversification methods deteriorate solutions that have a high density in their neighborhoods.

2.3.1.3 Elitism

In general terms, elitism consists in archiving the "best" solutions generated during the search (e.g., Pareto optimal solutions). A secondary population, named archive, is used to store these high-quality solutions. First, elitism has been used to prevent the loss of the obtained Pareto optimal solutions. In this passive elitism strategy, the archive is considered as a separate secondary population that has no impact on the search process. Elitism will only guarantee that an algorithm has a monotonically non-degrading performance in terms of the approximated Pareto front. Then, elitism has been used in the search process of multi-objective metaheuristics (active elitism), that is, the archived solutions are used to generate new solutions. Active elitism allows to achieve faster and robust convergence toward the Pareto front for a better approximation of

the Pareto front[Talbi, 2009]. However, a care should be taken to be trapped by a premature convergence if a high-elitist pressure is applied to the generation of new solutions.

2.3.2 Evolutionary Algorithms

In the nineteenth century, J. Mendelwas the first to state the baselines of heredity from parents to off-springs. Then in 1859, C. Darwin presented the theory of evolution in his famous book On the Origin of Species [Darwin, 1859]. In the 1980s, these theories of creation of new species and their evolution have inspired computer scientists in designing evolutionary algorithms. Different main schools of evolutionary algorithms have evolved independently during the past decades: genetic algorithms (GA), mainly developed by Holland [Holland, 1992]; evolution strategies; evolutionary programming; and genetic programming.

Evolutionary Algorithms (EAs) are stochastic population-based metaheuristics that have been successfully applied to many real and complex problems (epistatic, multi-modal, multi-objective, and highly constrained problems). They are the most studied population-based algorithms. Their success in solving difficult optimization problems in various domains (continuous or combinatorial optimization, system modeling and identification, planning and control, engineering design, data mining and machine learning, artificial life) has promoted the field known as evolutionary computation [Talbi, 2009].

EAs are based on the evolution of a population of individuals. Initially, this population is usually generated randomly. Every individual in the population is the encoded version of a tentative solution. An objective function associates a fitness value with every individual indicating its suitability to the problem. At each step, individuals are selected to form the parents, following the selection paradigm in which individuals with better fitness are selected with a higher probability. Then, selected individuals are reproduced using variation operators (e.g., crossover, mutation) to generate new offsprings. Finally, a replacement scheme is applied to determine which individuals of the population will survive from the off-springs and the parents. This iteration represents a generation. This process is iterated until a stopping criteria hold.

2.3.2.1 Genetic Algorithm

The genetic algorithm - presented in Algorithm 1 - is the most popular evolutionary algorithm. In this section we briefly describe the main mechanism of the genetic algorithm. Initially, an initial population is generated according to a heuristic rule or randomly. The population size typically depends on the nature of the optimization problem. Often, the initial population is generated in such a way as to allow a larger range of possible solutions inside the given search space. If the population size is too small, there is not enough evolution going on and consequently there is a risk of premature convergence towards a local optimum and ultimately extinction of

the population. However, a larger population will require more computational time and fitness evaluations.

At each successive generation, a percentage of the existing population is 'selected' to breed a new generation, thus ensuring the continuity of the population. Hence, a selection function chooses 'parents' based on a fitness-based selection process, where 'fitter' solutions are more likely to be selected. An individual can be selected more than once, in which case it transfers its genes to more than one offspring.

At each generation, the GA uses the current generation to create the new offspring that will define the next generation. The algorithm will apply a set of genetic operators (crossover and mutation) on the parents selected by the selection function to generate the children. Recombination (or crossover) is the combination of a pair of parents, analogous to biological reproduction. Mutated children are created by a random change (or mutation) of the genes of a single parent. Both genetic operators are essential for the success of the optimization search. Crossover enables the algorithm to preserve the best genes from different individuals and recombine them into possibly fitter children. This allows a better 'exploitation' of the search space. Whereas mutation increases the diversity of the population and permits a further 'exploration' of the search domain. The crossover probability is usually between 0.7 and 1.0, while the mutation probability is lower $0.001 \approx 0.05$ [Chehouri *et al.*, 2016].. Mutation probability is dependent upon the representation type and number of genes. In natural systems, if the mutation rate is too high under a high selection pressure, the population might become extinct. A suitable elitist selection function must be employed to avoid loss of good solutions. Selection, crossover and mutation are iteratively applied to the population until a stopping condition is satisfied.

Algorithm 1: Main loop of genetic algorithm

```
1 Generate initial population
2 Initialize the crossover and mutation probabilities
3 Evaluate fitness function of each individual
4 Selection of the current best for the next generation
5 Reproduction by crossover and mutation
6 while Stopping criteria is met do
7 |   Repeat 3 → 5
8 end
```

2.3.2.2 NSGAII

The Non-dominated Sorting Genetic Algorithm 2 (NSGAII) is criterion-based multi-objective genetic algorithm. Srinivas & Deb [1994] proposed NSGA in 1994 for multi-objective optimization. NSGA is different from a single-objective GA in the way the fitness is assigned to individuals.

In order to assign fitness, first the population is sorted according to their non-domination level. Thereafter, all solutions of the better non-dominated fronts are assigned better fitness. In order to preserve diversity among solutions of any front, a niche-preservation technique (sharing function method) is used to degrade raw fitness of solutions which are crowded by other solutions of the same front. NSGA is shown to find a number of non-dominated solutions to a variety of test problems and engineering design problems. The population is sorted in NSGAII based on non-domination. To lower the computing complexity, a particular bookkeeping procedure is used. Each solution is given a fitness rating based on its non-dominance level. To generate a child population, binary tournament selection, recombination, and mutation operators are used. The parent population is joined with the children population for each generation. This ensures elitism by allowing parent solutions to be compared to the child population. The population is then sorted according to non-domination. Following that, the solutions of the last accepted front are ranked using a crowded comparison criterion, and a fixed number of points is chosen. Because the diversity of the solutions is crucial, the greater crowded distance is used to compare two solutions belonging to the same Pareto front (distance from a solution to the nearest neighbor). In this way, less dense portions of the search space are given more weight when determining which solutions to choose.

2.3.3 Drawbacks and Conclusions

In the context of mechanical engineering optimization, the literature does not provide clear conclusion on which metaheuristic has the best performance. Yildiz *et al.* [2020] concluded that the best metaheuristic is the moth-flame optimization (MFO) on a benchmark of 6 mono-objective mixed-integer engineering design problems. Abderazek *et al.* [2019] found that the best metaheuristic is the mixed differential evolution (NAMDE) on a benchmark of 13 mono-objective mixed-integer engineering design problems. Dhiman *et al.* [2021] observed that EMoSOA - a new evolutionary multi-objective seagull optimization algorithm for global optimization - is the best performing algorithm on a benchmark of 4 multi-objective mixed-integer engineering design problems. Assiri [2021] concluded that the butter fly metaheuristic is the best on a benchmark of 4 mono-objective engineering design problems. Kaur *et al.* [2020] developed Emperor Penguin Optimizer (EPO) in terms of multi-objective problems solving capability, which is entitled as Multi-objective Emperor Penguin Optimizer (MOEPO). The authors observed that the outcomes from the empirical analyzes depict that the proposed algorithm is better than other existing algorithms on 7 multi-objective mixed-integer engineering problems. Therefore, it is not clear which is the metaheuristic that has the best performance on the mono-objective/multi-objective mixed-integer engineering problems.

As a matter of fact, Giagkiozis *et al.* [2015] reviewed in their recent work, the ability of seven metaheuristics to handle mixed variables problems. A scale ranging from 1 to 5 was given to to

measure the relative strengths and weaknesses of algorithm families for multi-objective problems with mixed variables. A value of 5 translates that this particular family of algorithms is very well suited for this type of problem, while a value of 1 means that the algorithm is poorly suited for this type of problems. The comparison showed that while six algorithms out of seven were able to handle mixed variables optimization, all of these six algorithms scored 1 in their suitability test. Socha [2009] mentioned in his review of native MINLP methods only 5 metaheuristics that can handle mixed-integer. One of them is the Genetic Adaptive Search (GeneAS) introduced by Deb [1997]. GeneAS is flexible enough to handle such MINLP engineering design problems since it uses a combination of binary-coded and real-coded GA depending on the nature of the design variables. Hence, although the mixed-integer nature of the problem is a serious issue in metaheuristics, GAs are kind of metaheuristics that are capable to handle such problems depending on the type of variable-coding technique used in GA. This flexibility is one of the reasons that have gained GAs their popularity in the MO-MINLP/MINLP field.

El Samrout [2019] in his thesis examined five well-known metaheuristics (A2.2) to the combinatorial optimization scientific society that are: (1) NSGAII; (2) MOPSO; (3) Cucko search; (4) Bat algorithm; and (5) firefly algorithm on three MO-MINLP engineering design problems. By comparing the obtained solution from the five metaheuristics to the true Pareto front of each problem, the author concluded that "one can safely assume that the traditional metaheuristics were unable to solve the three optimization problem". Even in the case of best metaheuristic, which is NSGAII in this case, not all Pareto solutions were found by the metaheuristic. Even with the use of the best metaheuristic for the three optimization problems, the calculated Pareto front is still far from being a good solution. The low percentage of intersection of solutions between the true front and the calculated one for the three problems verifies this conclusion. Moreover, "traditional" metaheuristics are not able, by nature, to handle mixed variables constrained problems. Only some of them can be considered native methods as mentioned previously. Thus, the need for a better approach to solve this kind of problems is now a point of interest of many researchers. Recently a new line of research has been arising which is hybridization of distinct algorithms. In the next section, an overview of those hybrids is presented.

2.4 Hybrid Methods

In recent years, a large number of algorithms that do not strictly fit the paradigm of a single classic metaheuristic have been published. They, on the other hand, integrate a variety of algorithmic components, many of which are derived from algorithms developed in other fields of optimization research. Hybrid metaheuristics are a term used to describe these approaches [Blum *et al.*, 2011].

Nowadays, the hybridization of metaheuristics has become a common strategy to solve optimization problems. The huge number of efficient hybrid metaheuristics proves that hybrid metaheuristics represent actually the most efficient algorithms for many classical and real-life difficult problems [El Samrout, 2019]. The major rationale for combining multiple algorithms is to take advantage of the complementary nature of diverse optimization strategies; hybrids are thought to gain from synergy. In fact, selecting the right mix of complementary algorithmic approaches can be the key to attaining top performance in various of difficult optimization problems. [Blum *et al.*, 2011]. Ehrgott & Gandibleux [2008] review the literature concerning the hybrid methods for multi-objective optimization. Blum *et al.* [2011] review the single objective hybrids and their classifications. Puchinger & Raidl [2005] discuss different approaches of combining exact algorithms and metaheuristics to solve combinatorial optimization problems.

2.4.1 Hybrids Classifications

The first approach for hybridization was to consider the cooperation between metaheuristics and other metaheuristics. At first, it seemed like the only straightforward approach, and others ways to hybridize metaheuristics were neglected. It was not until researchers realized the complementary aspect between some exact methods and metaheuristics that this type of hybridization emerged [El Samrout, 2019]. In fact, exact methods are often used for combinatorial optimization problems with small instances. They are known for their capability to solve these small instances problems and asses their optimality. However, they are not used to solve large NP-hard problems because they are computationally expensive. Hence, in this section, hybrids are classified into two groups: hybridization of metaheuristics with metaheuristics and hybridization of approximate methods with exact methods. A special concern is considered for the multi-objective hybrids and their classifications.

2.4.1.1 Metaheuristics with Metaheuristics (A2+A2)

Most researchers started looking at possible combinations with heuristics or other metaheuristics when they first examined hybridizing their favored metaheuristic with another technique for optimization. In fact, hybridization of various metaheuristics is fairly common nowadays, particularly when it comes to the use of local search methods within population-based methods. Nature-inspired methods, for example, are effective when it comes to exploring the search space and identifying areas with high-quality solutions. They usually try to acquire a global view of the search space during initialization. Then, as the search progresses, they narrow the search to more promising areas of the search space. However, they are usually not so effective concerning the exploitation of the accumulated search experience, that is, finding the best solutions in these high quality areas. The strength of local search, on the other hand, is its capacity to

rapidly locate better solutions in the neighborhood of certain starting solutions. In conclusion, population-based methods are effective in identifying promising parts of the search space, from which local search methods can quickly select the best solutions. This is why hybridization of this type is usually quite effective. This type of hybridization is relatively common. Several classifications and groups can be found in literature. Interested readers can refer to [Blum *et al.*, 2008, 2011; Ehrgott & Gandibleux, 2000; El Samrout, 2019]

2.4.1.2 Exact with Approximate (E+A)

Puchinger & Raidl [2005] present a general classification of existing approaches combining exact and metaheuristic algorithms for combinatorial optimization. The authors distinguish the following two main categories:

- Collaborative Combinations: The algorithms share information but are not one and the same. Exact and heuristic algorithms can be run sequentially, in parallel, or in combination.
- Integrative Combinations: A subordinate embedded component of one technique is a subordinate embedded component of another approach. As a result, there is an unique master algorithm and at least one integrated slave, which can be either an exact or a metaheuristic algorithm.

2.4.1.2.1 Collaborative Combinations They are top-level hybrids of metaheuristics and exact approaches in which the algorithms share information but are not part of each other. They are distinguished by the manner in which they are executed out, i.e. sequential or parallel. In sequential methods, either the exact method is executed as a kind of pre-processing before the metaheuristic, or vice-versa. Instead of a strictly sequential batch approach, exact and heuristic algorithms may also be executed in a parallel. This type of hybrids is less frequent. An illustrative example on this type of hybrids is the asynchronous teams (A-Teams) introduced by Talukdar *et al.* [1998]. An A-Team is a problem solving architecture in which the optimization algorithms are connected into a cyclic directed network. The basic idea of A-Teams is having these algorithms work asynchronously on shared memories that contain the trial solutions of the problem. Hence, each algorithm can add, delete or alter a solution in the memory.

2.4.1.2.2 Integrative Combinations They integrate exact algorithms and metaheuristics in such a way that one technique is a subordinate embedded component of another. Exact algorithms could be integrated with metaheuristics in ways such as using exact algorithms to search neighborhoods in local search-based metaheuristics, or enhancing metaheuristics with information gained from the exact solution of the relaxed problem, or even using exact algorithms as

metaheuristic decoders (as GA). On the other side, metaheuristics/heuristics could be incorporated with exact algorithm. The most popular approach is combining approximate methods with tree search algorithms (as branch and bound) in which the tree search technique explores the domain while heuristics and metaheuristics are used to determine the bounds and the optimal solutions.

2.4.1.3 Multi-objective Hybrids

While the previous sections may contain some mono-objective optimization hybrids, this section is dedicated to focus on the hybridization in multi-objective optimization. An "obvious" preliminary observation is that a lot of hybrid metaheuristics in this field were based on the cooperation between trajectory-based (or single solution based) metaheuristics and population based metaheuristics. The cooperation between these two different approaches for multi-objective optimization is justified by the fact that these two approaches are complementary to each other. Trajectory-based metaheuristics are well known for their capacity to intensify the search; they are able to converge aggressively towards good solutions, but they need to be guided along the non-dominated frontier. On the other hand, population-based approaches are good explorers of the search space; they are very well suited to maintain a diversified population of solutions along the non-dominated frontier, but often converge too slowly to the non-dominated frontier. Naturally, methods that try to take advantage of these features by combining components from both approaches, represent often a good trade.

Ehrgott & Gandibleux [2008] classify the multi-objective metaheuristics-metaheuristics hybrids based on the aim of hybridization, as follows:

- Hybridization to make a method more aggressive: the aim is to enhance the search aggressiveness of the approximation procedure. A clear example is to use multi-objective EA that drives the search procedure, and then activates a neighbor search algorithm, with the aim to improve as far as possible the promising solutions resulting from the evolutionary operators.

- Hybridization to drive a method: the aim is to provide guiding mechanism along the non-dominated frontier. Where global information about the current approximation is deduced from population-based metaheuristics (such as multi-objective GA) and then drives the local search processes in order to "guarantee" a good coverage of the non-dominated frontier.

- Hybridization for exploring complementary strengths: the idea is to use the complementary forces of the metaheuristics. The aim to take the advantage of, for example, a GA algorithm for building a set of good solutions in a first step followed by an aggressive search method (e.g., Tabu-search) in a second step.

El Samrout [2019] classifies the multi-objective metaheuristics-exact hybrids based on the concept of the algorithm, as follows:

- Metaheuristic to generate an upper bound: At first the multi objective metaheuristic is executed to get a Pareto set approximation. This approximation is used to initialize the multi-objective exact algorithm. Hence, many nodes of the search tree can be pruned by the branch and bound algorithm.
- Exact algorithm to solve subproblems: In this hybrid approach, the exact multi-objective algorithm solve subproblems that are generated by the multi-objective metaheuristic.

In this context, several trials of combination between metaheuristics and machine learning are proposed, however, these approaches are out of the thesis scope. Interested readers can refer to [El Samrout, 2019].

2.4.2 Combining Approximate Methods with Tree Search (A + E2.1)

The hybridization of metaheuristics/heuristics with tree search techniques is probably one of the most popular lines of combining different algorithms for optimization. One of the basic ingredients of an optimization technique is a mechanism for exploring the search space, that is, the space of valid solutions to the considered optimization problem. The hybridization of metaheuristics with branch and bound concepts is rather recent [Blum *et al.*, 2008]. On one side, it is possible to use branch and bound concepts within metaheuristics in order to increase the efficiency of the metaheuristics search process. On the other side, metaheuristics can be used within branch and bound in order to reduce the space and time consumption of branch and bound. Ideally, such hybrids are expected to obtain higher quality solution(s) than that obtained by metaheuristics with less effort than that consumed by branch and bound as illustrated in Figure 2.9.

2.4.2.1 Combining mono-objective branch and bound with metaheuristics (A2+E2.1)

In [Nagar *et al.*, 1996], one of the first works on the combination of branch and bound with an evolutionary algorithm. The first stage of the algorithm, branch and bound is executed up to a fixed tree level k. Lower bounds evaluated using a heuristic method. The second stage consists in the execution of the evolutionary algorithm on the active nodes. In [Cotta & Troya, 2003], a framework which is based on using branch and bound as an operator embedded in the evolutionary algorithm. The resulting hybrid operator will explore the possible children of the solutions being recombined, providing the best combination of them that can be constructed without introducing implicit mutation. In [Woodruff, 1999], a chunking based strategy to decide at each node of the branch and bound tree whether or not a metaheuristic is called in order

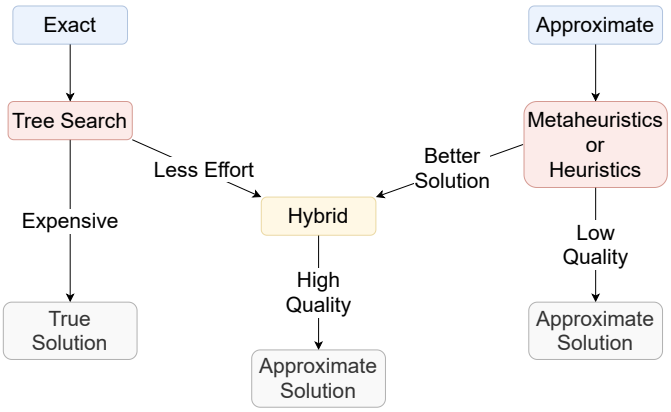


Figure 2.9: Combining approximate methods with tree search.

to eventually find a better incumbent solution. In [Jozefowiez *et al.*, 2007], NSGAII generates potentially Pareto optimal solutions, which are used to build subproblems; these subproblems are then solved using the branch-and-cut algorithm. In [Puchinger *et al.*, 2004] the evolutionary algorithm is based on an order representation, specific recombination and mutation operators, and a decoding heuristic. In one variant, branch-and-bound is occasionally applied to locally optimize parts of a solution during decoding. In [Blum *et al.*, 2011], only the remaining ("free") variables are optimized by the MIP-solver, with an appropriate part of the decision variables set to their values in an incumbent solution. If the MIP-solver discovers a better solution, it replaces the incumbent. Then, a new neighborhood is defined around it, and the process is repeated. In [Blum *et al.*, 2008], The branch and bound concepts are used to improve the efficiency of the ant colony optimization solution development process. In [Backer *et al.*, 2000], The authors only utilize the constraint programming system to assess the validity of solutions and identify the values of constrained variables, not to search for solutions, as it employs the depth-first branch and bound technique. Extensive literature reviews on this topic is available in [Blum *et al.*, 2011; Ehrgott & Gandibleux, 2008; Puchinger & Raidl, 2005].

2.4.2.2 Combining multi-objective branch and bound with heuristics (A1 + E2.1)

In the other hand researchers had done several trials of implementation of MCBB based on heuristic methods. In [Cacchiani & D'Ambrosio, 2017] the author propose a heuristic based on a branch-and-bound algorithm. It starts with a set of feasible points, obtained, at the root node of the enumeration tree, by iteratively solving, with an ϵ-constraint method, a single objective model that incorporates the other objective functions as constraints. Lower bounds are derived by optimally solving Non-Linear Programming problems (NLPs). Each leaf node of the enumeration tree corresponds to a convex multi-objective NLP, which is solved heuristically by varying the

weights in a weighted sum approach. In [Florios *et al.*, 2010], the authors use exact as well as approximate algorithms. The exact algorithm is a properly modified version of the multi-criteria branch and bound (MCBB) algorithm, which is further customized by suitable heuristics. Three branching heuristics and a more general purpose composite branching and construction heuristic are devised. In [Wang & Liu, 2013], the authors developed a heuristic method based on a branch and bound structure. Several lower bounds, upper bounds and dominance rules are combined in to the proposed heuristic method.

2.4.2.3 Discussion and Conclusion

The performance of metaheuristics on MINLP problems has been discussed in Section 2.3 concluding that the current performance is not sufficient for this type of problems. On the other hand, exact method are too expensive when they are applied to a large scale problems. Many researchers proposed to use hybrids of metaheuristics with mono-objective branch and bound to enhance the performance of metaheuristics. Others suggested using heuristics with multi criteria branch-and-bound. However, to the best of our knowledge, the lack of hybridization of multi-criteria branch and bound with metaheuristics in literature is noticed, in contrary to the mono-objective branch and bound as illustrated in Figure 2.10. This lack in literature may be due to the apprehension of high computational effort. Hence, any design of such hybrids must consider the "investment" in computational effort. In which, hybrids that combine MCBB with metaheuristic are expected to consume more effort than the metaheuristic it self. However, what if this investment in the computational effort is added to the metaheuristic itself (as increasing the maximum generations or population size in GA)? In fact, this is an open research question that should be considered when examining the performance of such hybrids. Moreover, Blum *et al.* [2011] have observed that the literature shows that a certain hybrid might work well for specific problems, but it might perform poorly for others. Nevertheless, there are hybridization types that have shown to be successful for many applications. They may serve as a guidance for new developments.

2.5 Conclusion and Propositions

Solving Multi-Objective Mixed Integer Non-Linear Programming (MO-MINLP) problems is a point of interest for many researchers as they appear in several real-world applications, especially in the mechanical engineering field. Many researchers have proposed using hybrids of metaheuristics with mono-objective branch and bound. Others have suggested using heuristics with Multi-Criteria Branch and Bound (MCBB). This thesis aims to enhance the solution of MO-MINLP problems by proposing a general hybrid approach. The proposed approach is based on (1) the exact enumeration of the combinatorial domain using the Multi-Criteria Branch and Bound

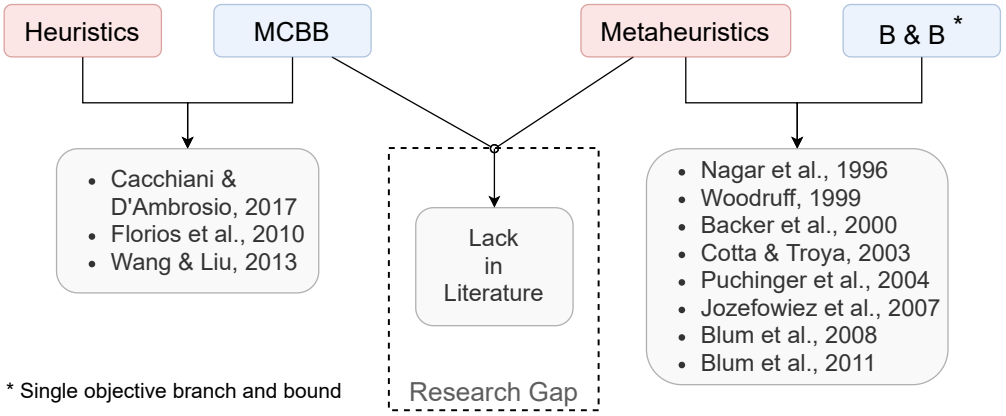


Figure 2.10: Hybrid tree search with approximate methods review chart.

(MCBB) algorithm (E2.1); (2) seeking a good approximated optimality using - (A2.2) - the Non-dominated Sorting Genetic Algorithm 2 (NSGAII). Our approach is referred to as BnB-NSGAII. Furthermore, a new feature is also proposed to enhance the performance of BnB-NSGAII, called the legacy feature. The legacy feature permits the inheritance of the elite individuals between the parents and their children nodes in the branch and bound tree.

A main component of any branch and bound algorithm is the branching strategy. Branching strategy is a crucial task that affects the performance of MCBB. *This thesis intends to enhance the performance of MCBB* by proposing new branching strategies: "Branching by Anchor points" and "Branching by Pareto elements". The proposed strategies are based on interval branching concept that benefit from the solution of the parent node to create children nodes.

Real-world problems, especially mechanical engineering problems, are usually constrained problems. Constraint Handling Techniques (CHT) have high impact on the performance of evolutionary algorithms and consequently on genetic algorithms as NSGAII. *This thesis intends to enhance the performance of NSGAII* by proposing a new ranking-based CHT that is based on the non-dominated sorting of constraint violation (NSCV Ranking). As a result, our proposal consists in four features as shown in Figure 2.11.

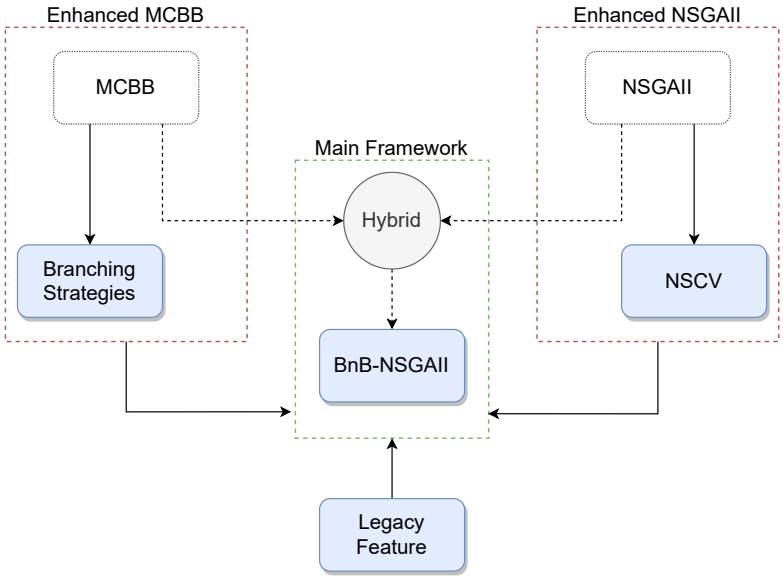


Figure 2.11: An overview of the proposed approach.

Chapter 3

Hybrid Branch and Bound Based Genetic Algorithm

Contents

3.1 Introduction **42**
3.2 MCBB **43**
3.3 BnB-NSGAII **44**
3.3.1 Parameters Tuning 45
3.3.2 BnB-NSGAII Algorithm 47
3.3.3 BnB-NSGAII Legacy Feature 49
3.4 Branching Strategy **50**
3.4.1 Branching by Anchor Points 52
3.4.2 Branching by Pareto Elements 55
3.5 Constraint Handling Technique **56**
3.5.1 Literature Review 57
3.5.2 NSCV 63
3.6 Conclusion **67**

3.1 Introduction

The general Multi-Objective MINLP problem ($\mathcal{P}_{\text{MO-MINLP}}$) is written as

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{minimize}} \quad & \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}) = f_1(\boldsymbol{x},\boldsymbol{y}),\ldots,f_p(\boldsymbol{x},\boldsymbol{y}) \\ \text{subject to} \quad & \\ & g_l(\boldsymbol{x},\boldsymbol{y}) \leq 0,\, l = 1,...,m \\ & \boldsymbol{x} \in [\underline{\boldsymbol{x}},\bar{\boldsymbol{x}}], \quad \boldsymbol{x} \in \mathbb{R}^{n_c} \\ & \boldsymbol{y} \in \{\underline{\boldsymbol{y}},\bar{\boldsymbol{y}}\}, \quad \boldsymbol{y} \in \mathbb{N}^{n_i} \end{aligned} \tag{3.1}$$

where f and g are the objective and constraint real functions respectively that are not necessarily convex or differentiable, p and m are the number of objectives and constraints respectively. $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the variables of the problem where n_c and n_i are the number of continuous and integer variables respectively.

The optimal solution of a multi-objective problem is a set of non-dominated solutions called Pareto front $\mathcal{F}^*$. Each non-dominated solution is called Pareto element. For a given $F^1 = F(x^1,y^1)$ and $F^2 = F(x^2,y^2)$ vector values of objective function for two given "points" solution in the feasible set, the dominance relation between them is defined by:

Definition 1 *$\boldsymbol{F}^1$ dominates $\boldsymbol{F}^2$, ($\boldsymbol{F}^1 \prec \boldsymbol{F}^2$), when no value of $\boldsymbol{F}^2$ is less than $\boldsymbol{F}^1$ and at least one value value of $\boldsymbol{F}^2$ is strictly greater than $\boldsymbol{F}^1$ [Srinivas & Deb, 1994].*

$$\forall\, k \in \{1,\ldots,p\} \quad f_k^1 \leq f_k^2; \quad \exists\, k^* \in \{1,\ldots,p\} : f_{k^*}^1 < f_{k^*}^2. \tag{3.2}$$

Definition 2 *An anchor point corresponds to the optimal value of one objective function in the feasible domain. Thus, p objective functions provide up to p anchor points.*

Definition 3 *The ideal point is a specific point in the objective space that corresponds to to the optimal value of all objectives simultaneously.*

$$P_i^I \triangleq \min f_k(\boldsymbol{x}_i,\boldsymbol{y}_i); \quad \forall\, k = 1,\ldots,p. \tag{3.3}$$

In this chapter, we propose a hybrid method based on MCBB and NSGAII to tackle this type of optimization problems called BnB-NSGAII. Additionally, we propose the legacy feature to enhance the performance of BnB-NSGAII. To furthermore enhance the performance of BnB-NSGAII, we propose to enhance the performance of MCBB and NSGAII independently. Hence, we propose new interval branching strategies to enhance the performance of MCBB. And since we are dealing with constrained problems, we propose a new CHT called NSCV to enhance the performance of NSGAII.

This chapter is organized as follows. First, MCBB is presented in Section 3.2. Then, BnB-NSGAII is introduced in Section 3.3. Next, the new branching strategies are demonstrated in Section 3.4. Finally, in Section 3.5, a general overview of distinct CHTs is addressed then NSCV technique is introduced.

3.2 MCBB

The main concept of MCBB is to convert $\mathcal{P}_{\text{MO-MINLP}}$ into several mono-objective Mixed-Integer Non-Linear Problems $\mathcal{P}_{\text{MINLP}}$ and Multi-Objective continuous Non-Linear Problems $\mathcal{P}_{\text{MO-NLP}}$. This is achieved by constructing a combinatorial tree (Figure 3.1) that aim to partition the root node problem - $\mathcal{P}_{\text{MO-MINLP}}$ - into a finite number of subproblems (nodes) $N_1, \ldots, N_i, \ldots, N_n$, $i \in \{1, \ldots, n\}$, where n is the total number of nodes. The partitioning (branching) is done by modifying the upper ($\bar{\boldsymbol{y}}$) and lower ($\underline{\boldsymbol{y}}$) bounds of the integer variables $\boldsymbol{y}$. Then each node is solved repeatedly for each objective function independently as $\mathcal{P}_{\text{MINLP}}$. Solving a node is to determine its lower and upper bounds. The upper bound of a node $\mathcal{F}_i^*$ is its Pareto front, which is then stored in an incumbent list $\mathcal{F}^*$ that contains all the candidate solutions. Whilst the lower bound is the ideal point P_i^I of the current node. By solving N_i, one of the following is revealed:

- N_i is infeasible, means that the solver didn't find any solution that satisfies all constraints, i.e. the feasible domain is empty due to the modification of the upper and lower bounds of integer variables. Hence, N_i is pruned (*fathomed*) by *infeasibility*.
- N_i is feasible, but, the current lower bound P_i^I is dominated by a previously found upper bound $\mathcal{F}^*$. Therefore, N_i is fathomed by *optimality*.
- N_i is feasible, and, P_i^I is not dominated by $\mathcal{F}^*$, $P_i^I \prec \mathcal{F}^*$. $\mathcal{F}^*$ is then updated by adding $\mathcal{F}_i^*$ to it. Hence, $\mathcal{F}^* = \bigcup_i \mathcal{F}_i^*$, $\forall\, \mathcal{F}_i^* \prec \mathcal{F}^*$.

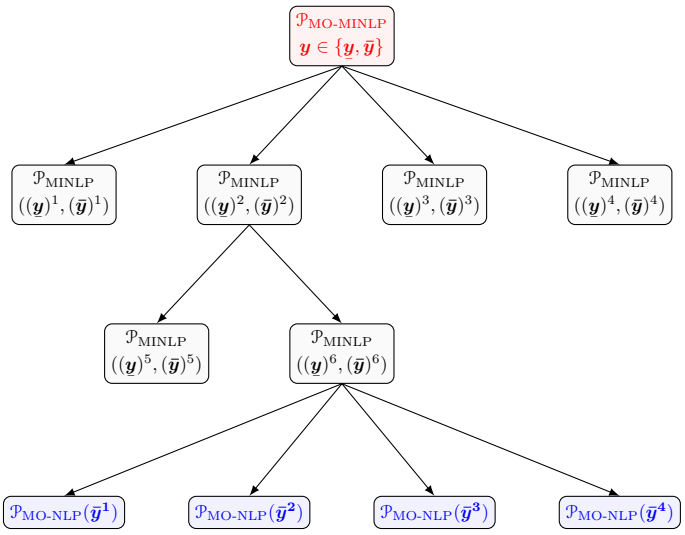


Figure 3.1: Illustrative example of the branching tree of MCBB on the problem $\mathcal{P}_{\text{MO-MINLP}}$.

If the node is not fathomed, the combinatorial tree is farther branched by dividing N_i into farther subproblems, called children nodes. If a node cannot be divided anymore, it is called a leaf node. Leaves are solved as $\mathcal{P}_{\text{MO-NLP}}$, since all integer variables are fixed such that $\boldsymbol{y}^i = \bar{\boldsymbol{y}}^i = \underline{\boldsymbol{y}}^i$, where $\bar{\boldsymbol{y}}^i$ and $\underline{\boldsymbol{y}}^i$ are the upper and lower bounds of the integer variables for the current node i. Hence, $\mathcal{P}_{\text{MO-NLP}}$ can be written as:

$$\begin{aligned} \underset{\boldsymbol{x},\, \bar{\boldsymbol{y}}^i}{\text{minimize}} \quad & \boldsymbol{F}(\boldsymbol{x}, \bar{\boldsymbol{y}}^i) = f_1(\boldsymbol{x}, \bar{\boldsymbol{y}}^i), \ldots, f_p(\boldsymbol{x}, \bar{\boldsymbol{y}}^i) \\ \text{subject to} \quad & \\ & c_l(\boldsymbol{x}, \bar{\boldsymbol{y}}^i) \geq 0,\, l = 1, ..., m \\ & \boldsymbol{x} \in [\underline{\boldsymbol{x}}, \bar{\boldsymbol{x}}], \quad \boldsymbol{x} \in \mathbb{R}^{n_c}. \end{aligned} \tag{3.4}$$

$\mathcal{F}_i^*$ of each leaf is then added to $\mathcal{F}^*$. MCBB is converged when all the nodes are leaves or fathomed. The overall Pareto front is obtained by removing the dominated elements from $\mathcal{F}^*$. Figure 3.2 shows the flowchart of MCBB approach.

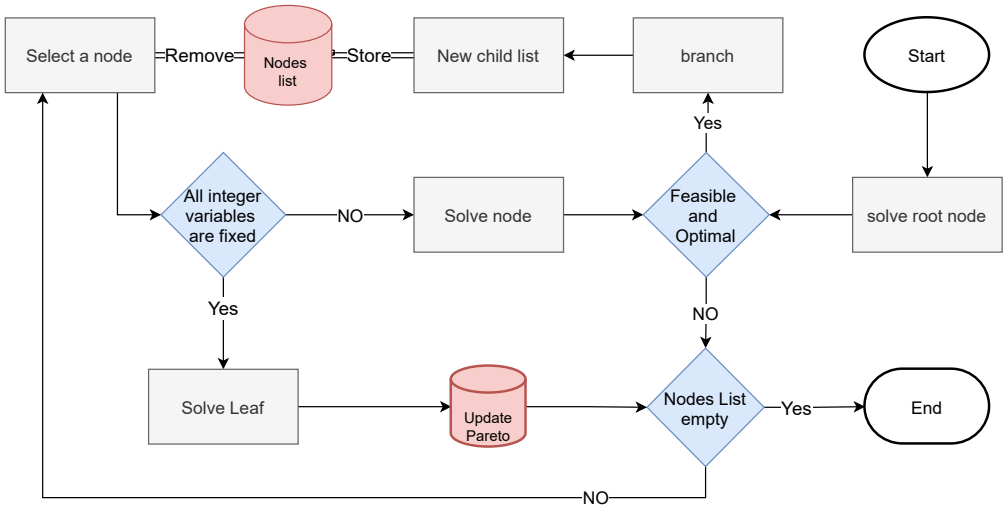


Figure 3.2: Flowchart of MCBB algorithm.

3.3 BnB-NSGAII

To determine a good approximated Pareto fronts, we propose a general hybrid approach based on an MCBB and NSGAII, in which to the best of our knowledge, this is the first attempt to combine MCBB with metaheuristics. The success of using metaheuristics with mono-objective branch and bound is the main motive to propose using them with the multi-criteria branch and bound. As mentioned in Chapter 2, NSGAII consists in exploration force and exploitation one. In our proposed algorithm, MCBB is used to enhance the exploration force of NSGAII by

investigating the mixed-integer domain space through branching it to subdomains, then NSGAII bounds each one. In this way, MCBB guides the search using the lower bounds obtained by NSGAII. We call this approach BnB-NSGAII.

$\mathcal{P}_{\text{MO-MINLP}}$ is complex and expensive to solve. The general idea is thus to solve several simpler problems instead. A trivial approach is to enumerate all possible configurations of integer variables. Then for each combination, $\mathcal{P}_{\text{MO-MINLP}}$ is converted into $\mathcal{P}_{\text{MO-NLP}}$. This technique is applicable when the number of configurations is relatively small, i.e the combinatorial space of the integer variables is small. The computation time increases prohibitively when the combinatorial space is large. The growth is exponential with respect to the number of integer variables. Instead, MCBB is used to divide $\mathcal{P}_{\text{MO-MINLP}}$ by constructing a combinatorial tree that aim to partition the root node problem, $\mathcal{P}_{\text{MO-MINLP}}$, into a finite number of subproblems.

The main concept of MCBB is to solve each objective function in each node N_i independently, i.e. MCBB converts $\mathcal{P}_{\text{MO-MINLP}}$ to p mono-objective MINLP problem at each node N_i. In contrary, BnB-NSGAII divides the domain of $\mathcal{P}_{\text{MO-MINLP}}$ into smaller subdomains by constructing an exploratory tree shown in Figure 3.3 that aims to direct the exploitation force in NSGAII to potential elements of the solution that were out of its sight. Therefore, the main difference between MCBB and BnB-NSGAII is the aim of partitioning of the problem, although, both use the same partitioning principle. In fact the idea of BnB-NSGAII is inspired by El Samrout [2019] who recommends to use a mono-objective metaheuristic to bound the nodes in MCBB and Cacchiani & D'Ambrosio [2017] who proposes to use a multi-objective -weighted sum- method to bound the nodes of MCBB.

3.3.1 Parameters Tuning

In NSGAII, population size, number of generations, cross-over and mutation probabilities,...etc affect its performance. Tuning these parameters may result in a more desirable solution. In BnB-NSGAII, we propose three different tuning levels: root, node and leaf levels. This provides three levels of parameters tuning of NSGAII aiming for more adaptability to distinct problems. Despite, similar to NSGAII, these parameters are highly problem dependent. Hence, a good parameter tuning depends on the expertise of the designer and his pre-knowledge about the problem being optimized.

Generally, in BnB-NSGAII, three different instances of NSGAII are called at different levels: root, node and leaf levels. At the root level, NSAGII_{root} is called to initially solve the problem $\mathcal{P}_{\text{MO-MINLP}}$ obtaining the initial upper bound $\mathcal{F}_0^*$. This is done in the initialization level illustrated in Figure 3.3. The quality of $\mathcal{F}_0^*$ has a huge effect on the convergence of the explanatory tree. Here, the term quality is defined by (1) the number of solutions obtained; (2) the distance between the obtained solution(s) and the global optimum one(s). In fact, these two quality indicators directly affects the fathoming potential of BnB-NSGAII in the following ways. As the

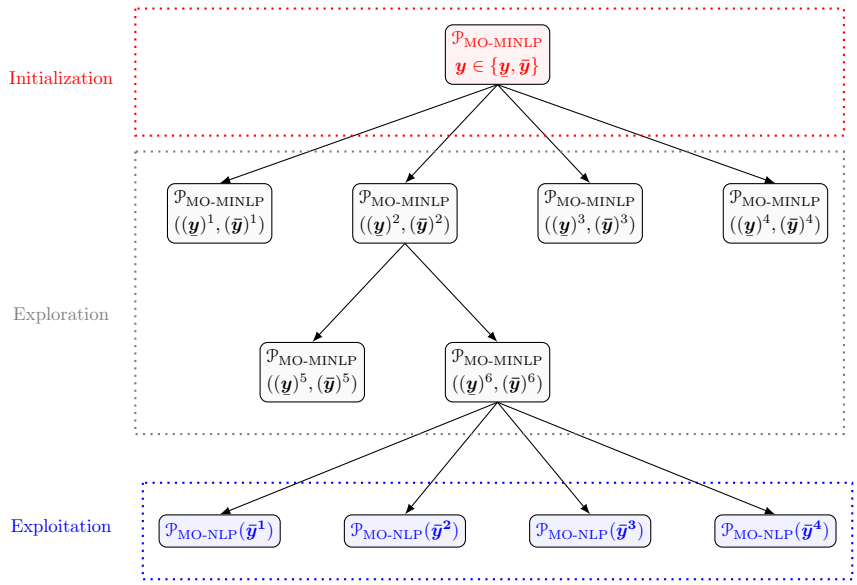


Figure 3.3: Illustrative example of the branching tree of BnB-NSGAII on the problem $\mathcal{P}_{\text{MO-MINLP}}$.

number of solutions in $\mathcal{F}_0^*$ increases as the probability of fathoming the upcoming children nodes increases, especially the first generations. Since the ideal points of the children are compared to current upper bound. On the other side, the low distance between the solution(s) in $\mathcal{F}_0^*$ and the true Pareto aids BnB-NSGAII to eliminate the upcoming children nodes with low quality solutions resulting in reducing the size of the combinatorial tree. Therefore, it is important to obtain as best as possible $\mathcal{F}_0^*$ from $\text{NSGAII}_{\text{root}}$. And since $\text{NSGAII}_{\text{root}}$ is called once during BnB-NSGAII process, it is recommended to invest as much as reasonable computational effort (i.e. population size and number of generations) at this level. Increasing the mutation rate - which affects the exploration force of NSGAII- might be beneficial here, since NSGAII is exploring the entire space. In fact, the best tuning parameters of $\text{NSGAII}_{\text{root}}$ are the ones that the decision maker could set if he attends to solve the problem by NSGAII itself.

At the node level, BnB-NSGAII aims to explore the domain space, as shown in Figure 3.3, by solving the same $\mathcal{P}_{\text{MO-MINLP}}$ but with smaller subspaces. Hence, in principle, $\text{NSGAII}_{\text{node}}$ may have the same parameters as $\text{NSGAII}_{\text{root}}$ except the computational effort parameters (i.e. population size and number of generations). These parameters may be tuned based on how much the decision maker intends to explore the space, regarding that $\text{NSGAII}_{\text{node}}$ is called frequently during BnB-NSGAII process. The exploration force of BnB-NSGAII is directly affected by the selected parameters at this level. At the leaf level, BnB-NSGAII ideally tries to capture the best Pareto solutions, hence, this is the exploitation level of BnB-NSGAII as shown in Figure 3.3. At this level, $\text{NSGAII}_{\text{leaf}}$ is called to solve a different type of problem, in particular $\mathcal{P}_{\text{MO-NLP}}$.

Hence, the decision maker might choose entirely different parameters such as cross-over and mutations functions, variable-coding technique, constraint handling technique,.. etc. Moreover, the decision maker may consider to reduce the mutation rate, since the subproblem is relatively small which does not necessarily require considerable exploration effort. Instead, the exploitation force is more advantageous here. Hence, the decision maker may choose to increase the cross-over rate or to invest more in the computational effort parameters depending on how much the accuracy is desirable.

3.3.2 BnB-NSGAII Algorithm

Algorithm 2 demonstrates the main loop of BnB-NSGAII showing the three tuning levels of BnB-NSGAII. At the root node, the incumbent list $\mathcal{F}^*$ is initialized by adding the initial Pareto front $\mathcal{F}_0^*$ obtained via $\text{NSGAII}_{\text{root}}$ by solving the main $\mathcal{P}_{\text{MO-MINLP}}$ problem. $\mathcal{F}^*$ is then updated as NSGAII enumerates the combinatorial tree by adding the non-dominated solutions found in each children and leaf nodes. The node is said to be a leaf by examining its lower and upper bounds of its integer variables $\boldsymbol{y^i}$, if $\underline{\boldsymbol{y}}^i = \bar{\boldsymbol{y}}^i$, then it is a leaf, otherwise it is a node. Hence, by identification of the node type, the suitable NSGAII solver is called to obtain the upper bound $\mathcal{F}_i^*$ of the node N_i. Then, the lower bound of the node P_i^I is deduced from $\mathcal{F}_i^*$ by applying Definition 3. If $\mathcal{F}^*$ dominates (Definition 1) P_i^I or the node is infeasible, then N_i is fathomed. Fathoming a subset consists in not exploring it because this one and therefore all the subsets included in it will not contain optimal Pareto front elements. A node can be fathomed if its lower bound vector is dominated *by at least one of the non-dominated points* of the incumbent set $\mathcal{F}^*$, i.e. $P_i^I \geq \mathcal{F}^*$, or if the node is infeasible meaning that NSGAII fails to find at least one solution that satisfies all the constraints.

If the node is not fathomed, then the incumbent list $\mathcal{F}^*$ is updated by adding the obtained Pareto elements in $\mathcal{F}_i^*$ of the node N_i. It should be noted here that the conventional MCBB updates the incumbent list by adding the anchor points of non-fathomed nodes. While BnB-NSGAII returns a set of Pareto solutions from each non-fathomed node. This increases the number of non-dominated solutions in $\mathcal{F}^*$ that the lower bound of any child node will be compared with. Hence, BnB-NSGAII provides a higher fathoming potential which directly affects the convergence of MCBB. This is noticed in the results obtained from the work of Cacchiani & D'Ambrosio [2017].

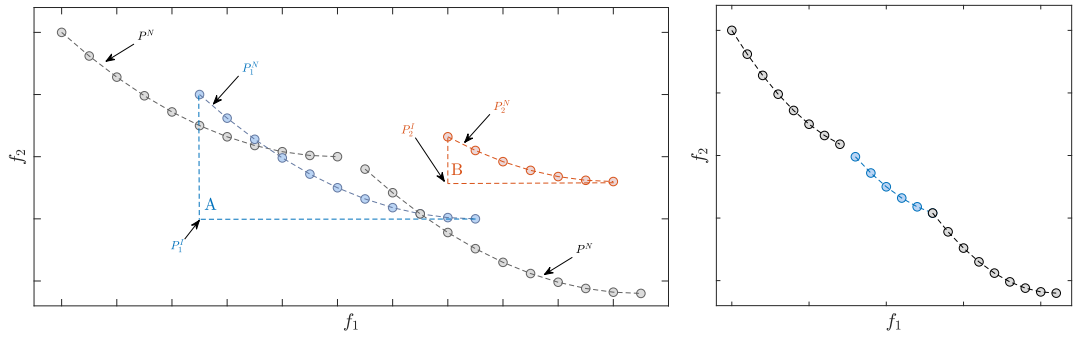
To this end, $\mathcal{F}^*$ may contain some solutions that are dominated by $\mathcal{F}_i^*$ and vice-versa as shown in the illustrative example in Figure 3.4a. In this example, two different scenarios are illustrated. $\mathcal{F}_1^*$ (blue) and $\mathcal{F}_2^*$ (red) are the Pareto fronts (upper bounds) obtained from two distinct nodes, say N_1 and N_2. Assume that the current incumbent list $\mathcal{F}^*$ is the one illustrated in black. The lower bounds of N_1 and N_2 are designated as A and B respectively. It is clearly observed that the point B is dominated by $\mathcal{F}^*$. Thus, N_2 is obviously fathomed. In contrary, although point

Algorithm 2: Main loop of BnB-NSGAII

```
Initialization
    F* ← ∅
    N ← ∅
    i ← 0                                                        // Node ID
create root node N_0: N ← N_0
while N ≠ ∅ do
    if y^i = ȳ^i then
        F*_i ← Solve N_i as P_MO-NLP (x, ȳ^i) by NSGAII_leaf     // N_i → leaf
    else
        if i=0 then
            F*_i ← Solve N_i as P_MO-MINLP (x, y) by NSGAII_root   // N_i → root node
        else
            F*_i ← Solve N_i as P_MO-MINLP (x, y^i) by NSGAII_node   // N_i → node
        end
    end
    P^I_i ← Ideal point of F*_i                                  // By Definition 3
    if P^I_i dominates F* And N_i is feasible then
        F* ← F* + F*_i                                           // Update Incumbent List
        F* ← Pareto filtering of F*                              // By Definition 1, see Figure 3.4
        if y^i ≠ ȳ^i then
            N_i' ← Branch( N_i )                                 // Create set of children nodes (Section 3.4)
            N ← N + N_i'                                         // Store children nodes in nodes list
        end
    end
    N ← Remove N_i from N
    i ← Select next node                                         // Selection of active node
end
```

A is not dominated by any of the solutions in $\mathcal{F}^*$, $\mathcal{F}_1^*$ contains some Pareto elements that are clearly dominated by $\mathcal{F}^*$. Moreover, $\mathcal{F}^*$ also contains some elements that are dominated by $\mathcal{F}_1^*$. Hence, each time $\mathcal{F}^*$ is updated, a Pareto filtering is done on it to ensure that all dominated elements are discarded from the incumbent list as shown in Figure 3.4.

If the non-dominated feasible node is not a leaf, then the node N_i is branched (divided to further subproblems). The nodes list $\mathcal{N}$ is updated by adding the obtained set of children nodes $\mathcal{N}_{i'}$ of the node N_i. To this end, N_i is fully explored, meaning that the node is solved, is checked for optimality and feasibility and is branched. Therefore, the node is then removed from the nodes list $\mathcal{N}_i$ to avoid exploring it again and a new node is selected to explore it. It should



(a) Incumbent List. **(b)** Filtered $\mathcal{F}^*$.

Figure 3.4: Example of updating the incumbent list in BnB-NSGAII.

be noted here that several techniques are proposed in the literature concerning the choice of active node, interested readers can refer to the references indicated in Figure 2.6. The process is then iterated until all the nodes are explored meaning that the node list $\mathcal{N}$ is an empty set. BnB-NSGAII is said to be converged if and only if the latter condition is verified. Hence, for a large combinatorial space problems, it is recommended to add a time limit for BnB-NSGAII as a stopping criteria in line 3 in Algorithm 2. Moreover, the decision maker should note that if $\text{NSGAII}_{\text{root}}$ fails to find a feasible solution(s), BnB-NSGAII will converge immediately after the line 14 in Algorithm 2 returning an empty Pareto set.

3.3.3 BnB-NSGAII Legacy Feature

In BnB-NSGAII, each node is solved independently. The output of each node is the captured Pareto front only. However, in NSGAII, the best population is that found in the last generation, since it contains the elite individuals among all the previous generations. This last population in each node is thus discarded during the BnB-NSGAII process, although it might be valuable to other nodes. Hence, we propose to permit the legacy between nodes. Where each child node inherits the last population from its parent node. The child node then initializes NSGAII by this population.

The children nodes are subproblems of their parent node. Thus, the boundary of parent node is different than that for the children nodes, i.e. $\boldsymbol{y}^i \neq \boldsymbol{y}^{i'}$ where i and i' are the node IDs for the parent and child nodes respectively. Hence, the population is rebounded - using Equation 3.5 on each individual I_{nd} in the population - before initializing NSGAII.

$$\boldsymbol{y}_{I_{nd}} = \max\{\min\{\boldsymbol{y}_{I_{nd}}, \bar{\boldsymbol{y}}^{i'}\}, \underline{\boldsymbol{y}}^{i'}\} \tag{3.5}$$

Rebounding the population may change the solution(s) leading to the loss of the elite individuals, though some of the elite genes may still conserved. This feature allows the child node to start

exploring the domain from where its parent finishes. Hence, this feature aims to guide the exploration force in BnB-NSGAII. Figure 3.5 illustrates the concept of the legacy feature.

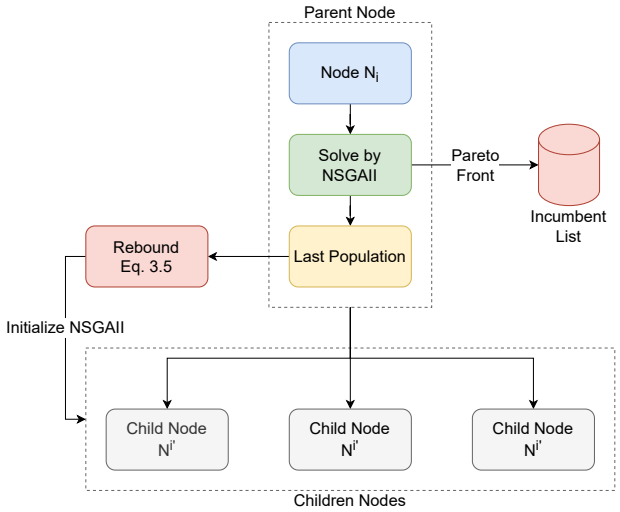


Figure 3.5: Concept of the legacy feature in BnB-NSGAII approach.

3.4 Branching Strategy

One of the main components of MCBB is the branching strategy that is called at line 18 in Algorithm 2. The branching strategy divides each node into several children nodes while ensuring that the union of the subdomains of the children nodes is equal to domain of the parent node. Although, all strategies works on the modification of the lower and upper bounds of the integer variables, each has its way of constructing the combinatorial tree. Ideally, the best strategy is that can converge MCBB with minimum exploration effort (i.e. examine minimum number of nodes).

The most commonly-used strategy in MCBB algorithms (e.g. Cacchiani & D'Ambrosio [2017]; Mavrotas & Diakoulaki [2005]) is to consecutively fix the integer variables in a specific order called "separation order". We call this approach "branching by integer" (BI) strategy. Where the combinatorial tree is constructed by fixing all the values of each integer variable as follows. For each level j, an integer variable is selected - according to the separation order - to be branched. The j^{th} selected integer variable is then fixed such as $y_j = a, \forall\ a \in \{\underline{y}_j, \bar{y}_j\}$. Meaning that for $j = 0$ - at the root node - all integer variables are free. While at the 1st level, the problem is divided to i' subproblems, where i' is the number of integer values in $\{\underline{y}_1, \bar{y}_1\}$ - y_1 is the 1st selected integer variable - and so on for further levels. Figure 3.6 shows an example of BI for four integer variables problem.

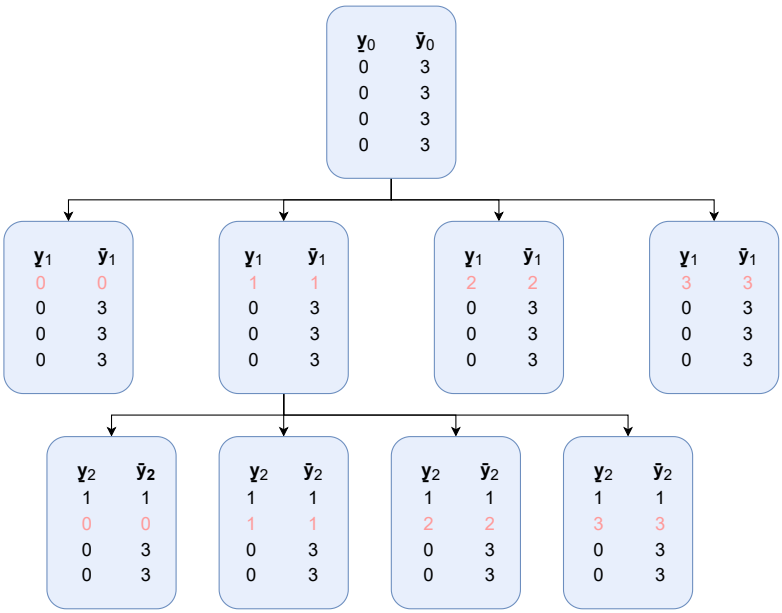


Figure 3.6: Example of branching by integer strategy.

This partitioning procedure is adopted from the single objective branch and bound. However, the strategies used in the single-objective case are not transposed so easily to the multi-objective case. Difficulties appear even with the most basic strategies, more details are available in Przybylski & Gandibleux [2017].

Other branching strategies exist in literature but they are not popular as BI such as interval branching strategies. Interval branching is commonly used in single-objective branch and bound to solve nonlinear optimization problems Araya *et al.* [2019]. This strategy starts with an initial node (root node) and then builds the search tree by splitting it into several sub-intervals. This is done by dividing the domain of one integer variable and generating the child nodes in the search tree. The procedure iterates until a termination criteria is reached. Generally, nodes are no longer branched when their sizes reach a given precision. In Araya *et al.* [2019], interval branching strategy is proposed for MCBB that aim to obtain a good envelope for the non-dominated vectors by splitting each domain in the middle into two sub-domains. Although, this branching strategy was designed for multi-objective case, the static splitting point (mid-point) remains the main issue in this approach.

In this thesis, we propose two multi-objective interval branching strategies that split the domain dynamically based on the obtained solution for each node in MCBB. We call them "Branching by Anchor Points" and "Branching by Pareto Elements".

3.4.1 Branching by Anchor Points

MCBB solves the current node N_i resulting in p anchor points, where p is the number of objectives in $\mathcal{P}_{\text{MO-MINLP}}$. In Branching by Anchor points (BA) strategy, $p+1$ intervals are then created by splitting one of the integer variables domain according to the defined separation order. The j^{th} selected integer variable domain in node i, $\boldsymbol{y}_j^i$ is divided as following:

$$\begin{aligned}\boldsymbol{y}_j^i = \{\underline{y}_j^1, (y_j^1)_1\} \cup \{(y_j^2)_1, (y_j^2)_2\} \\ \cdots \cup \{(y_j^{i'})_k, (y_j^{i'})_{k+1}\} \cdots \cup \{(y_j^s)_p, \bar{y}_j^s\},\end{aligned} \tag{3.6}$$

where k is the solution of the k^{th} objective function $k \in \{1, 2, \ldots, p\}$. Each interval $y_j^{i'}$ $\forall$ $i' \in \{1, \ldots, s\}$ represents a child node with a total of s children nodes, where $s \leq p+1$. Then, the intervals are properly shifted to avoid intersection, i.e. redundant solutions. For example, assume that $\boldsymbol{y}_j^i = \{0,1\} \cup \{1,2\} \cup \{2,3\}$, interval lower limits then shifted up by 1 starting from the second interval resulting in $\boldsymbol{y}_j^i = \{0,1\} \cup \{2,2\} \cup \{3,3\}$. Figure 3.7 shows an example of BA strategy for a bi-objective problem, where the integer variable vectors of the anchor points of the root node are $(\boldsymbol{y}_0^0)_1 = [1\ 1\ 1\ 1]$ and $(\boldsymbol{y}_0^0)_2 = [2\ 2\ 2\ 2]$.

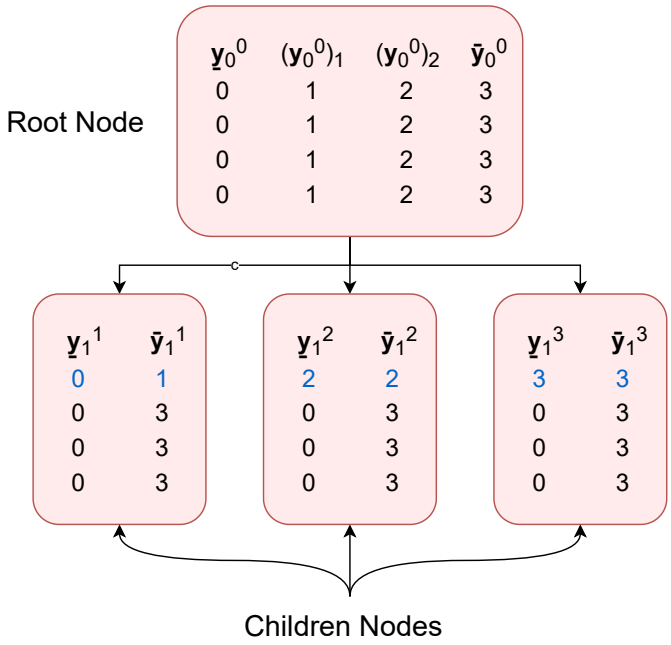


Figure 3.7: Example of BA strategy for a bi-objective problem.

3.4.1.1 Branching Algorithm

Algorithm 3 demonstrates BA strategy. The algorithm is designed assuming that for the i^{th} node, the integer domain $\{\underline{y}^i, \bar{y}^i\}$ and the node Pareto set $\mathcal{F}_i^*$ are stored in the node data structure

N_i. The branching function returns the list of children nodes $\mathcal{N}_{i'}$ as an output for each input N_i. After initializing the function, the anchor points of node N_i are obtained from the node upper bound $\mathcal{F}_i^*$. Then an integer variable $\boldsymbol{y}_j^i$ is selected. If $\underline{y}_j^i = \bar{y}_j^i$, i.e. the selected integer is singleton, another integer variable is selected. This is iterated until a non-singleton integer is selected. It should be noted that BnB-NSGAII (Algorithm 2) - or MCBB in general - does not call the branching function if all the integer variables are singletons since the node is thus a leaf. Equation 3.6 is then applied on the selected integer splitting it to s intervals. After adjusting the intervals limits, a child node N_i' is created for each interval $y_j^{i'}$ in y_j^i, $\forall\, i' \in \{1, 2, \ldots, s\}$. The children nodes are finally stored in the nodes list $\mathcal{N}_{i'}$.

Algorithm 3: Branching by Anchor Points Function

Input: N_i // Parent node
Output: $\mathcal{N}_{i'}$ // List of children nodes
Function `Branch`(N_i)**:**
  $\mathcal{N}_{i'} \leftarrow \emptyset$ // Initialize children nodes list
  $\mathcal{F}_i^* \leftarrow$ anchor points of node N_i
  $j \leftarrow$ select an integer variable
  while $\underline{y}_j^i = \bar{y}_j^i$ **do** // Seek for a non-singleton node
    $j \leftarrow j + 1$
  end
  $y_j^i \leftarrow$ equation 3.6 // Split the domain
  foreach $y_j^{i'} \in y_j^i$ **do**
    if $i' \neq 1$ ***And*** $\underline{y}_j^{i'} < \bar{y}_j^{i'}$ **then** // Shift interval limits
      $\underline{y}_j^{i'} \leftarrow \underline{y}_j^{i'} + 1$
    end
    Create child node $N_{i'}$
    $y^{i'} \leftarrow y^i$ // Set the integer domain as parent node
    $y_j^{i'} \leftarrow i'$ interval of y_j^i // Assign limits to the j^{th} integer variable
    $\mathcal{N}_{i'} \leftarrow \mathcal{N}_{i'} + N_i$ // Update children nodes list
    $i' \leftarrow i' + 1$ // Select next interval
  end
  return $\mathcal{N}_{i'}$
End Function

3.4.1.2 Constructing the Combinatorial Tree

The combinatorial tree (search tree) is constructed as illustrated in the flow chart in Figure 3.8. Starting from the root node, the node is solved using a MO-MINLP solver. Initially, the first integer is selected to be branched, i.e. $j = 1$. The process of selecting the integer variable (Step

4 in Algorithm 3) is the separation order of integer variables. There is several methods to define this order (see Section 4.3). Although in this thesis, the integer variable is selected in the order of declaration of variables. Starting from the 1st variable, the root node is branched via Algorithm 3 resulting in list of children nodes $\mathcal{N}_{i'}$ of branching level j. The children nodes are then stored in the nodes list $\mathcal{N}_i$ (Step 19 in Algorithm 2).

A node is then selected and removed from the list. Node selection process (Step 23 in Algorithm 2) is a crucial task that define the order of exploration of unexplored subproblems. The selected node is then checked if all of its integer variables are fixed, i.e $\underline{\boldsymbol{y}}^i = \bar{\boldsymbol{y}}^i$, then the node is a leaf. Only nodes are farther explored - if they verify the feasibility and optimality condition - while increasing the branching level j progressively. When the branching level j exceeds the index of the last integer variable $j > n_i$, then the first integer is re-selected $j = 1$.

The incumbent list $\mathcal{F}^*$ is updated by the Pareto fronts resulted from leaf and optimal nodes and then filtered. The process is iterated until exploring all the unexplored nodes in $\mathcal{N}$, i.e. $\mathcal{N}$ is empty.

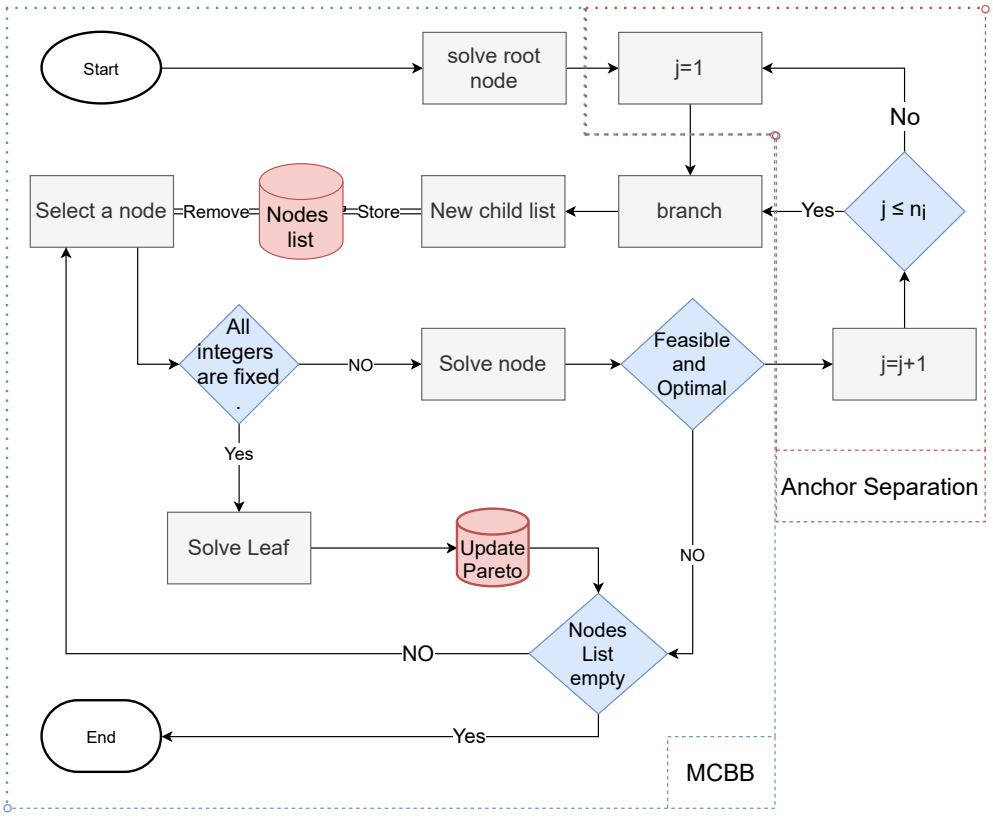


Figure 3.8: Flow chart of MCBB with BA strategy.

3.4.2 Branching by Pareto Elements

Two types of solvers can be used to bound a node in MCBB, a mono-objective solver or a multi-objective one. The mono-objective solver solves the problem for each objective independently, which results in deducing the anchor points of the node only. While the multi-objective solver returns the Pareto front of the node as in the case of BnB-NSGAII.

Branching by Pareto elements (BP) is more general than BA strategy. Though, it can be implemented in MCBB with multi-objective solver only. BP is similar to BA, yet, it separates the domain based on Pareto elements (Definition 1) instead of anchor points as shown in Figure 3.9, where PE is the number of elements in $\mathcal{F}_i^*$. Therefore, the number of nodes s' resulted from BP **in each level** is always greater than or equal to that from BA, i.e. $s \leq s'$. Since the anchor points are subset of the Pareto set. Thus, BP provides higher exploration potential *at each level*. This exploration potential is mostly desirable when MCBB is used with heuristic or metaheuristic solvers (e.g. BnB-NSGAII, Adiche & Aider [2019]; Cacchiani & D'Ambrosio [2017]). However, this does not mean that the total number of nodes of BP is higher, since all the intervals (children nodes) in both strategies are then divided to be singletons. Hence, it can not be predicted which strategy can converge with lower number of total nodes.

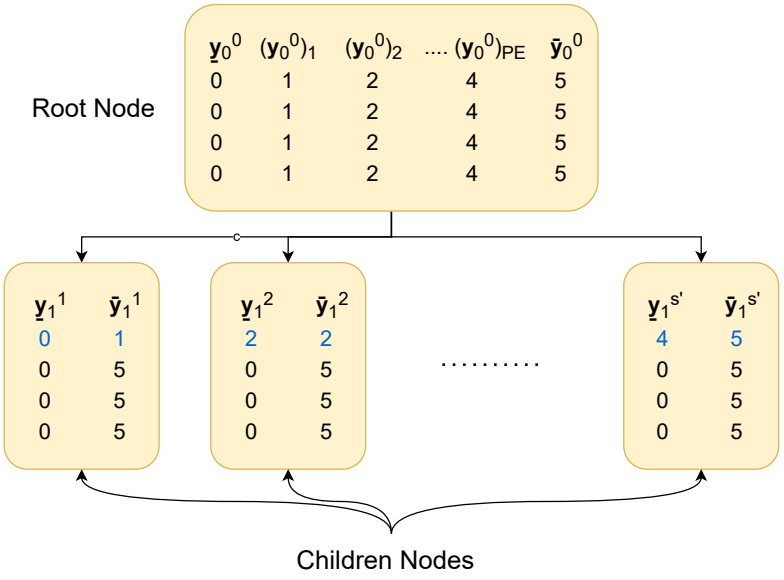


Figure 3.9: Example of BP strategy for bi-objective problem.

3.5 Constraint Handling Technique

Evolutionary Algorithms (EAs) have been significantly used to solve a wide variety of real-world problems, including in science, economics and engineering Peng *et al.* [2020]. Although EAs were originally designed to deal with unconstrained search spaces Garcia *et al.* [2017]; Peng *et al.* [2020], they have been successfully complemented by constraint-handling techniques (CHTs) to solve constrained problems. CHTs guide the search process to the feasible regions and ideally provide solutions that do not violate any constraint.

Several methods were proposed for handling constraints by EAs to solve constrained optimization problems. Penalty function methods are among the most common methods for solving COPs Cai & Wang [2006]. However, due to the well-known difficulties associated with them Coello Coello & Mezura Montes [2002], several researchers have developed a considerable amount of alternative approaches to handle constraints. These approaches were classified in Michalewicz & Schoenauer [1996] as methods that are based on a search of feasible solutions and methods that are based on preserving feasibility of solutions.

Recently, an important line of research has been arising which consists in studying ranking-based CHTS Peng *et al.* [2020]. Ranking-based CHTs sort and rank the population based on combination of both objective function value and constraints violations. Each method has a distinct mechanism for combination of these functions. However, the main issue might be how to combine into one term all values involved in the evaluation and comparison of the individuals from a given population, i.e. the objective and violation functions that may have different orders of magnitude and/or different units.

In this thesis, we propose a new ranking-based CHT based on the non-dominated sorting of the constraint violation (NSCV). NSCV is designed to handle constraints with different orders of magnitude and/or different units, without additional computational associated to the estimation of the range of violation values, and without user intervention. Moreover, NSCV is an "uncoupled" approach where the CHT is not embedded into the optimization algorithm. This allows the implementation of NSCV along with different evolutionary algorithms. NSCV affects the infeasible population only, hence, it can be implemented in mono-objective or multi-objective algorithms.

3.5.1 Literature Review

A mono-objective Constrained Optimization Problem (COP) can be written as following:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_j(\boldsymbol{x}) \leq 0, \;\; j = 1, ..., q \\ & h_j(\boldsymbol{x}) = 0, \;\; j = q+1, ..., m \\ & \boldsymbol{x} \in \boldsymbol{X}, \end{aligned} \tag{3.7}$$

where m is the total number of the constraints in the problem, $\boldsymbol{X}$ denotes the set of feasible solutions of the problem. $f(x)$, $g(x)$ and $h(x)$ are the objective function, the inequality and equality constraints respectively. They may be linear or non-linear functions and not necessary differentiable. Equality constraints are usually handled by converting them into inequality constraints with tolerance ϵ as follows,

$$g_j(\boldsymbol{x}) = |h_j(\boldsymbol{x})| - \epsilon \leq 0 \quad j = q+1, \ldots, m. \tag{3.8}$$

In the previous reviews Michalewicz & Schoenauer [1996] on CHTs embedded in EAs for searching in constrained spaces, the CHTs were classified into four categories: (1) methods based on a search of feasible solutions, (2) methods based on preserving feasibility of solutions, (3) methods based on penalty functions, and (4) other hybrid methods. Recently, an important line of research is arising which consists in studying ranking-based CHTS Garcia *et al.* [2017]. In this section, our main goal is to provide enough descriptive information for the newcomers in this area about the research that has been done and that is currently under way. This section emphasizes the way in which constraints are handled, and from each subset, the most representative work is selected.

3.5.1.1 Preserving feasibility methods

It has been distinguished between two types of methods that fall into this category in Mezura-Montes & Coello Coello [2011]:

3.5.1.1.1 Decoders decoders were one of the most competitive constraint-handling techniques in the early years of this area Mezura-Montes & Coello Coello [2011]. They are based on the idea of mapping the feasible region (Figure 3.10) onto an easier-to-sample space where a EAs can provide a better performance. In Koziel & Michalewicz [1999], the homomorphous maps (HM) is proposed, where the feasible region is mapped into an n-dimensional cube. Although decoders are an interesting constraint-handling technique from a theoretical perspective, their actual implementation is far from trivial and involves a high computational cost Mezura-Montes & Coello Coello [2011]. This has made decoders such as HM a relatively rare approach nowadays.

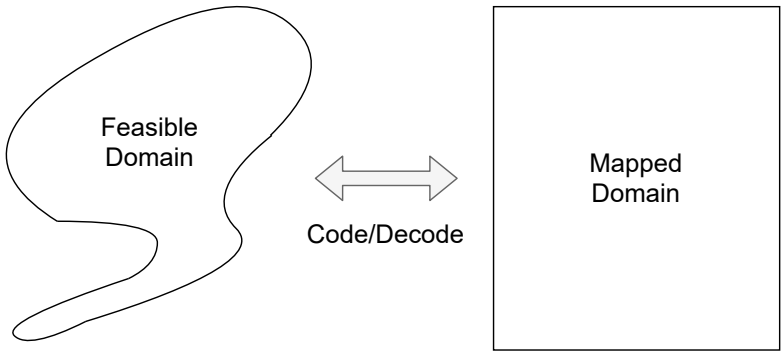


Figure 3.10: Graphical view of a general decoder [Mezura-Montes & Coello Coello, 2011].

3.5.1.1.2 Special operator the idea behind this type of methods is based on specialized operators which transform feasible individuals into other feasible individuals. In Michalewicz & Janikow [1996], the authors propose GENOCOP, which uses a variation operator which constructs linear combinations of feasible solutions to preserve their feasibility. The method assumes linear constraints only and a feasible starting point. Linear equations are used to eliminate some variables; they are replaced as a linear combination of remaining variables. Some authors have reported highly competitive results when adopting special operators Mezura-Montes & Coello Coello [2011]. However, the primary drawback of such approaches is their limited applications. Furthermore, the majority of them require an ad-hoc initialization process or at least one feasible or partially feasible solution in the initial population. This might be computationally expensive when dealing with highly constrained optimization problems Mezura-Montes & Coello Coello [2011].

3.5.1.2 Search for feasibility methods

3.5.1.2.1 Behavioral memory method This approach considers the problem constraints in a sequence; a switch from one constraint to another is made upon arrival of a sufficient number of feasible individuals in the population. It starts with a random population of individuals. Then the population is evolved until a given percentage of the population (so-called flip threshold θ) is feasible for the first constraint. The current population is the starting point for the next phase of the evolution, where the points that do not satisfy one of the first and second constraints are eliminated from the population. The process is iterated until all the points satisfy all the constraints. The drawback of this method is that it has been found to generate an important diversity loss Mezura-Montes & Coello Coello [2011].

3.5.1.2.2 repair by line search The closest feasible individual in the population is assigned to the infeasible individual to be repaired.Following the creation of the infeasible-feasible pair-

ings, a random search along the line connecting the two points is done until a second feasible point is obtained to be added into the population to replace the infeasible individual individual Simionescu *et al.* [2004].

3.5.1.2.3 repair by crossover The line search requires a number of objective function evaluations. Instead, a single crossover operation (for example, a midpoint crossover) can be done between the present infeasible and its closest feasible individual. Because the offspring who will replace the infeasible parent may be infeasible as well, the approach is more of an imperfect repair Simionescu *et al.* [2004].

3.5.1.2.4 repair by cloning The infeasible individual is substituted with an identical replica of the feasible individual nearest to it. When only one or two feasible individuals are available in the population, it may become necessary to repair only a portion of the infeasible individuals (a partial repair) to avoid standard deviation becoming too small, or to impose a lower limit on the components of the standard deviation vector in order to preserve diversity (especially for elitist algorithms) Simionescu *et al.* [2004].

The main drawback of repairing methods is that they require that at least one feasible individual exists in the current population, which may be quite difficult and/or computationally expensive when dealing with highly constrained optimization problems.

3.5.1.3 Penalty Methods

Penalty function methods are among the most common methods for solving COPs Cai & Wang [2006]. The principal idea of this kind of methods is to reformulate a constrained optimization problem as an unconstrained one by introducing a penalty term into the original objective function to penalize constraint violations Runarsson & Yao [2000]. The introduction of the penalty term r enables the transformation of a constrained optimization problem into an unconstrained one Deb [2000]:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + r \sum_{j=1}^{m} \max(0, g_j(\boldsymbol{x})), \tag{3.9}$$

where $F(\boldsymbol{x})$ is the new fitness function to be optimized. This method is called the penalty method.

A crucial consideration is determining the magnitude of the penalty term. The penalty term cannot be too high, or the algorithm will become trapped inside the feasible domain, unable to proceed towards the infeasible region's boundary. If the term is set too low, it will be irrelevant to the objective function, and the search will remain in the infeasible range. It's still uncertain and difficult to know how to leverage the search area in order to lead the search in the most desired direction. Chehouri *et al.* [2016].

3.5.1.3.1 Static Penalty In this group, the penalty factors remain constant during the evolution process and do not vary during each generation. An interesting static penalty method is proposed in Homaifar *et al.* [1994], where several levels of violation penalties are defined and then attributed to each higher level k a greater penalty coefficient r_k. However, the main downside in this approach is the necessity of a high number of parameters. They are difficult to describe and may not always be easy to obtain for real-world applications.

3.5.1.3.2 Dynamic Penalty The penalty function depends on the generation number and usually the penalty term will increase over each generation. A common dynamic penalty method is proposed in Joines & Houck [1994] based on the notion of simulated annealing. The authors propose a dynamic penalty function that is defined as follows:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + r \times g^{\alpha} \sum_{j=1}^{m} \max(0, g_j(\boldsymbol{x}))^{\beta}, \tag{3.10}$$

where g is the number of the current generation, α and β are user defined cooling parameters. Hence, the solution is extremely sensitive to the cooling parameters. An incorrect choice of the cooling parameters may lead to a local feasible solution or an infeasible solution Chehouri *et al.* [2016]. The difficulties of choosing an appropriate cooling parameters is a typical drawback of simulated annealing method Coello Coello [2000]. Another popular dynamic penalty method is the adaptive penalty method Ben Hadj-Alouane & Bean [1997]. This method decreases the penalty term if the best individuals in the last generations were feasible, and increases the penalty term, if the best individuals in the last generations were infeasible. The method introduces three additional parameters Michalewicz & Schoenauer [1996], hence, similarly to simulated annealing method, tuning these parameters is the major drawback of this method.

3.5.1.3.3 Co-Evolution In Coello Coello [2000], the authors propose to evaluate the fitness of the individual as follows:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + r_1 \times \sum_{j=1}^{m} \max(0, g_j(\boldsymbol{x})) + r_2 \times NV, \tag{3.11}$$

where NV is the number of violated constraints. According to this approach, the penalty is actually split into two values, so that the EA has enough information not only about how many constraints were violated, but also about the amounts in which such constraints were violated. However, this technique still requires the definition of two parameters and according to the authors, they must be empirically determined. A major drawback of this penalty method is that it is very subtle to variations in the parameters in addition to its rigorous definition and high computational cost Chehouri *et al.* [2016].

3.5.1.3.4 Death Penalty The "death penalty" strategy is a common and simple way to handle constraints. It simply ignores any solution that violates any of the assigned constraints Coello Coello [1999]. This strategy is simple to implement, but in most real-world problems, finding a feasible solution is a major problem. In some cases, it is difficult to find even one feasible solution Long [2014].

3.5.1.4 Ranking-Based Methods

A common major drawback of penalty methods is the difficulty of determination of penalty terms. Apart of being an arduous procedure to define and control this terms, in Chehouri *et al.* [2016], the authors argue that "such methods deviate from the essence of the philosophy of the evolutionary algorithms". This argument is based on the preservation of the objective function during the evolution process - i.e. without adding penalty function to it - which is analogous to the theory of biological evolution mimicking the principle of the survival of the fittest. On the other hand, one important line of research is recently arising which consists in studying ranking-based CHTs Garcia *et al.* [2017]. These methods sort and rank the population based on their objective and constraints violation values independently. Then each method has its own mechanism to combine these ranks.

3.5.1.4.1 Superiority of Feasible (SF) Deb [2000] SF ranks separately the solutions according to their objective function values or constraint violation values. It comprises a binary tournament selection, i.e. a pairwise comparison that basically prefers a feasible solutions over any infeasible one. The fitness value of each individual is calculated as follows:

$$F(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if feasible,} \\ \sum_{j=1}^{m} \max(0, g_j(\boldsymbol{x})) & \text{if infeasible.} \end{cases} \tag{3.12}$$

Each set of two individuals are then compared according to the following criteria: (1) a feasible solution is always preferred over an infeasible one; (2) between two feasible solutions, the one having the best objective function value is preferred; and (3) between two infeasible solutions, the one having the lowest constraint violation value is preferred.

3.5.1.4.2 Stochastic Ranking (SR) Runarsson & Yao [2000] SR intends to balance the dominance of the objective and constraint functions by evaluating each individual through a stochastic ranking procedure similar to the bubble sort, where each two consecutive individuals are compared. The comparison criterion may be based either on the objective function or the sum of constraint violations depending on a randomly determined user-defined probability parameter P_f. In other words, feasible individuals are always preferred and are sorted based on their objective values. However, if both compared individuals are infeasible, (1) a probability of P_f the

individuals are compared based on objective values; (2) a probability of $(1 - P_f)$ the individuals are compared based on the summation of the constraint values. Despite the improvement in search performance, this approach is sensitive to the parameter P_f Peng *et al.* [2020].

3.5.1.4.3 Global Competitive Ranking (GCR) Runarsson & Yao [2002] Later, the same authors of the SR method proposed another ranking-based CHT, called GCR. GCR is an advanced version of SR, however unlike SR, a solution candidate is sorted by analogizing it with not the only adjacent neighborhood but also all other members of the population. Another difference between SR and GCR is that fitness evolution is made according to both objective function and constraint violation at the rate of P_f in GCR. The fitness level of each individual in the GCR method can be evaluated as follows:

$$\begin{aligned} F(\boldsymbol{x}) = P_f \frac{rank(f(\boldsymbol{x})) - 1}{N - 1} \\ + (1 - P_f) \frac{rank(\sum_{j=1}^{m} \max(0, g_j(\boldsymbol{x}))) - 1}{N - 1}, \end{aligned} \tag{3.13}$$

where N is the number of individuals in the population, and $P_f \in [0, 0.5]$. The population is then sorted based on $F(\boldsymbol{x})$. The major drawback of GCR is that, like SR, its performance is sensitive to the parameter P_f.

3.5.1.4.4 Ho and Shimizu Ranking Ho & Shimizu [2007] In this method, the population is sorted using three ranks. The first rank R_f compares the value of the objective function $f(\boldsymbol{x})$; the second rank R_{CV} compares the squared sum of constraint violations $g_j(\boldsymbol{x})$; and the third rank R_{NV} compares the number of violated constraints. If the population contains only infeasible individuals, then the population is sorted based on R_{CV} and R_{NV} only, since the aim is to seek for the first feasible solution. Hence, the overall fitness function is defined as follows:

$$F = \begin{cases} R_f + R_{CV} + R_{NV} & \text{if} \quad N_f > 0, \\ R_{CV} + R_{NV} & \text{if} \quad N_f = 0, \end{cases} \tag{3.14}$$

where N_f is number of feasible solutions in the population. The ranks R_f, R_{CV} and R_{NV} start form 1, then, for any feasible individual $F = R_f + 2$.

3.5.1.4.5 Multiple Constraint Ranking (MCR) Peng *et al.* [2020] MCR is based on HO and Shimizu method. However, due to the different orders of magnitude and/or different units of constraints, the MCR sets a distinct rank R_c^j for each constraint $g_j(\boldsymbol{x}) \; \forall \; j \in \{1, \ldots, m\}$. Then the same mechanism of Ho and Shimizu method is applied where:

$$R_{CV} = \sum_{j=1}^{m} (R_c^j). \tag{3.15}$$

Although, both MCR and Ho and Shimizu methods were designed to prefer feasible individual over infeasible ones, in some cases the mechanism may prefer the infeasible solution. For example, if the rank R_f of a feasible individual is 10, then its fitness $F = 10 + 2 = 12$. Assume that the best objective value belongs to an infeasible individual which violates only one constraint, so for this individual $R_{NV} = 2$ and $R_f = 1$. Suppose that for the same individual, the constraint violation rank R_{CV} is 5. Hence, the fitness of the infeasible individual $F = 1 + 2 + 5 = 8 < 12$, so the infeasible individual is the winner. This may lead to the omission of the feasible solutions from the population.

3.5.1.4.6 Violation Constraint-Handling (VCH) Chehouri *et al.* [2016] VCH combines the advantages of both SF and Ho and Shimizu methods. Where in SF, the feasible individual is always preferred. In VCH, the population is split into feasible set and infeasible set. The feasible set is sorted based on R_f. While the infeasible set is sorted based on R_{NV}. Then, the overall population is sorted by pair-wise comparison, where the feasible individual is preferred over the infeasible one. If two individuals have the same R_{NV} rank, the individual with lower Constraint Violation Factor (CVF) is the winner, where:

$$\text{CVF} = \sum_{j=1}^{m} \max(0, G_j(\boldsymbol{x})), \tag{3.16}$$

$G_j(\boldsymbol{x})$ being the normalized constraint $g_j(\boldsymbol{x}) \; \forall \; j \in \{1, \ldots, m\}$. VCH eliminates the drawback of Ho and Shimizu method, i.e. the probable omission of feasible individuals. Moreover, it handles the difference of orders of magnitude of constraints by normalizing them. However, there are many real-world engineering applications for which the ranges of violation values are not available and cannot be estimated in advance Peng *et al.* [2020], therefore these cases cannot be easily dealt with by usual normalization procedures.

3.5.2 NSCV

The global-optimal solution is located at the boundary of the feasible region of a COP, if at least one of the constraints is active. In fact, this is the case in which the optimization problem is correctly formulated, i.e the formulation make sense from a engineering point of view. Moreover, the optimal solution may exist at the boundary of more than one constraint, i.e. more than one constraint are active. Thus, the number of constraints violated rank R_{NV} used in Ho and Shimizu, MCR and VCH methods is not theoretically an indicator for the best individual.

Although, the distance between the individual and the feasible boundary is a more sensible indicator, in the case of multiple constraints which may have different orders of magnitudes, merging all violations into a single sum is not numerically stable. Since the influence of constraints

with lower magnitude may become insignificant Peng *et al.* [2020]. Normalizing these values is a good approach, yet it is not always applicable (see Section 3.5.1).

The need for parameter-free CHT (unlike penalty, SR and GCR methods) in addition to the previously mentioned concerns are the main key motives to propose a new ranking-based CHT based on the non-dominated sorting of the constraint violations (NSCV). The non-dominated sorting concept is presented in this section before demonstrating the NSCV technique.

3.5.2.1 Non-Dominated Sorting

Non-Dominated sorting is a well known sorting method in multi-objective optimization field introduced firstly in Srinivas & Deb [1994]. Unlike mono-objective problems, in multi-objective problems there may not exist a one global best solution with respect to all objectives. In a typical multi-objective problem, there exist a set of solution that are better than the rest.

The main concept of non-dominated sorting is to rank the population based on an individual's non-domination described in Definition 1. The non-dominated individuals are first identified by pair-wise comparison between the current population and ranked 1 (first non-dominated front). In order to find the next non-dominated front, the first non-dominated front is temporary discounted and the above procedure is repeated. The resultant ranking is illustrated in Figure 3.11 on the projected objective domain.

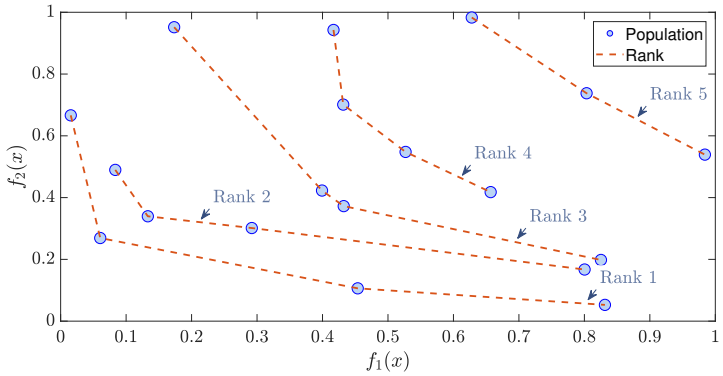


Figure 3.11: Non-dominated sorting of the population of a bi-objective problem

The idea of using multi-objective techniques to handle constraints in EAs is not entirely new. The most common approach is to redefine the single-objective optimization problem as a multi-objective optimization problem in which we will have $m + 1$ objectives Coello Coello & Mezura Montes [2002], where m is the number of constraints. Then, any multi-objective optimization technique could be applied to the new vector $\boldsymbol{F}(\boldsymbol{x}) = f(\boldsymbol{x}),\ g_1(\boldsymbol{x}),\ g_2(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})$. Surry et al. Surry *et al.* [1995] propose the use of non-dominated sorting to handle constraints. In their approach, called COMOGA, the population is ranked based on constraint violations. Then,

one section of the population is chosen based on constraint ranking, and the rest based on the individuals' objective values. COMOGA performed comparably to a penalty-based strategy in a pipe-sizing problem, however the outcomes obtained were not superior to those obtained with a penalty function. Surry *et al.* [1995]. Furthermore, COMOGA necessitates the use of numerous additional parameters, albeit the authors suggest that the technique is not overly sensitive to the values of these parameters.

In Coello Coello & Mezura Montes [2002], the authors present a proposal based on tournament selection decided through non-dominance. The fundamental idea is to use a sample individuals to identify which of two candidate solutions to choose from is the winner, and then to choose one of them based on non-dominance with respect to the samples collected. The tournament that is held is not totally deterministic. The method employs a parameter known as the selection ratio Sr, which specifies the minimal number of individuals who will not be chosen through tournament selection. A probabilistic procedure will be used to choose these individuals. This indicates that the population's $(1 - Sr)$ individuals are probabilistically chosen. Because it governs the population's diversity, the parameter Sr is critical in this strategy [**?**]. However, the authors claim that with a value closer to 1 for Sr, $0.7leqSrleq1$, the algorithm does not require detailed adjustment of this parameter.

3.5.2.2 NSCV Concept

In NSCV, the population is split into feasible and infeasible sets. The individuals in the feasible set are sorted and ranked based on their objective values. Then, each feasible individual takes a rank R_f. On the other hand, the infeasible set is treated as a multi-objective optimization problem where the constraints are considered as the objective functions. Then, the infeasible population is sorted and ranked based on the individuals' non-domination of vector $\boldsymbol{G}$, where,

$$\boldsymbol{G} = g_j(\boldsymbol{x}) \quad j = 1, \ldots, m. \tag{3.17}$$

Hence, by pair-wise comparison, each individual in the infeasible set takes a rank R_C based on its $\boldsymbol{G}$.

This vector-wise comparison between constraint violations eliminates the need of normalization where different orders of magnitudes of constraint violations exist. An illustrative example is shown in Figure 3.12 plotted on the projected constraint domain. Assuming that the problem consists in 2 constraints, the first $g_1 \in [0, 1]$ and the second $g_2(\boldsymbol{x})$ having an order of magnitude of 10^7. In this example, a direct sum of constraint violation will neglect the effect of the first constraint, since, any change in $g_1(\boldsymbol{x})$ is relatively too small to $g_2(\boldsymbol{x})$. Hence, points A$(0, 8.9999 \times 10^7)$ and C$(0.5, 9 \times 10^7)$ are considered the same, and point B$(0.7, 0.)$ is considered the best. While in NSCV, without any user-intervention and/or normalization, point A is considered better than point C, since C is dominated by A. Without any additional information, points A and B are

considered in the same rank. The latter consideration is more realistic, since none of them is absolutely better than the other. The choice of one solution over the other requires problem knowledge.

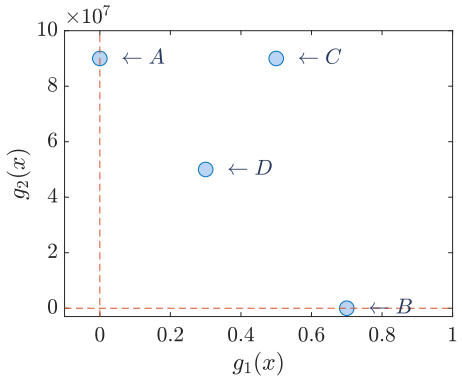


Figure 3.12: Illustrative example plotted on the projected constraint domain

In the same example, methods that rely on number of constraints violation (as NCH, MCR and Ho and Shimizu) consider points A and B are better than point D$(0.3, 5 \times 10^7)$. Since, both points A and B violates only one constraint, while point D violates 2 constraints. However, as discussed previously, number of constraint violation cannot determine the better solution without previous knowledge about the problem. In NSCV, since D is not dominated by either A or B, points A, B and D take the same rank.

To this end, NSCV promotes no solution over another in the same front (same non-dominated set). However, to increase the pressure towards a better optimal solution, we propose to sort the solutions that have the same R_C rank based on their fitness rank R_f. Therefore, NSCV algorithm can be summarized as follows. For any two solutions:

- If both solutions are feasible, the solution with better fitness rank is the winner.
- If only one of them is feasible, the feasible solution is the winner.
- If both are infeasible, the one with lower non-dominated constraint violation rank is the winner.
- If both are infeasible and in the same non-dominated front, the one with better fitness rank is the winner.

Algorithm 4 describes the pseudo-code for a fast implementation of NSCV algorithm. Where, the input is the current population and the output is the sorted population. This implementation reduces the high computational effort needed for pair-wise comparison. In step 4, the algorithm calls a fast non-dominated sorting algorithm proposed in Deb *et al.* [2002] that preserves the same concept of conventional non-dominated sorting but with less computational effort.

Algorithm 4: NSCV Algorithm

$\mathcal{N} \leftarrow$ Current Population
$\boldsymbol{G} \leftarrow$ Equation 3.17
$\boldsymbol{G} \leftarrow \max(\mathbf{0}, \boldsymbol{G})$ // $\boldsymbol{G} = \mathbf{0} \iff$ Feasible Solution
$R_C \leftarrow$ Non-Dominated Ranking of $\boldsymbol{G}$ for each individual in $\mathcal{N}$
// Feasible Solutions $\implies R_C = 1$
$R_f \leftarrow$ Rank each individual in $\mathcal{N}$ by $f(\boldsymbol{x})$
$\mathcal{N} \leftarrow$ Sort $\mathcal{N}$ by R_C
$\mathcal{Q} \leftarrow \emptyset$
foreach $r \in R_C$ **do**
    $\mathcal{T} \leftarrow$ Sort $\mathcal{N}(R_C = r)$ by R_f
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathcal{T}$
end
$\mathcal{N} \leftarrow \mathcal{Q}$

3.6 Conclusion

To this end, we have proposed a hybrid algorithm - BnB-NSGAII - that is based on the multi-criteria branch and bound - that explores the combinatorial space of the MO-MINLP problem - and NSGAII that provides the upper and lower bounds for each node in the explanatory tree. In the aim of guiding the exploration force of BnB-NSGAII, we proposed a dependency-related solving between the parent and children nodes by the legacy feature. The latter allows the children nodes to inherent the last population of the parent node then use it as an initial population. BnB-NSGAII is a general framework that is based on both MCBB and NSGAII, therefore, we have intended to enhance both algorithms which consequently enhance the performance of BnB-NSGAII. We have proposed new branching strategies to improve the performance of MCBB. Additionally, we have proposed a new constraint handling technique - NSCV - to enhance the performance of NSGAII on constrained problems. Although NSCV is proposed to be implemented in a mono-objective evolutionary algorithm, in fact, NSCV affects the infeasible individuals only. Hence, NSCV could be implemented in a mono or multi-objective EA.

In the next chapters, a numerical experiment is done for each proposed feature independently (i.e. (1) BnB-NSGAII; (2) Branching strategies; (3) NSCV) to examine its impact on the solution. First, The performance of BnB-NSGAII (only) is compared to that of NSGAII using well-known metrics from the literature. To evaluate the computational efficiency, a new metric - the Investment Ratio (IR) - is proposed that relates the quality of solution to the consumed effort. Numerical experiments are carried out on five mechanical engineering problems and two mathematical ones. These benchmark problems are in the same class (same type) of the real-world mechanical problems that are in the thesis scope.

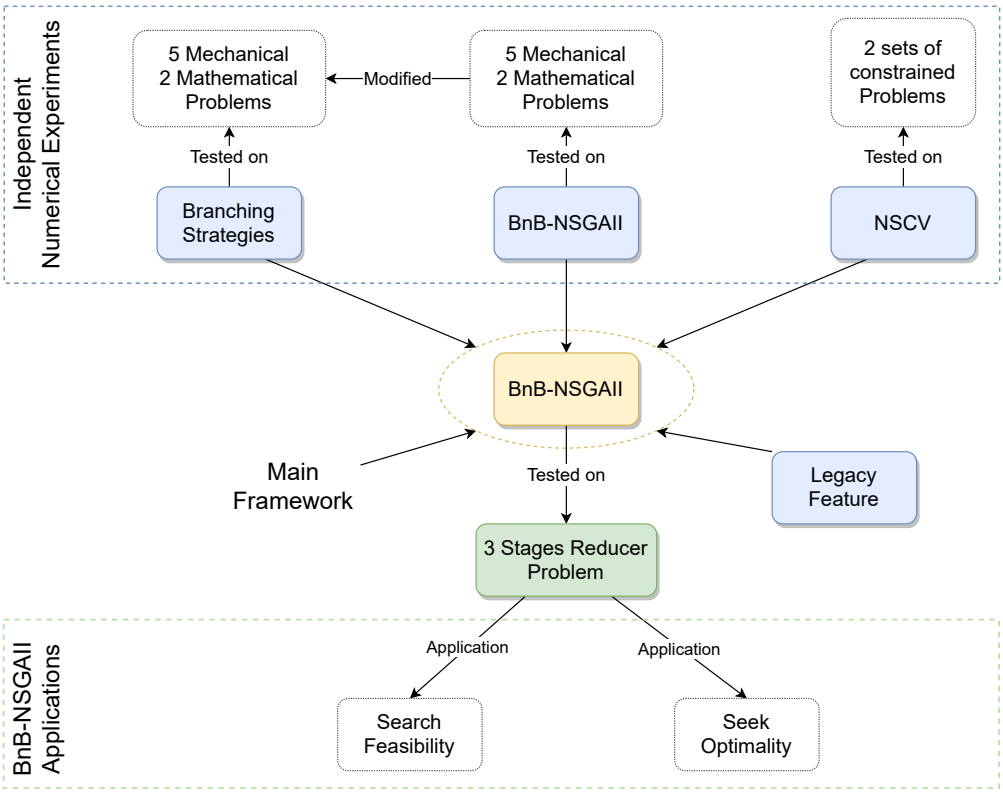


Figure 3.13: Benchmarking and applications of the proposed approach.

Second, another numerical experiment is designed to examine the performance of the proposed branching strategies with respect to the traditional branching strategy on the same (with some modifications) 5 mechanical engineering and 2 mathematical optimization problems. The experiment also studies the effect of integer variable branching order on the performance of each strategy.

Next, extensive studies are performed to assess NSCV's accuracy and robustness, compared to five other up-to-date CHTs, all implemented into the same mono-objective genetic algorithm to allow a neutral and unbiased evaluation.

Then, a computational experiment based on statistical assessment is presented to examine the overall performance of the proposed algorithms by embedding all the features [i.e. (1) BnB-NSGAII; (2) Legacy feature; (3) Branching strategies; (4) NSCV (hence NSCV is implemented in a multi-objective EA, particularly NSGAII)] into one solver (BnB-NSGAII). NSGAII and BnB-NSGAII (overall) are tested on a 3 stage reducer problem (3SR). The 3SR problem is a state-of-art problem which is a point of interest for many optimization researchers due to its complexity. In this thesis, 3SR is reformulated to a bi-objective problem with additional

constraints to meet the ISO mechanical standards. Those additional constraints increase the complexity of the problem.

In this work, two applications are proposed for BnB-NSGAII. The first application is to solve mid-sized non-convex MO-MINLP problems seeking for optimal solution(s). The second application is to use BnB-NSGAII as an initializer for another metaheuristic method (as NSGAII) to search for feasible individuals in the initial population. Figure 3.13 shows the different benchmarks done on the BnB-NSGAII approach and its applications.

Chapter 4

Numerical Experiment

Contents

4.1 Introduction **72**
4.2 BnB-NSGAII **73**
4.2.1 Test Problems 73
4.2.2 Evaluation Criteria 76
4.2.3 Benchmark Experiment 79
4.2.4 Results and discussion 80
4.3 Branching Strategies **89**
4.3.1 Results and Discussion 93
4.4 NSCV **96**
4.4.1 Profile Performance 97
4.4.2 Results and Discussion 99
4.5 Conclusion **102**

4.1 Introduction

In terms of global optimality, metaheuristic optimization methods are frequently used as a compromise between optimization effort and accuracy. Despite the fact that a wide range of optimization algorithms have been invented, finding efficient algorithms for a new instance of technologically relevant black-box optimization problem remains remarkably challenging [Sala & Müller, 2020]. Identifying which algorithms work effectively on specific problem instances or types is hence the remaining open challenge. The "per instance" or "per set" algorithm selection problem is another name for it. Empirical algorithm performance performance analysis appears to be the last resort in the lack of theoretical methodologies to determine heuristic optimization algorithm performance on specific non-trivial problems [Borenstein & Poli, 2004].

The No Free Lunch (NFL) theorems and the Law for Generalization Performance for search, optimization, and learning Schaffer [1994]; Wolpert & Macready [1997] imply that if an algorithm performs well on one set of problems it must perform correspondingly poor on all other problems. Alternatively, for the setting of optimization: the average performance of all non-resampling algorithms is identical when averaged over all possible problems. The theorems and their implications emphasize the limitations of generalizing optimization benchmark results to other problems. Although it has sometimes been argued that the assumptions of the NFL theorems are irrelevant for practical applications, because real-world problems are likely to possess some exploitable structure, the theorems also imply that if the knowledge of the exploitable problem structure is not incorporated into a particular algorithm, no formal assurances exist that the algorithm will be effective Sala & Müller [2020]. The remaining challenge for metaheuristic optimization, from a practical standpoint, is to match certain optimization algorithms with specific problems or sets of problems for which they are well suited and perform reasonably well. Theoretical algorithm performance analysis approaches for non-convex multi-modal optimization issues are severely limited. As a result, "empirical" comparison assessments must be used, such as numerical experiments on test problems or benchmark functions [**?**]. The statistical measurement of expected algorithm performance on a certain range of problems is thus required to explore generalization.

In this chapter, each proposed feature (i.e. BnB-NSGAII, branching strategies and NSCV) is examined independently to monitor its impact on the solution solely. Each experiment is done a on set of problems to provide a statistical assessment for comprehensive results analysis. The evaluation criteria vary from experiment to another depending on how the feature impacts the solution. Hence, in each of the following sections, the utilized set of test problems and evaluation metrics are defined in addition to the benchmarking scenario. Results are illustrated and analysed in each section independently. The overall performance of all features is examined in Chapter 5 on a real-world problem.

4.2 BnB-NSGAII

To evaluate the performance of BnB-NSGAII algorithm, we present a benchmark experiment based on a statistical assessment to compare its performance to that of NSGAII. Both are tested on seven MO-MINLP problems from the literature. The true Pareto solutions of these problems are known, so the Pareto fronts resulting from both algorithms can be compared to the true ones. We present evaluation criteria from literature for quantitative comparison of the performances.

4.2.1 Test Problems

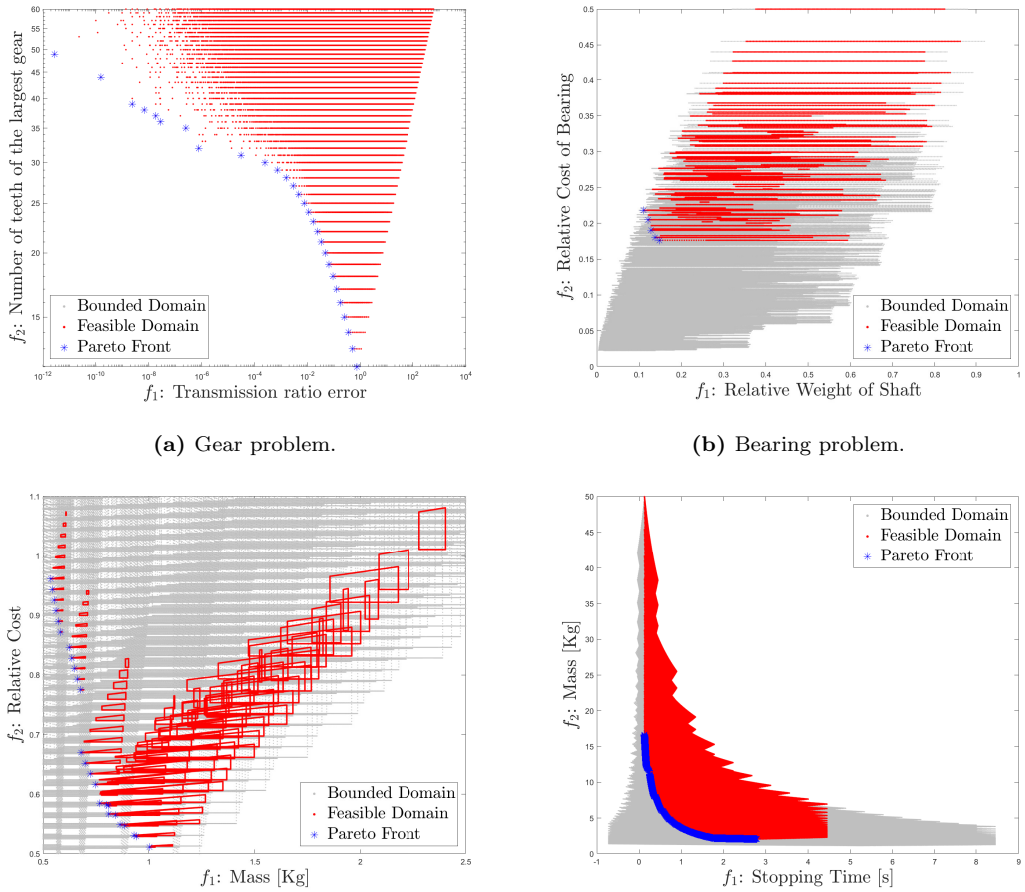
To demonstrate the generality of the algorithm, the presented test problems are mixed integer/binary/discrete constrained/unconstrained problems having different Pareto shapes, discrete, continuous or discontinuous. In particular, five of the selected problems are mechanical optimization problems and two of them are mathematical ones. The formulations of these problems are declared in the Appendix A. Although the proposed algorithm makes no assumptions on the number of objective functions, we select bi-objective problems for more comprehensive results, i.e. the Pareto front can be illustrated on a 2-dimensional objective domain.

Figure 4.1 illustrates the exact representation of the seven problems presented in this section. For each problem, the overall domain projection on the objective space is illustrated in grey, while the feasible one in red. The true Pareto for each problem is plotted in blue. The method used to obtain the true Pareto solution is provided in Appendix A. Table 4.1 summarizes various properties of the presented problems.

The gear train problem proposed in Deb *et al.* [2000], represents a compound gear train which is to be designed to achieve a specific gear ratio between the driver and driven shafts. The objective of the gear train design is to find the number of teeth in each of the four gears to minimize (i) the error between the obtained gear ratio and a required gear ratio of and (ii) the maximum size of any of the four gears. The solution of this unconstrained pure integer problem is illustrated in Figure 4.1a showing that its Pareto front is discrete.

The ball bearing pivot link was initially proposed in Giraud & Lafon [1999] as a mono-objective problem, then reconstructed in El Samrout [2019] to a bi-objective one. In this problem, the aim is to minimize the relative mass of the system composed of a shaft and two ball bearings, and relative cost of bearings. Figure 4.1b shows the complexity of the domain. It also shows that the Pareto front of this problem is discrete.

The coupling with bolted rim proposed in Giraud-Moreau & Lafon [2002] for torque transmission was reconstructed in El Samrout [2019] to a bi-objective problem for which goal is to dimension the mechanism in a way that minimizes its cost and weight. Figure 4.1c shows a part of the bounded domain to focus on the feasible domain that is represented as patches of solutions. The Pareto solution of this problem is a discrete one. In the disk brake design problem, reported



(a) Gear problem.

(b) Bearing problem.

(c) Coupling problem.

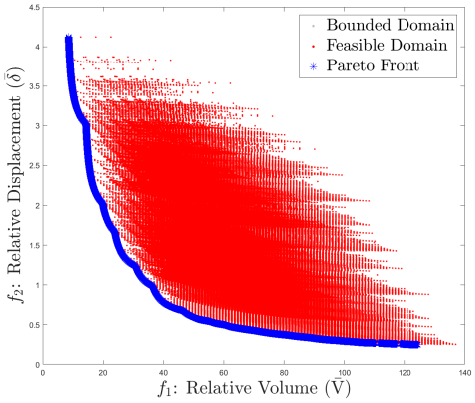
(d) Brake problem.

Figure 4.1: The attainable set of solutions of the test problems.

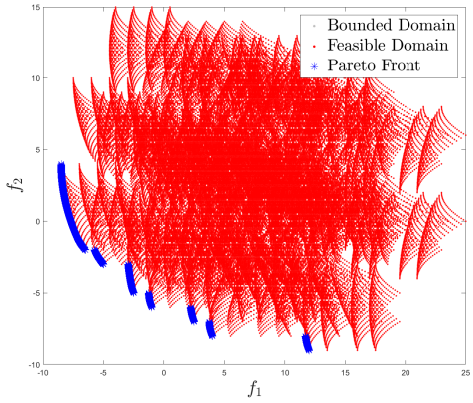
in Osyczka & Kundu [1995]; Tong *et al.* [2016], there are two design objectives: minimize the mass of the brake and minimize the stopping time. The non-convex discontinuous Pareto front is observed in figure 4.1d.

Next, bi-criterion optimization of the nine bar truss was considered in Mela *et al.* [2007]. The goal is to minimize the material volume V and the vertical displacement δ of the loaded node simultaneously. It is generally known that material volume and flexibility are strongly conflicting quantities in structural design. Figure 4.1e shows the minimal points of all the components of the problem. The convex minimal curves of the components intersect each other, resulting in a highly non-convex minimal curve for problem.

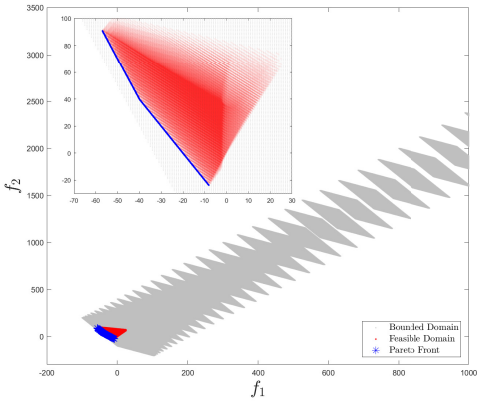
Mela problem is a mathematical problem proposed in Mela *et al.* [2007]. It is unconstrained and it is nearly linear, the quadratic term being the only nonlinear part of the problem. Figure



(e) Truss problem.



(f) Mela problem.



(g) Tong problem.

Figure 4.1: The attainable set of solutions of the test problems.

4.1f shows that even a mildly nonlinear problem can have a very complicated attainable set, there are many locally minimal points. The set of minimal points is not connected, but consists of subsets of minimal points of seven components.

The analytical problem, which is adapted from Dimkou & Papalexandri [1998]; Tong *et al.* [2016], is referred to as Tong problem. In this problem, the continuous variables are unbounded. However their bound can be deduced analytically from the constraints. To examine the exploration force of the optimization method, the bounds are set to generate a large-scale domain as shown in figure 4.1g.

Table 4.1: Test problems properties

Problem	# Variables	# Integer Vars	Var Type	# Constraints	Pareto Type
Gear	4	4	Integer	0	Discrete
Bearing	4	2	Discrete	10	Discrete
Coupling	4	2	Discrete	4	Discrete
Brake	4	1	Integer	5	Discontinuous
Truss	9	6	Discrete	0	Discontinuous
Mela	10	8	Binary	0	Discontinuous
Tong	6	3	Binary	9	Continuous

4.2.2 Evaluation Criteria

The evaluation of the optimization method is a necessary task that should be conducted properly. In this section, we present three ways to evaluate a multi-objective solver; namely accuracy metrics, computational effort and robustness.

Accuracy metrics are a sort of performance indicators that try to assess the quality of solutions in terms of precision. Several classifications of metrics can be found in the literature including convergence-based metrics, diversity-based indicators Audet *et al.* [2018] and others. In this section, true Pareto front based metric will be used, since the true Pareto solutions of the proposed benchmark problems are known.

4.2.2.1 Cardinality Indicators

These metrics focus on the number of non-dominated points generated by a given algorithm. Cardinality indicators were reviewed in Audet *et al.* [2018], among them is the overall non-dominated vector generation (ONVG). ONVG is the number of the Pareto front approximation generated by the algorithm:

$$\text{ONVG}(S) = \mid S \mid \tag{4.1}$$

Where $\mid S \mid$ is the number of points of the approximation Pareto set. Another indicator called the purity metric is found in Custódio *et al.* [2011]. It is used when comparing two or more approximate methods in the absence of the knowledge of the true Pareto. This indicator is slightly modified to involve the known true Pareto front. Let $\mid P \mid$ be the number of elements in the true Pareto set and $\mid F \mid = \mid S \mid \cup \mid P \mid$, and then removing from $\mid F \mid$ any dominated points. The modified purity metric is then given by:

$$\text{Purity}(S) = \frac{\mid S \mid \cap \mid F \mid}{\mid S \mid} \tag{4.2}$$

Therefore, this value represents the ratio of non-dominated points over the solution set. This metric is thus represented by a number between zero and one. Higher values indicate a better Pareto front in terms of the percentage of non-dominated points.

4.2.2.2 Convergence Metrics

Generational Distance (GD) Van Veldhuizen & Lamont [2000] and Inverted Generational Distance (IGD) Li & Zhang [2009] are well-known metrics with regard to convergence metrics. They compute the average distance from the approximated Pareto to the true one. For bi-objective problem, GD is defined as follows:

$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{|S|} d_i^2}}{\mid S \mid} \tag{4.3}$$

Where d_i is the euclidean distance in objective function space between each approximated solution to the nearest true solution. A value of $GD = 0$ indicates that all the approximated solutions are on the true Pareto front. IGD substitute $\mid S \mid$ by $\mid P \mid$.

In the case of discrete Pareto, IGD indicator is used, while for continuous/discontinuous Pareto, GD indicator is used. Effect of the method used to obtain the true Pareto on this metric is discussed in Appendix A.

4.2.2.3 Distribution and spread indicators

The spread (Δ) is another accuracy metric criteria introduced in Deb [2001]. It includes information about both uniformity and spread. The formulation of this metric for a bi-objective problem is:

$$\Delta = \frac{d_f + d_l + \sum_{n=1}^{n-1} |d_i - \bar{d}|}{d_f + d_l + (n-1)\bar{d}} \tag{4.4}$$

where d_f and d_l are the euclidean distances between the extreme solutions in the non dominated set obtained using the algorithm and the real Pareto front, n is the number of approximated solutions $n = \mid S \mid$, d_i is the euclidean distance between two consecutive solutions and $\bar{d}$ is the average of all distances d_i, $i \in [1, n-1]$. The lower the spread indicator is, the better the solution is. Nevertheless, this indicator will never be zero, unless the spread of the true Pareto Δ_{True} is equal to zero and the solver succeeded to find it. Thus, for more comprehensible evaluation, relative spread formulated in equation 4.5 is used,

$$d\Delta = \mid \Delta_{\text{True}} - \Delta_{\text{Approximate}} \mid . \tag{4.5}$$

To calculate Δ_{True}, d_f and d_l are considered 0 by their definition. And the true Pareto is discretized uniformly (see Appendix A). Therefore, Δ_{True} (Equation 4.6) is not necessarily to be zero.

$$\Delta_{\text{True}} = \frac{\sum_{n=1}^{n-1} |d_i - \bar{d}|}{(n-1)\bar{d}} \tag{4.6}$$

4.2.2.4 Computational Effort

The quantification of computational effort used by the algorithm to produce the optimal Pareto front is a good way to assess its effectiveness. In Talbi [2009], the author considers the number of objective function evaluations is a good representation of the computational effort .

4.2.2.5 Robustness

The robustness of the algorithm should also be measured. This is done by running it several times to compensate for the impact of random parameters, which may cause the fluctuation of solution results. The lower the variability of the obtained solutions the better the robustness Montgomery [2017].

4.2.2.6 Investment Ratio

The necessity of more accurate solutions with the toleration of more computational effort, a new metric that measures the potential improvement of the solution concerning computational cost is defined. In our case, the quality is the accuracy of the solution, while the cost is the computational effort. In the purpose of relating the quality and the cost in a single indicator that can represent how much the quality increased with respect to the cost, we propose the investment ratio indicator. Let Q_1 and Q_2 be the quality indicators resulting from 2 solvers being compared, and C_1, C_2 be their cost. Let $q = Q_2/Q_1$ be quality ratio and $c = C_2/C_1$ be the computational cost ratio. The proposed Investment Ratio indicator (IR) is designed to regard various cases and is defined in equation 4.7.

$$\mathrm{IR}(q,c) = \begin{cases} \frac{q}{c} & \text{if } q \geq 1 \\ -\frac{c}{q} & \text{if } q < 1 \end{cases} \tag{4.7}$$

The positive value of IR indicates that the quality is enhanced regardless of the cost ratio and vice versa. IR ≥ 1 indicates a good investment, as $Q_2 > Q_1$ by improvement ratio higher than the cost one. For IR $\in]0, 1[$, this means that although the cost ratio is higher than the quality ratio, still the quality is enhanced. Here, the investment cannot be assessed obviously since it depends on the weights of the desire of the investor for a certain problem. IR $= -1$ indicates that the quality is reduced by the same cost reduction ratio, which is also an acceptable investment. Next, IR $\in]-1, 0[$ means that solver 2 obtained lower quality with lower computational effort, yet the cost is reduced by a higher ratio than the quality degradation ratio, this value also might be acceptable depends on the investor desire. Finally, IR < -1 indicates a bad investment in the absence of profitable improvement in either cost or quality.

In this experiment, GD and Δ are used as quality indicators, but since equation 4.7 assumes that the higher values of q mean higher quality, q is defined as the geometric mean between the

inverse of the spread and GD as stated in equation 4.8. In particular, the mean value of the metrics is used,

$$q = \sqrt{\left(\frac{(GD_{\text{NSGA}})_{\text{mean}}}{(GD_{\text{BnB}})_{\text{mean}}} . \frac{(\Delta_{\text{NSGA}})_{\text{mean}}}{(\Delta_{\text{BnB}})_{\text{mean}}} \right)} \tag{4.8}$$

4.2.3 Benchmark Experiment

Both NSGAII and BnB-NSGAII performances are affected by selecting different values of tuning parameters. In the purpose of impartial comparison between them, different tuning parameters are applied to each method on each of the 7 problems independently. To examine the robustness of the results, each problem is solved 20 times by each method for each combination.

In this experiment, the effect of tuning the parameters on NSGAII and BnB-NSGAII are tested considerably, mainly for population size and the number of generations for their direct effect on both quality of the solution and the computational effort consumed. It should be noted that the number of generations is affected by two parameters. First, the maximum allowable generations which terminate NSGAII process even if the optimality condition is not attained. Second, the process is terminated if the optimality condition didn't change over m_g generations, m_g being the maximum stall generations value to be tuned. This test is done by varying these parameters iteratively seeking for the best performance of NSGAII and BnB-NSGAII. The test considers the three tuning levels of BnB-NSGAII mentioned previously. Table 4.2 shows the values of distinct parameters used for both algorithms.

Best to best comparison is commonly used to compare the performances of distinct solvers. However, the best solution may be an outlier. For that, the presented benchmark is based on the distribution of values for each metric over the 20 iterations.

Table 4.2: Parameters used for NSGAII and BnB-NSGAII algorithms

Parameters	**Value**
Cross over probability	0.9
Mutation Probability	0.95
Population size	variable: 20 $\rightarrow$ 1000
Allowable generations	variable: 20 $\rightarrow$ 1000
Stall generations	variable: 20 $\rightarrow$ Allowable generations
Constraint handling	Legacy method Deb *et al.* [2002]
Crossover operator	Simulated Binary crossover (SBX) Maruyama & Tatsukawa [2017]
ETAC	100
Mutation operator	Partially-mapped crossover (PMX) Maruyama & Tatsukawa [2017]
ETAM	10

4.2.4 Results and discussion

Figures 4.2 to 4.8 show the distribution of GD, Spread, Purity, and the number of evaluations over the 20 iterations for each combination of parameters, the latter being represented as parameter ID. It should be noted that an outlier is detected when a value is more than 1.5 times the interquartile range away from the top or bottom of the box.

The tuning of the parameters is done manually per each problem for each solver. Hence, the number of combinations of parameters are limited and *not systematic* (i.e. the number varies from problem to another and from solver to another depending on how the author observed the variation of the performance with respect to the tuning). All the tested combinations of parameters for BnB-NSGAII and NSGAII are listed in Appendix C with their IDs.

As expected, Figures 4.2 to 4.8 show that parameters tuning has significant effect on the performance of NSGAII and BnB-NSGAII. Hence, in the purpose of comprehensive discussion, best performance of NSGAII in terms of solution convergence and uniformity is selected for each problem. While for BnB-NSGAII, parameters resulting in 1) most accurate results and 2) best compromise between solution quality and computational effort (i.e best investment) are selected to be discussed. Tables 4.3 and 4.4 show the selected parameters and their ID numbers for NSGAII and BnB-NSGAII respectively.

Table 4.3: Selected parameters for NSGAII

Problem	ID	Population size	Allowable generations	Stall generations
Gear	1	1000	1000	1000
Bearing	1	1000	1000	1000
Coupling	1	1000	1000	1000
Brake	2	1000	1000	1000
Truss	1	1000	1000	1000
Mela	1	1000	1000	1000
Tong	6	1000	1000	1000

Compared to NSGAII, BnB-NSGAII provides better performance in terms of GD and spread indicators for all the problems except for truss problem which BnB-NSGAII has higher spread values. Regarding purity indicator, BnB-NSGAII overcomes NSGAII for all the problems except for Mela problem which has slightly lower ratio. For gear problem, Figure 4.2 shows that BnB-NSGAII, ID = 18, is able to find the exact solution of this problem robustly. It is also observed that BnB-NSGAII succeeds to find a better solution with lower computational effort for coupling, brake, Mela and Tong problems in Figures 4.4, 4.5, 4.7, 4.8 for ID $= 25, 12, 10, 10$ respectively.

To specifically explore the computational efficiency of the BnB-NSGAII algorithm, investment ratio is calculated for each of the selected combinations. Table 4.5 shows the mean values of GD,

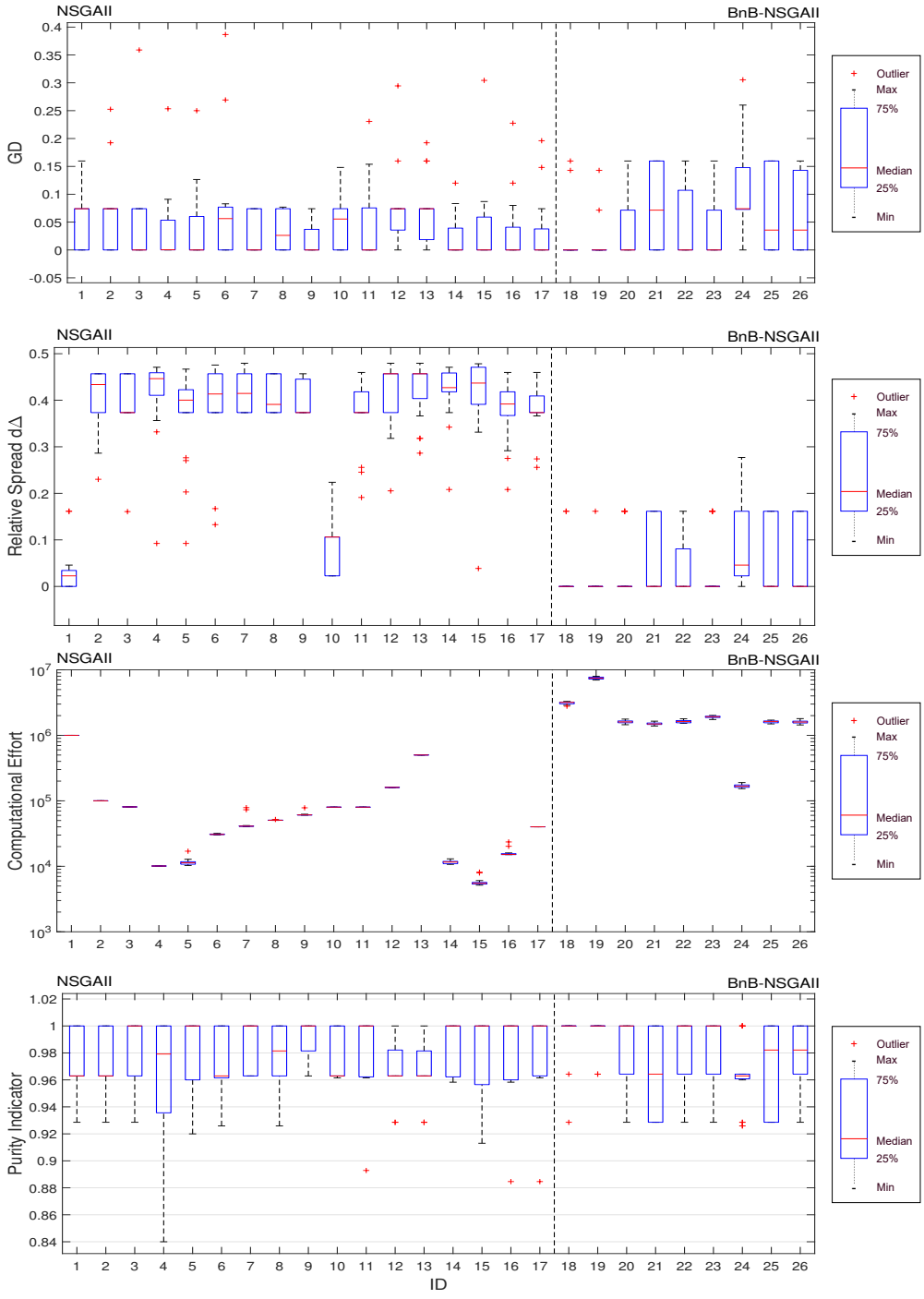


Figure 4.2: Distribution of metrics for gear problem.

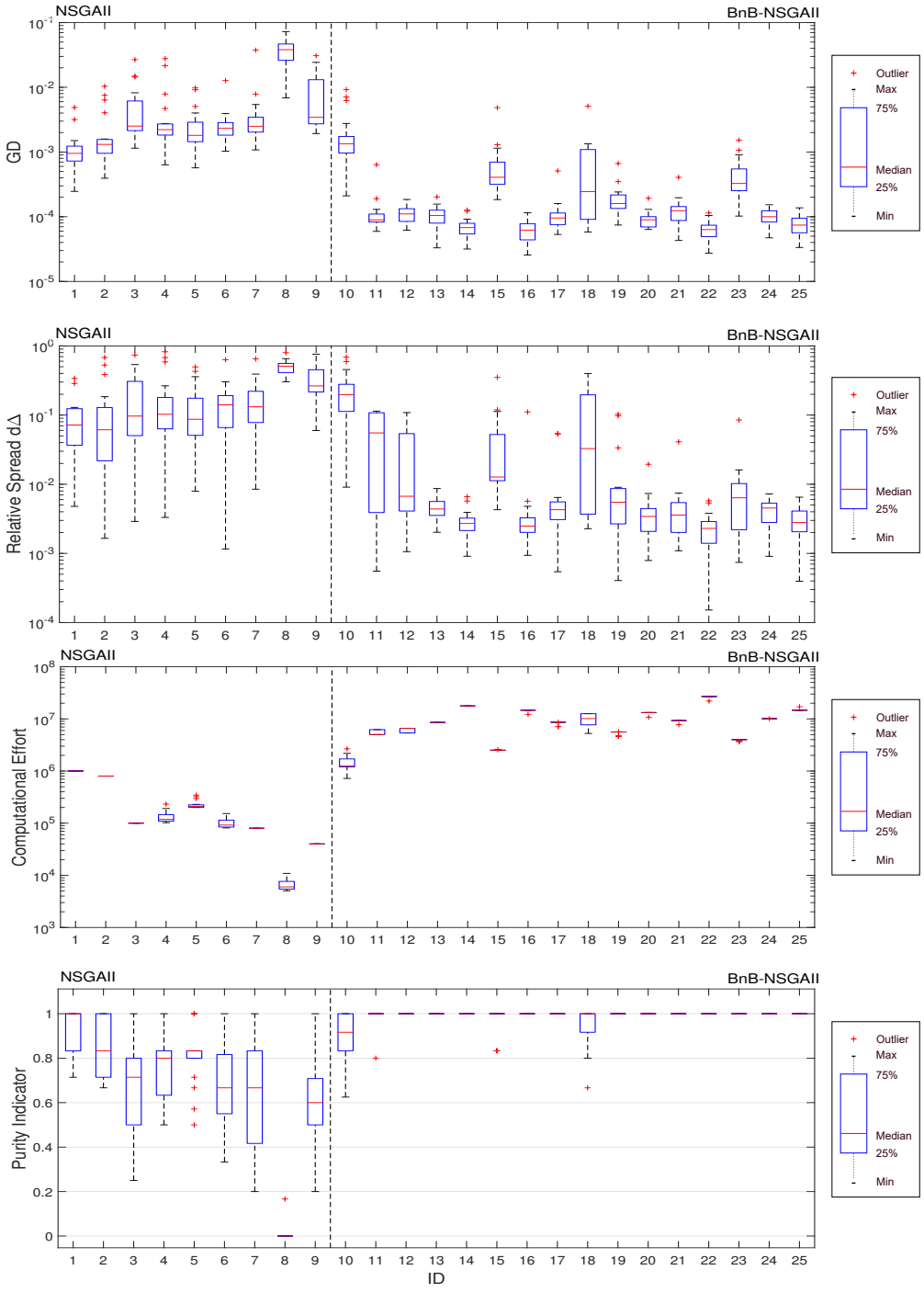


Figure 4.3: Distribution of metrics for bearing problem.

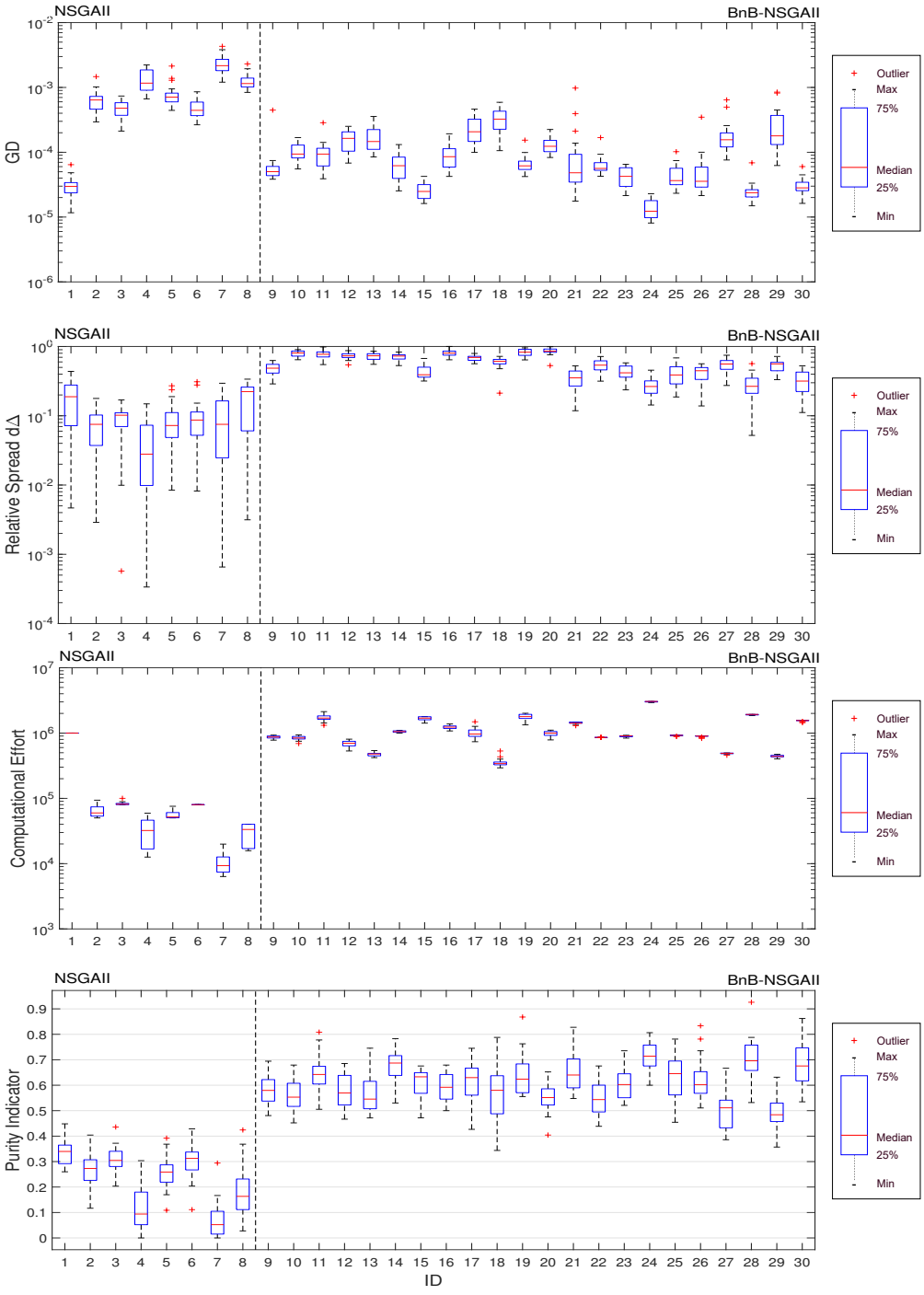


Figure 4.4: Distribution of metrics for coupling problem.

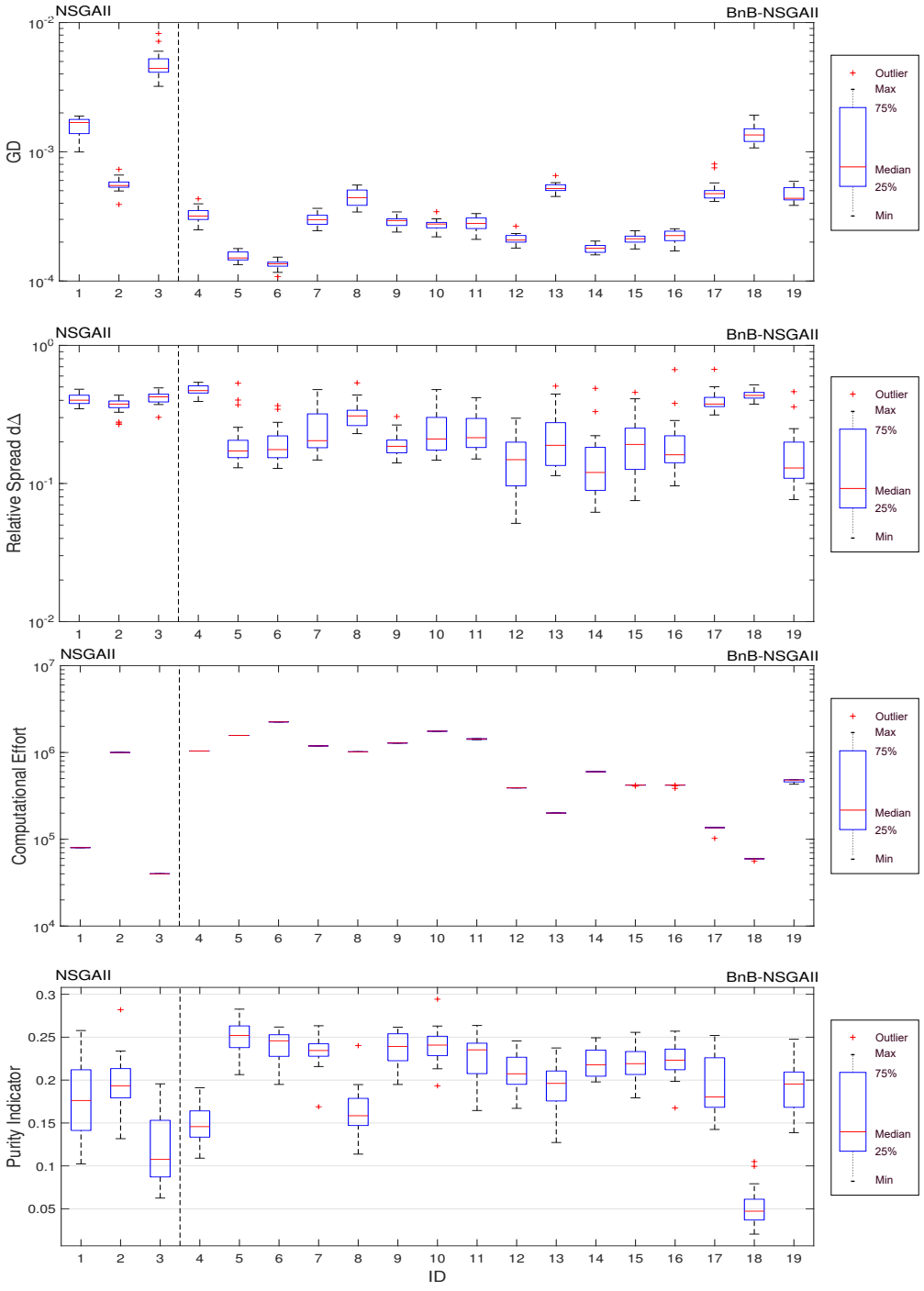


Figure 4.5: Distribution of metrics for disk brake problem.

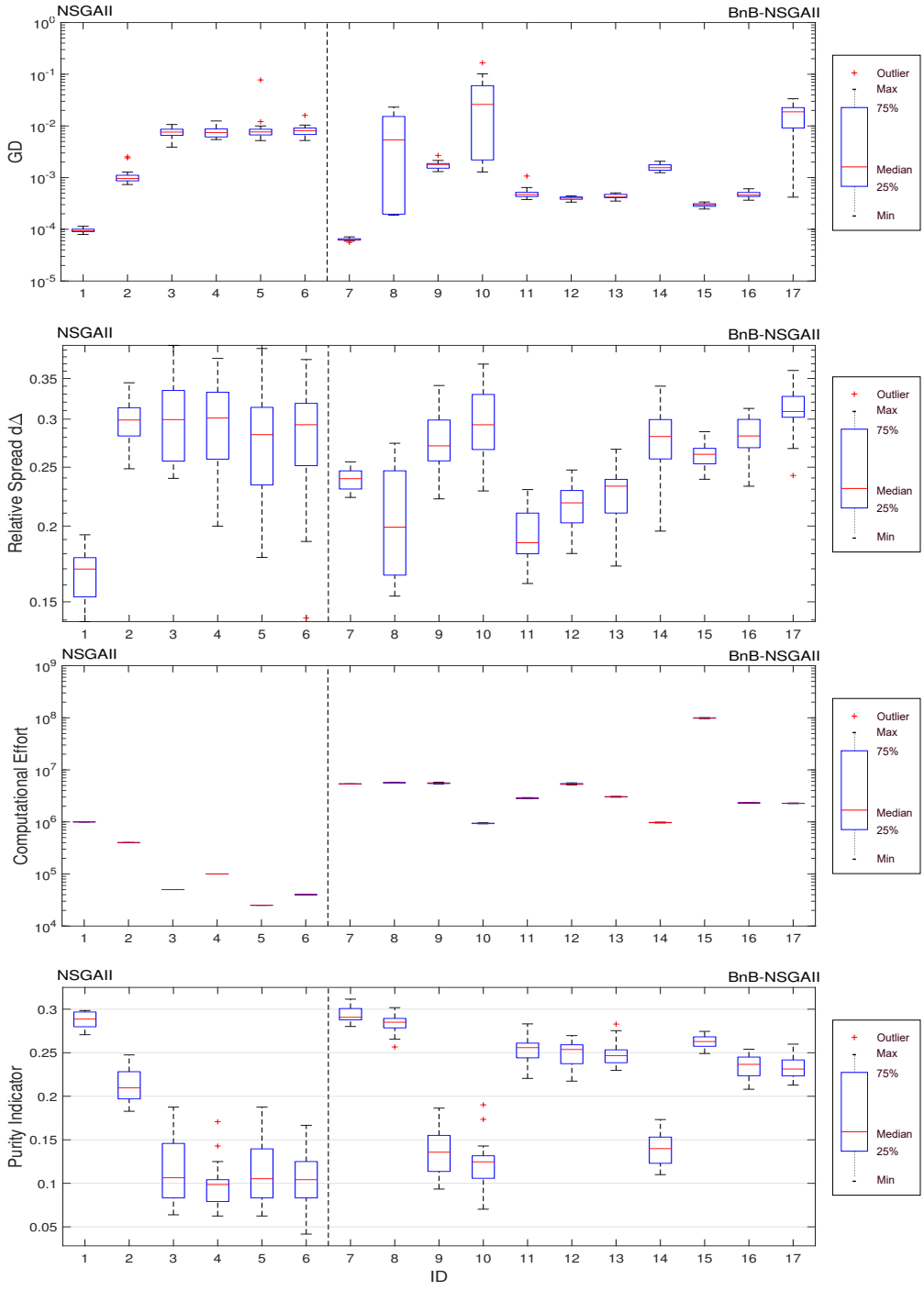


Figure 4.6: Distribution of metrics for truss problem.

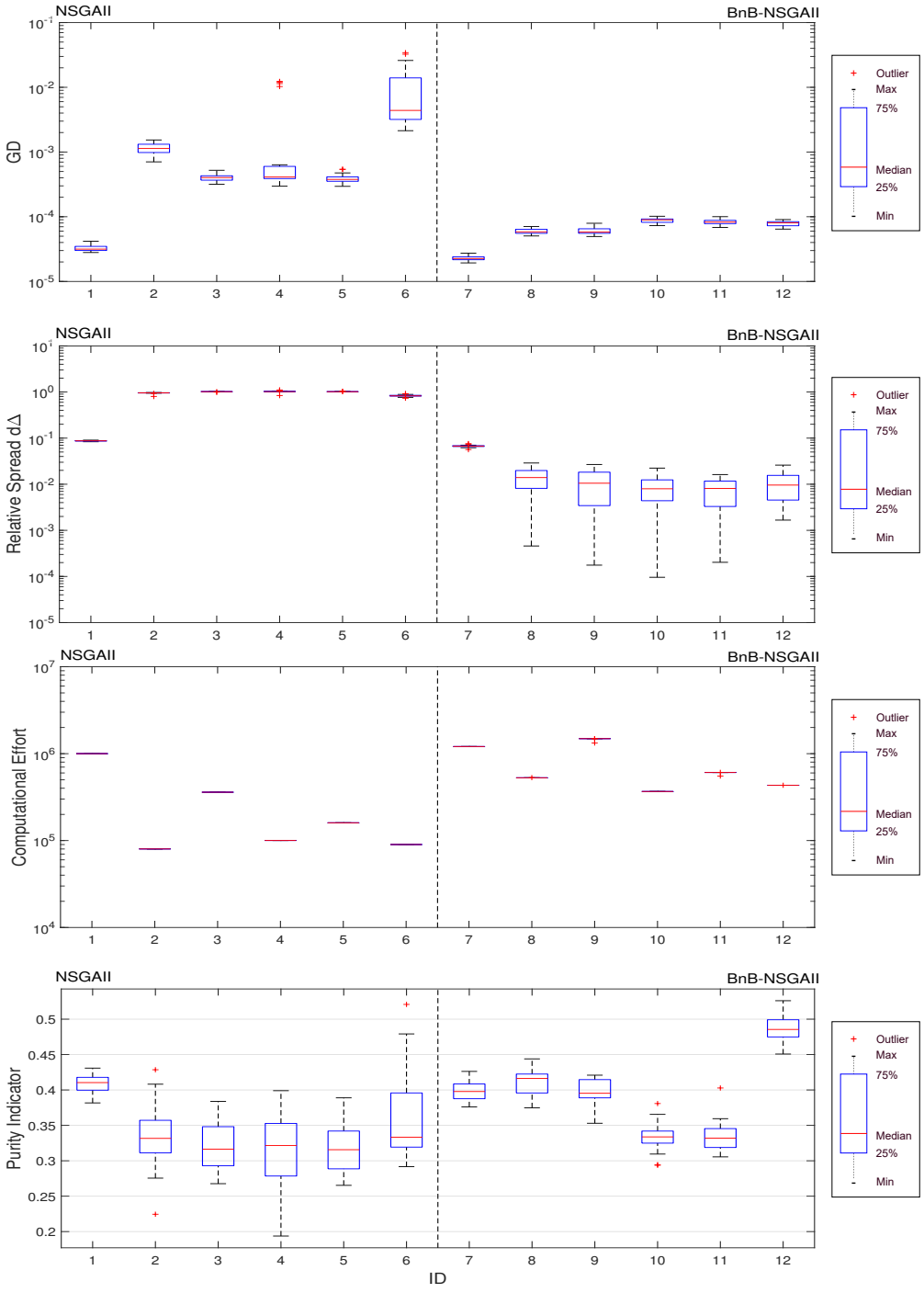


Figure 4.7: Distribution of metrics for Mela problem.

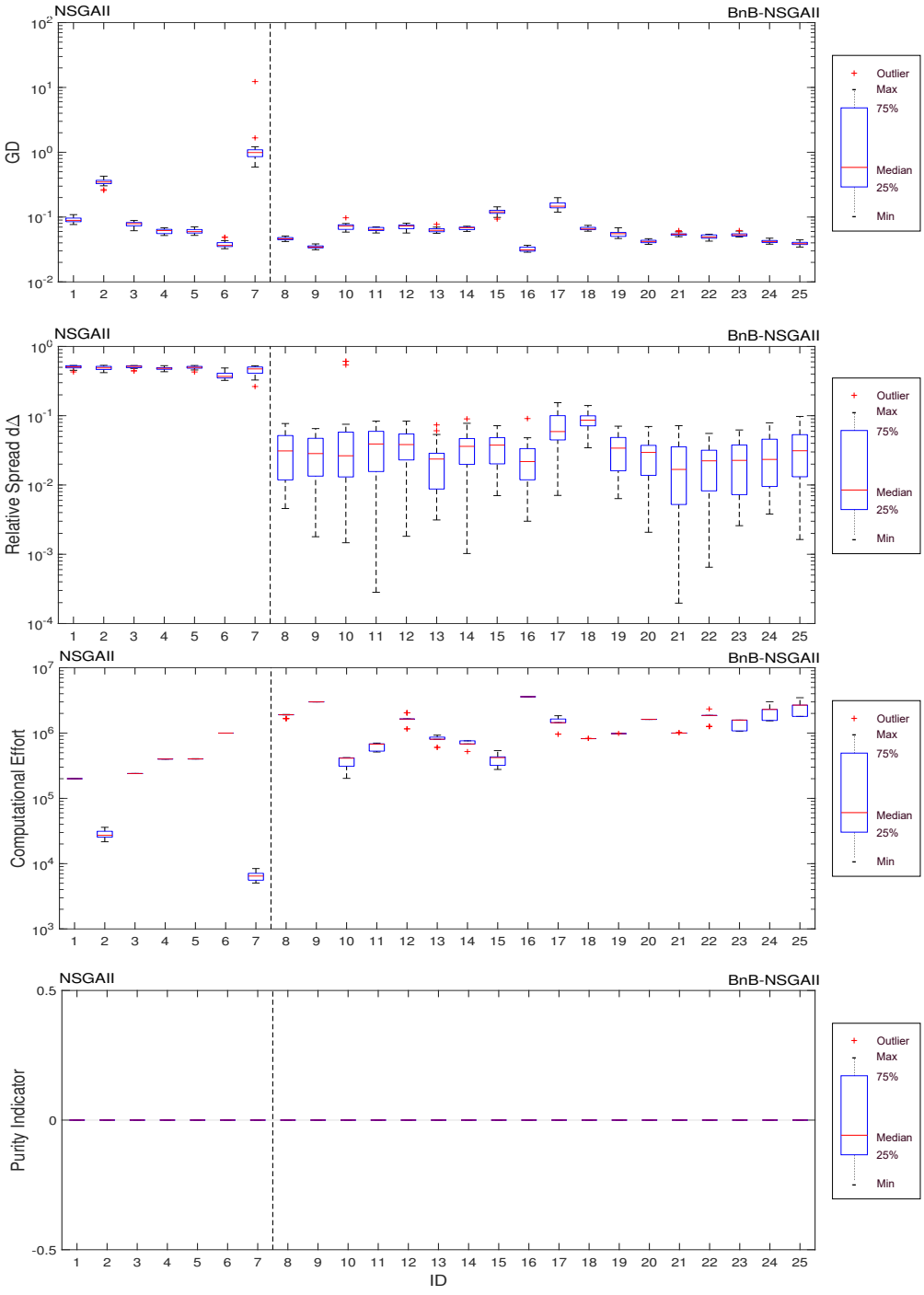


Figure 4.8: Distribution of metrics for Tong problem.

Table 4.4: Selected parameters for BnB-NSGAII

Problem	ID	Root			Node			Leaf		
		Pop	*Gen*	*Stall*	*Pop*	*Gen*	*Stall*	*Pop*	*Gen*	*Stall*
Gear	20	100	800	500	50	100	20	50	20	20
	18	100	800	500	100	50	20	50	20	20
Bearing	19	50	800	800	50	800	300	50	800	300
	22	50	800	800	50	800	800	100	800	800
Coupling	25	10	100	100	10	150	100	10	800	200
	24	10	100	100	10	100	100	10	800	800
Brake	12	50	200	200	50	200	200	100	200	200
	6	1000	1000	1000	50	20	20	100	800	500
Truss	7	1000	1000	1000	100	20	20	100	20	20
Mela	10	200	800	100	50	20	20	50	200	200
	7	1000	1000	1000	50	20	20	50	200	200
Tong	10	200	1000	1000	50	20	20	100	800	500
	16	200	1000	1000	200	1000	1000	200	1000	1000

relative spread dΔ, number of solutions in the approximated Pareto front $\mid S \mid$, purity ratio and computational effort after removing the outlier points, in addition to the investment ratio indicator. It is observed that BnB-NSGAII is able to provide a "good investment" (IR > 1) for all problems except for truss problem. Therefore, the effectiveness of BnB-NSGAII is justified for 6 out of 7 problems. IR $= \infty$ in gear problem (see Table 4.5) means that BnB-NSGAII is able to find a solution with best quality, i.e. the true solution.

For continuous/discontinuous Pareto problems, the maximum number of Pareto elements could be obtained by NSGAII is limited by the population size. In Table 4.5, it is shown that more elements are found by BnB-NSGAII ($\mid S \mid_{\text{BnB-NSGAII}} > \mid S \mid_{\text{NSGAII}}$) since in BnB-NSGAII the number of elements is limited by (population size $*$ number of leaf nodes). Figure 4.9 shows the best solution obtained by both methods compared to the true Pareto front for each problem. As shown, BnB-NSGAII is able to approximately converge to the true Pareto front for all the problems. While NSGAII failed to catch all the Pareto elements for gear and bearing problems as shown in Figures 4.9a and 4.9b respectively. On a detailed examination of the Pareto front of Tong problem Figure 4.9g, it is found that NSGAII does not capture the section of the Pareto front close to the anchor points. Finally, the results generally verify the competitive performance of BnB-NSGAII with respect to NSGAII in terms of the quality of the solution.

Table 4.5: Mean values of GD, relative spread (dΔ), number of solutions ($\mid S \mid$) and number of evaluations in addition to IR for the selected combinations ID. (Better results are **emphasized**)

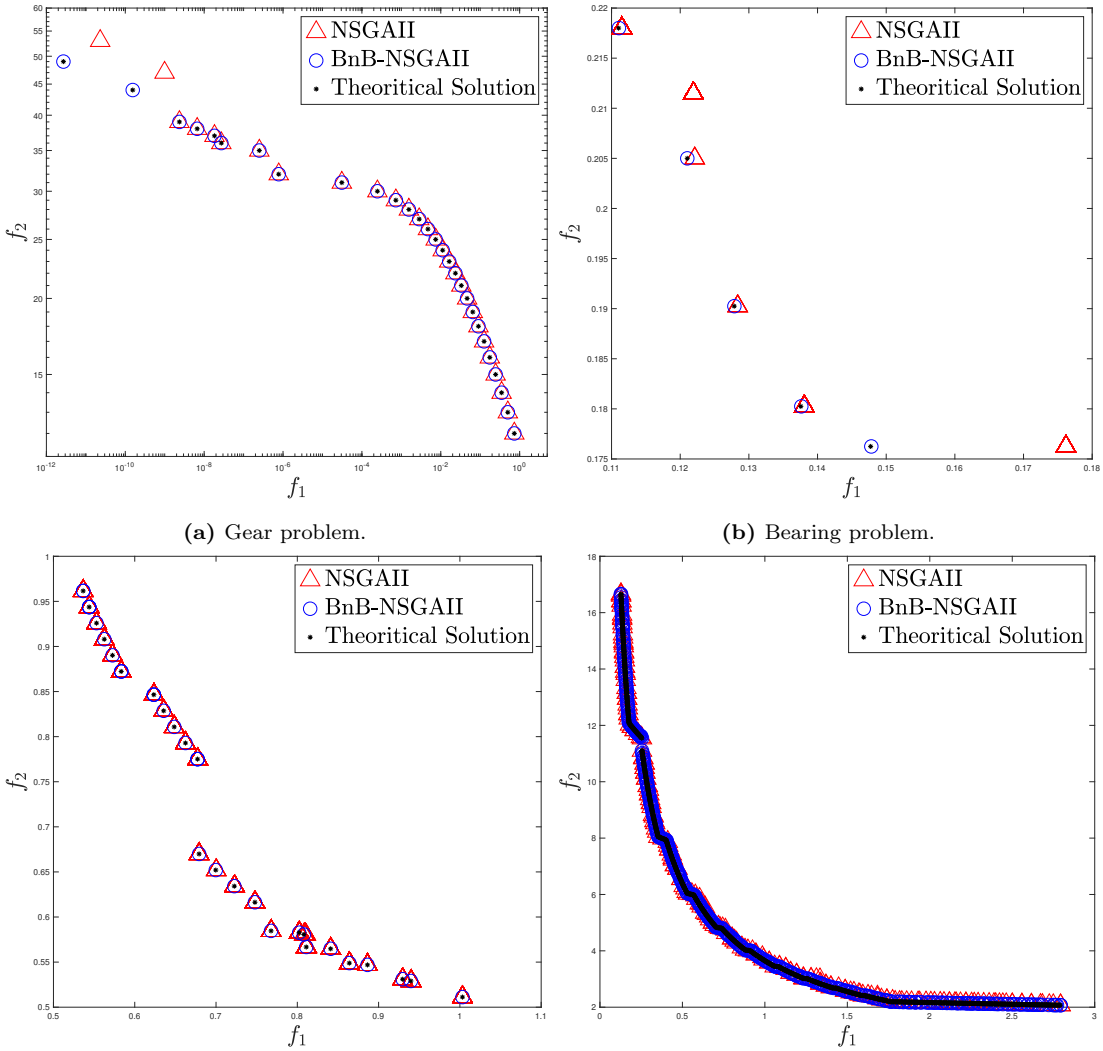
Problem	Method	ID	GD	dΔ	$\mid S \mid$	Purity %	#Evals 1×10^5	IR
Gear	NSGAII	1	5.6×10^{-2}	1.9×10^{-2}	27	97	10	1
	BnB-NSGAII	20	3.6×10^{-2}	0	28	99	16	24.5
		18	**0**	**0**	**28**	**100**	**31.2**	∞
Bearing	NSGAII	1	8.8×10^{-4}	6.7×10^{-2}	5	92	10	1
	BnB-NSGAII	**19**	1.6×10^{-4}	5.0×10^{-3}	**5**	**100**	**5.4**	**1.59**
		22	6.2×10^{-5}	2.0×10^{-3}	5	100	26.6	0.82
Coupling	NSGAII	1	2.9×10^{-5}	6.0×10^{-1}	47	34	10	1
	BnB-NSGAII	**25**	4.2×10^{-5}	4.0×10^{-1}	**40**	**63**	**9.2**	**1.11**
		24	1.4×10^{-5}	2.8×10^{-1}	35	71	30.2	0.71
Brake	NSGAII	2	5.6×10^{-4}	1.2×10^{-1}	188	20	10	1
	BnB-NSGAII	**12**	2.1×10^{-4}	1.5×10^{-1}	**399**	**21**	**3.9**	**3.68**
		6	1.4×10^{-4}	1.8×10^{-1}	474	24	22.5	0.73
Mela	NSGAII	1	3.3×10^{-5}	8.7×10^{-2}	998	41	10	1
	BnB-NSGAII	**10**	8.8×10^{-5}	8.6×10^{-3}	**835**	**33**	**3.6**	**5.29**
		7	2.3×10^{-5}	6.7×10^{-2}	1535	40	12.1	1.13
Truss	NSGAII	**1**	9.6×10^{-5}	1.7×10^{-1}	**915**	**29**	**10**	**1**
	BnB-NSGAII	7	6.4×10^{-5}	2.4×10^{-1}	1438	30	53.7	0.19
Tong	NSGAII	6	3.7×10^{-2}	1.5×10^{-1}	141	0	10	1
	BnB-NSGAII	**10**	7.0×10^{-2}	2.7×10^{-2}	**119**	**0**	**3.6**	**4.85**
		16	3.2×10^{-2}	2.2×10^{-2}	181	0	36	0.78

4.3 Branching Strategies

The performances of the BI, BA and BP [1] strategies are tested on the same benchmark problems. The mixed binary problems are not desirable in this experiment, since the performances of all the strategies are obviously the same on this type of problems. Thus, the mixed-binary problems (Mela and Tong problems) are converted to mixed-integer problems by setting $\boldsymbol{y} = \{0, 1, 2, 3\}$ for all the integer variables of both problems (Mela-v2 and Tong-v2).

Ideally, best branching strategy is the one that can converge MCBB with minimum explo-

[1] BI, BA and BP are Branching by Integer, Anchor and Pareto elements strategy respectively.



(a) Gear problem.

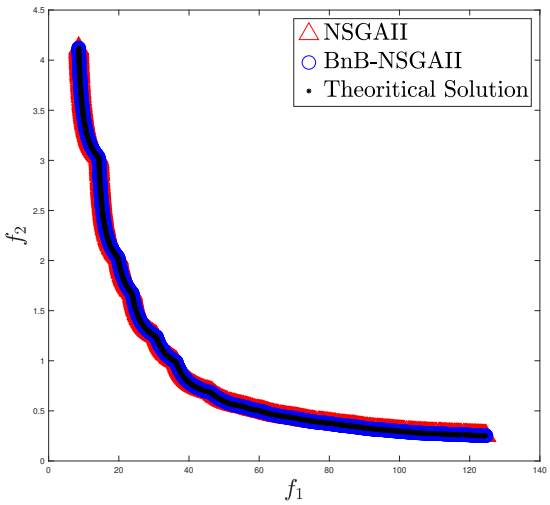
(b) Bearing problem.

(c) Coupling problem.

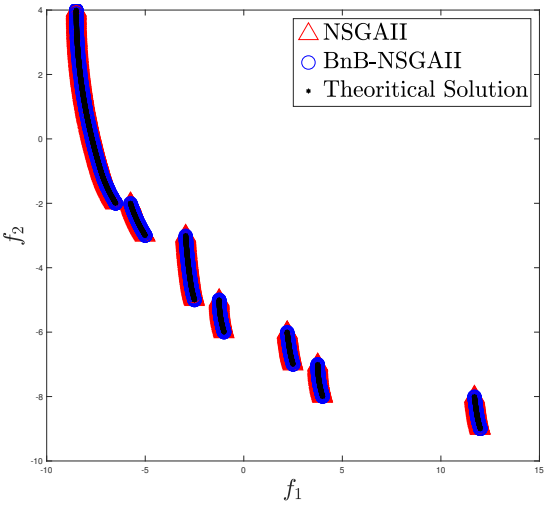
(d) Brake problem.

Figure 4.9: Pareto front obtained by BnB-NSGAII and NSGAII vs the true Pareto for each problem.

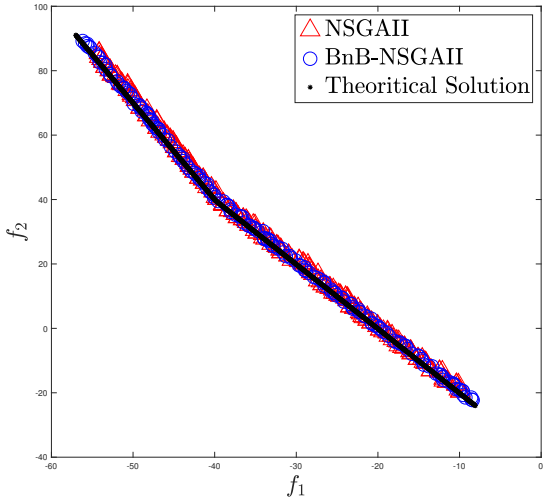
ration effort, i.e. minimum number of total explored nodes Belotti *et al.* [2009]. Hence, in this experiment, the branching strategies are compared based on the computational effort of MCBB which is evaluated by (1) number of total explored nodes; (2) number of total leaf nodes. The number of leaf nodes illustrates the fathoming force of MCBB, which is a critical criteria for any branch and bound algorithm. A strong fathoming force is proved as lower as the number of leaves, which means that more nodes are fathomed. Arguably, other criteria could be used, such as computational time or number of objective function evaluations. However, those are excluded since they are highly affected by the solver used to bound the node. While, the scope of this



(e) Truss problem.



(f) Mela problem.



(g) Tong problem.

Figure 4.9: Pareto front obtained by BnB-NSGAII and NSGAII vs the true Pareto for each problem.

experiment is to evaluate the performance of MCBB with different branching strategies only.

Separation order (integer selection order) significantly affects the exploration effort of MCBB Ghasemi Saghand *et al.* [2019]. In light of this observation, several well-known methods are proposed in literature, e.g. "random branching", "most infeasible branching", "pseudo-costs branching" and "reliability branching" Bonami *et al.* [2011-09]. The aim of these methods is to find the best separation order that helps MCBB to explore minimum number of nodes. Therefore, the numerical experiment that tests the effectiveness of branching strategy must consider the effect of the separation order. Hence, for impartial comparison in this experiment, each problem

is solved repeatedly for each possible separation order by each strategy. Then, for each problem, a strategy is said to be better when its performance is better on more separation orders. As a result, the performance is monitored by a normalized metric called Winning Ratio (WR), which is the number of times each strategy wins (has the best performance) over the number of all possible separation orders $\mathcal{O}$ for each problem.

For more sensible results, the number of nodes is normalized by the size of the combinatorial space $\mathcal{M}$ (total number of combinations of integer variables) in each problem. In BI, the theoretical maximum nodes of the exploration tree is equal to $\mathcal{M}$. While the number of nodes in interval branching methods like BA and BP theoretically can exceeds $\mathcal{M}$. Practically, a strategy is said to be ineffective when the number of nodes exceeds $\mathcal{M}$. $\mathcal{M}$ is defined as El Samrout [2019]:

$$\mathcal{M} = 1 + \sum_{i=1}^{n_i} \left(\prod_{j=1}^{i} n_j \right), \tag{4.9}$$

where n_i is the number of integers in the problem and n_j is the range of the integer variable $\boldsymbol{y}_j$. The combinatorial space of each problem and the total number of separation orders are presented in Table 4.6. The normalized number of nodes is referred to as the exploration ratio, since, it indicates the ratio of the explored portion of the combinatorial domain space.

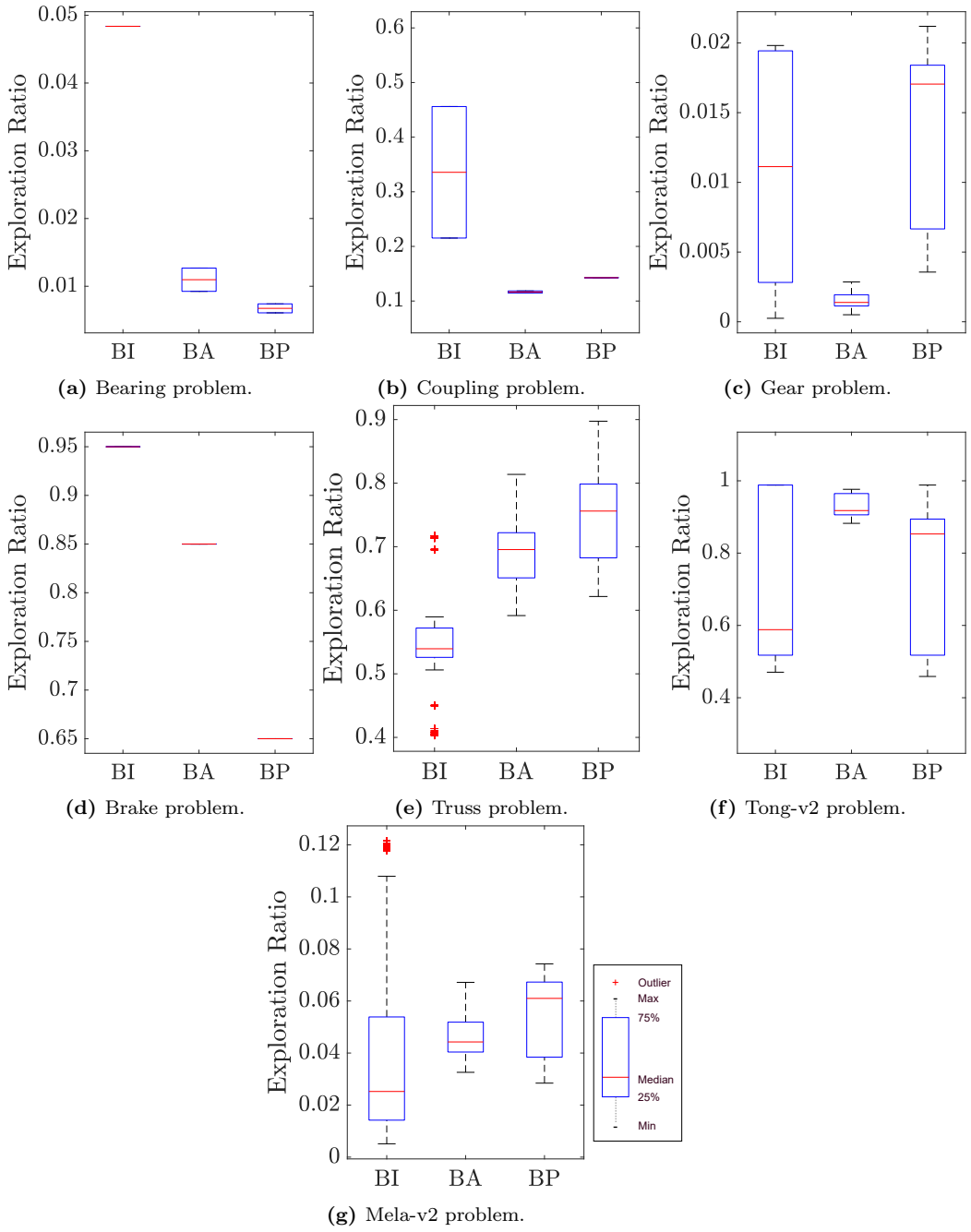
In this experiment, the problems are solved by MCBB with "depth-first" Przybylski & Gandibleux [2017] node selection strategy. At each node, the continuous variables are properly discretized, then a brute-force[1] solver is called to solve the problem. Brute force solver is selected for (1) its ability to provide exact solution even for non-convex MINLPs as it is the type for some of the benchmark problems in this experiment; (2) to eliminate the influence of the solver on the performance (e.g. failure of the solver to find a solution in a specific node for example).

Table 4.6: Benchmark problems sizes

Problem	**Combinatorial Size $\mathcal{O}$**	**# Separation Orders $\mathcal{M}$**
Gear	24	5.8×10^6
Bearing	2	3.8×10^4
Coupling	2	7.8×10^2
Brake	1	2.0×10^1
Truss	720	4.1×10^3
Mela-v2	40320	3.3×10^4
Tong-v2	6	6.8×10^1

[1]The brute force solver is a well-known general problem-solving technique that consists of exhaustively enumerating all possible solutions.

4.3.1 Results and Discussion



(a) Bearing problem. **(b)** Coupling problem. **(c)** Gear problem.

(d) Brake problem. **(e)** Truss problem. **(f)** Tong-v2 problem.

(g) Mela-v2 problem.

Figure 4.10: Distribution of total nodes exploration with respect to the separation orders on each problem.

Results are obtained on Matlab ver. 2017b. Figure 4.10 shows the distribution of the total

nodes exploration ratio obtained by each strategy along all the possible combinations of separation orders. The narrow boxes of BA indicates that BA comparatively is less dependent on the separation order. BA outperforms BI in 4 out of 7 problems, contrariwise, it is outperformed by BI in the rest. Hence, it cannot be clearly observed which strategy is better in terms of exploration effort. Therefore, for each separation order of each problem, there is a specific probability for each strategy to performs better than others.

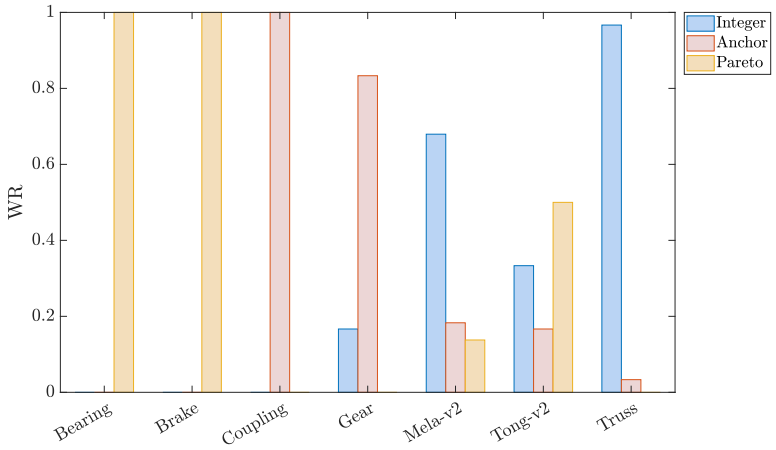
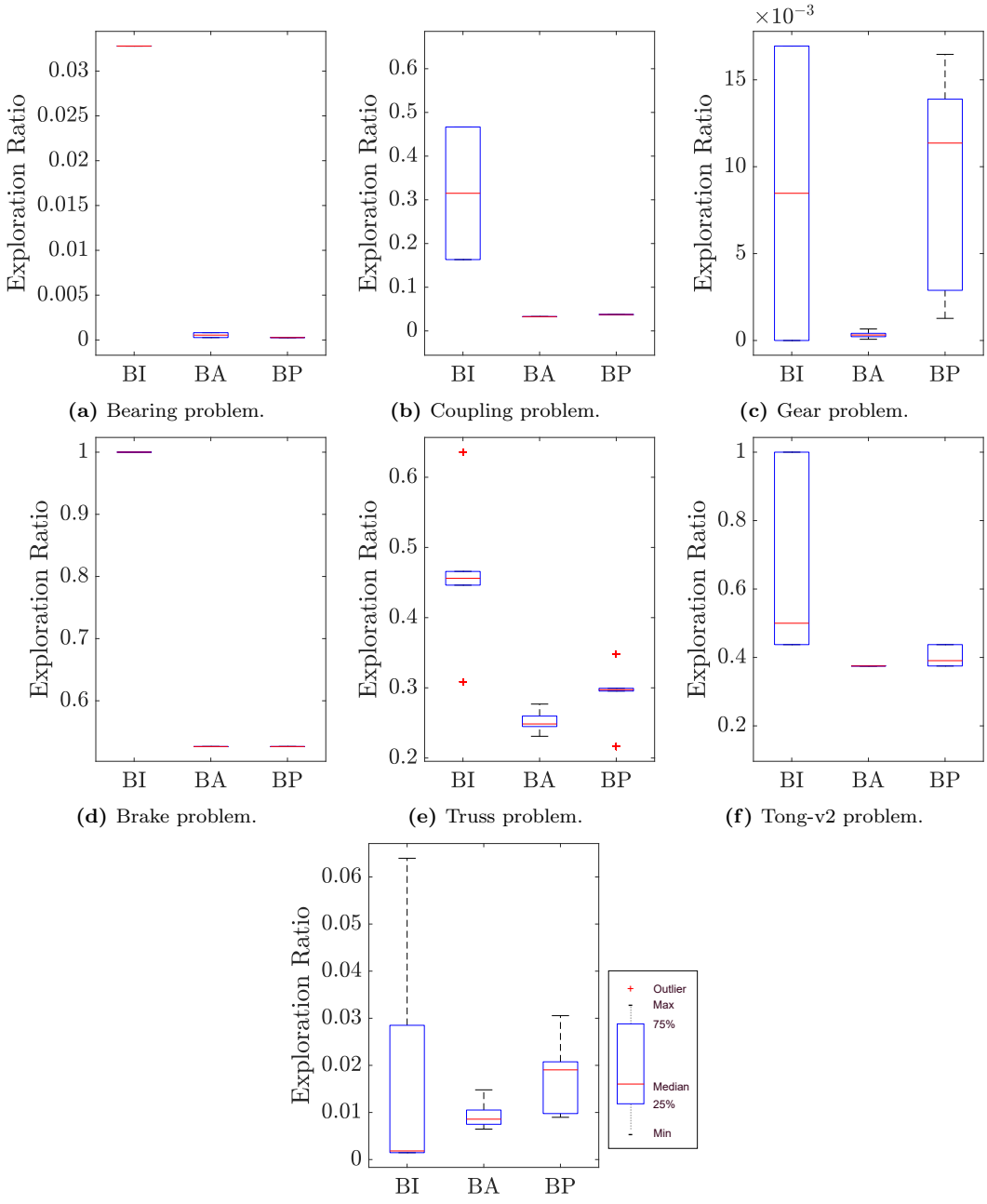


Figure 4.11: Winning ratio of each strategy on each problem based on total nodes exploration ratio.

Figure 4.11 shows the winning ratio (WR) of each strategy on each problem based on the total nodes exploration ratio. WR is the ratio of separation orders where a strategy is considered a winner (minimum nodes exploration ratio) on each problem. Assuming that the designer randomly selects arbitrary separation order, WR can be interpreted as the probability of each strategy to have the best performance with respect to the selected separation order. In the Gear problem for example, BA is the best strategy for 80% of the separation orders, hence for a randomly selected separation order, BA and BI each has probability of 0.8 and 0.2 respectively to performs better than others. In the Bearing problem for example, BP is the best strategy for all separation orders. Although, the results show that no strategy can be said to be the best, the probabilities illustrated in Figure 4.11 show the competitive performance of both BA and BP to BI.

Figure 4.12 shows the distribution of the leaf nodes exploration ratio obtained by each strategy along all the possible combinations of separation orders. BA has lowest exploration ratio on all the problems except on Mela-v2 problem. This indicates that BA has the highest fathoming force among the compared strategies. Moreover, in BA, number of leaf nodes is slightly affected by the separation order on all the problems comparatively to BI. BP overcomes BI on 5 out 7



(a) Bearing problem. **(b)** Coupling problem. **(c)** Gear problem.

(d) Brake problem. **(e)** Truss problem. **(f)** Tong-v2 problem.

(g) Mela-v2 problem.

Figure 4.12: Distribution of leaf nodes exploration with respect to the separation orders on each problem.

problems, which makes BP the next best strategy in terms of fathoming potential. The high performance of BA can be obviously observed in Figure 4.13 that shows the winning ratio of

each strategy on each problem based on the leaf nodes exploration ratio. As a result, BA and BP outperform BI in terms of number of leaf nodes.

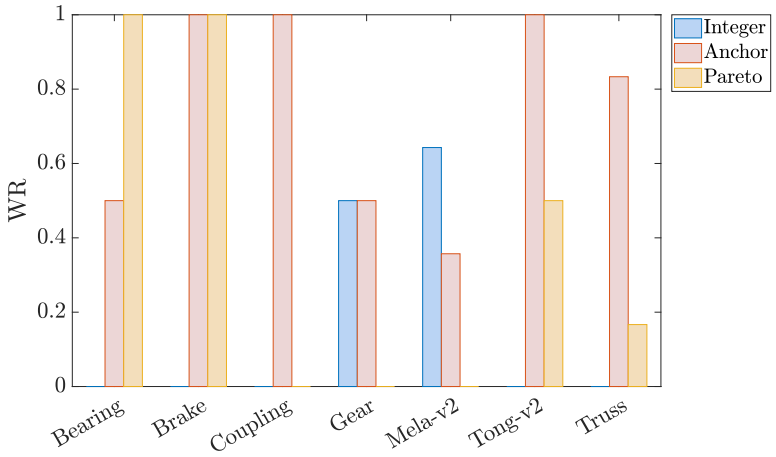


Figure 4.13: Winning ratio of each strategy on each problem based on leaf nodes exploration ratio.

4.4 NSCV

NSCV is tested on two sets of mono-objectives problems. The first set consists of 13 well-known benchmark problems Runarsson & Yao [2002] which have been widely adopted to validate the constraint handling techniques in EAs Cai & Wang [2006]; Ho & Shimizu [2007]; Runarsson & Yao [2002]. This set is referred to as G-Problems in this section. The second set consists of 19 real-world mechanical engineering problems (RW-Problems) introduced in Kumar *et al.* [2020]. Tables 4.7 and 4.8 show distinct properties of G-Problems and RW-Problems sets respectively. The performance of NSCV on these problems is compared to 5 ranking-based CHTs (mentioned in Section 3.5.1) by implementing all of them in the same mono-objective genetic algorithm. All CHTs are initialized by the same random initial population. Table 4.9 shows the characteristics of the genetic algorithm used in this experiment. To tackle the robustness of CHTs, each solver is executed 10 times on each problem. The performances of CHTs are then evaluated based on (1) best optimal solution obtained by each CHT; (2) density of feasible solutions in the last generation (mean of the 10 runs); (3) failure ratio which is the number of failures of a CHT over the number of overall runs of each CHT on each set.

Table 4.7: Properties of G-Problems set.

Name	Dimension	Total Constraints	Active Constraints
G01	13	9	6
G02	20	2	1
G03	10	1	1
G04	5	6	2
G05	4	5	3
G06	2	2	2
G07	10	8	6
G08	2	2	0
G09	7	4	2
G10	8	6	3
G11	2	1	1
G12	3	9^3	0
G13	5	3	3

4.4.1 Profile Performance

One of the many difficulties that arise in the empirical evaluation of new computational techniques is the analysis and reporting of experiments involving a large number of test problems and algorithms. The performance profiles are a methodology specifically developed for this purpose which provides a simple means of visualizing and interpreting the results of large-scale benchmark experiments. Performance profiles (PPs) were introduced in Barreto *et al.* [2010] to evaluate and compare the performance of a set of solvers $\mathcal{S}$ for a given set of optimization problems $\mathcal{P}$ for a specific metric μ. In this experiment, $\mu_{P,S}$ is the mean optimal solution obtained by each solver $S \in \mathcal{S}$ for each problem $P \in \mathcal{P}$. The performance ratio $r_{P,S}$ is defined as:

$$r_{P,S} = \frac{\mu_{P,S}}{b_P}, \tag{4.10}$$

where $b_P = \min(\mu_{P,S} \ \forall \ S \in \mathcal{S})$ is the best solution obtained by all solvers for problem P. Assuming that μ is strictly positive, $r_{P,S}$ thus indicates how many times the optimal solution obtained by S is higher than the best optimal solution obtained by the best performing solver. Let δ be a function defined as:

$$\delta(r_{P,S}, \tau) = \begin{cases} 1 & \text{if} \quad r_{P,S} \leq \tau, \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Table 4.8: Properties of RW-Problems set.

Name	Description	Dimension	Constraints
RW15	Speed reducer weight	7	11
RW16	Industrial refrigeration system	14	15
RW17	Tension/compression spring	3	4
RW18	Pressure vessel design	4	4
RW19	Welded beam design	4	5
RW20	Three-Bar truss	2	3
RW21	Multiple disk clutch brake	5	8
RW22	Planetary gear train design	9	11
RW23	Step-Cone pulley	5	11
RW24	Robot gripper	7	7
RW25	Hydro-Static thrust bearing	4	7
RW26	Four-stages gear box	22	86
RW27	Ten-Bar truss	10	3
RW28	Rolling element bearing	10	9
RW29	Gas transmission compressor	4	1
RW30	Tension/compression spring	3	8
RW31	Gear train design	4	8
RW32	Himmelblau's function	5	6
RW33	Topology optimization	30	30

Hence, $\delta(r_{P,S}, \tau) = 1$ can be interpreted as "solver S solved problem P at a tolerance level τ". Finally, the performance profile ρ_S of a given solver S is defined as:

$$\rho_S(\tau) = \frac{1}{n_p} \sum_{P=1}^{n_p} \delta(r_{P,S}, \tau), \tag{4.12}$$

where, n_p is the number of problems in $\mathcal{P}$, $\rho_S(\tau) \in [0, 1]$ is the ratio of problems that solver S solved at a tolerance level τ. Therefore, if $\tau = 1.01$, this means that an optimal solution for problem P is acceptable up to 1% worse than the best solution obtained for P. Thus, $\rho_S(1.01) = 0.7$, for example, means that solver S solved 70% of the problems within 1% tolerance. Two particular values of $\rho_S(\tau)$ have interesting interpretations:

- $\rho_S(1)$ represents the ratio of problems in which solver S provides the best solution.

Table 4.9: Parameters used in genetic algorithm.

Parameters	Value
Cross over probability	0.9
Mutation Probability	0.95
Population size	100
Maximum generations	500
Crossover operator	Simulated Binary crossover (SBX) Maruyama & Tatsukawa [2017]
ETAC	100
Mutation operator	Partially-mapped crossover (PMX) Maruyama & Tatsukawa [2017]
ETAM	10
Selection method	Tournament selection method (TSM) Fang & Li [2010]
P_f (for SR and GCR)	0.35

- $\rho_S(\infty)$ represents the fraction of problems solver S is able to eventually solve.

PPs are then illustrated by several curves $[\rho_S(\tau), \tau]$, one for each solver. An illustrative example is presented in Figure 4.14. Solver S_1 outperforms the other solvers for 60% of the problems; this is reflected in its value $\rho_{S_1}(1) = 0.6$. Solvers S_1 and S_3 provide solutions for all problems, so their curves both reach the maximum value $\rho_S = 1$. However, S_1 is the most robust since it reaches $\rho_{S_1} = 1$ at lower τ. In the other hand, the curve of S_2 stops at $\rho_{S_2} = 0.8$, which means that S_2 fails to solve 20% of the problems. Therefore, in this example it is clear that S_1 is the best Solver. The PPs are thus an interesting tool for the easy visualization and interpretation of results from computational experiments. If one is interested only in the number of wins, it is enough to compare the values of $\rho_S(1)$ for all methods. While, if one is interested in methods with a high probability of success, then the final section of the curves should be investigated. Overall, the highest the $\rho_S(\tau)$ values, the better the method is, so a quick visual assessment of the performance of all methods may be performed simply by observing which one has the curves above all others.

4.4.2 Results and Discussion

The results are obtained using Matlab v 2017b. Figure 4.15 shows the profile performance of the 6 CHTs on G-Problems set. At $\tau = 1$, GCR has the best accuracy with $\rho \approx 0.4$ which means that GCR provides the best optimal solution for 40% of the problems. The next best is NSCV with $\rho = 0.2$. Yet, at $\tau = 1.01$, the performance of NSCV promptly increases to $\rho = 0.77$. This means that NSCV provides the best solution of 10 out of 13 problems with a 1% tolerance. The

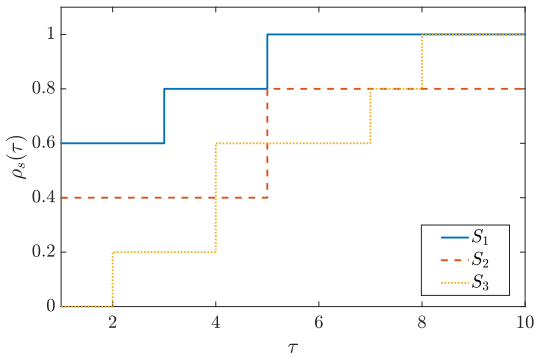


Figure 4.14: Example of profile performance

performance of NSCV furtherly increases to reach $\rho = 0.85$ at $\tau = 2.8$ which means that NSCV is eventually capable to solve maximum number of problems relatively to other CHTs with 11 out of 13 problems. This robustness of NSCV is additionally shown in Figure 4.16 which shows the ratio of failure of each CHT on the G-Problems set. NSCV has the minimum failure ratio (< 0.3), while the rest of CHTs have failure ratios between 0.4 (SF) and 0.7 (SR). Another interesting criterion is the feasible solution density in the last generation. The mean feasible density of all runs for each CHT is shown in Figure 4.17. VCH has the maximum feasible density (45%). The next maximum belongs to NSCV with a 40% feasible density. Since we are instantaneously

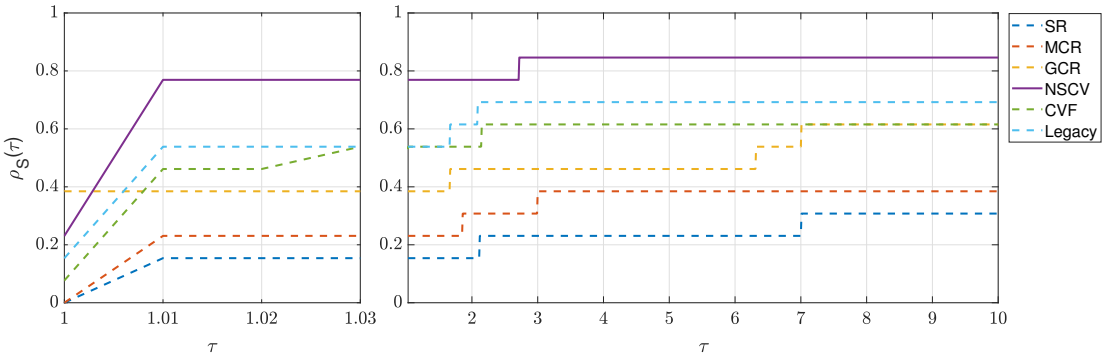


Figure 4.15: Profile performance of CHTs on G-Problems set

interested in both accuracy and robustness of the CHT, the performance of NSCV is considered the best comparatively to the other compared CHTs on the G-Problems set.

Figure 4.18 shows the profile performance of each CHT on the RW-Problems set. The best accuracy ($\rho(1)$) is achieved by NSCV. Where, NSCV provides the best solution of 30% of RW-Problems. At $\tau = 1.01$, NSCV performance curve increases to reach $\rho = 0.83$ which is the best

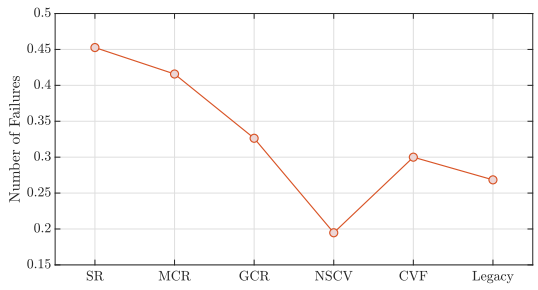


Figure 4.16: Ratio of failure of each CHT on G-Problems set

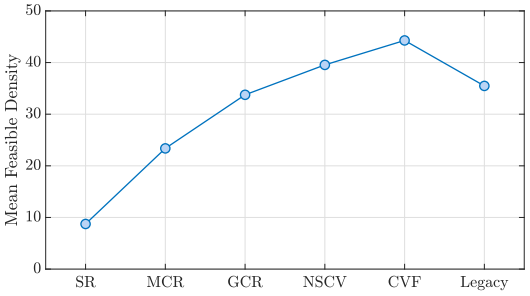


Figure 4.17: Overall mean feasible density on G-Problems set

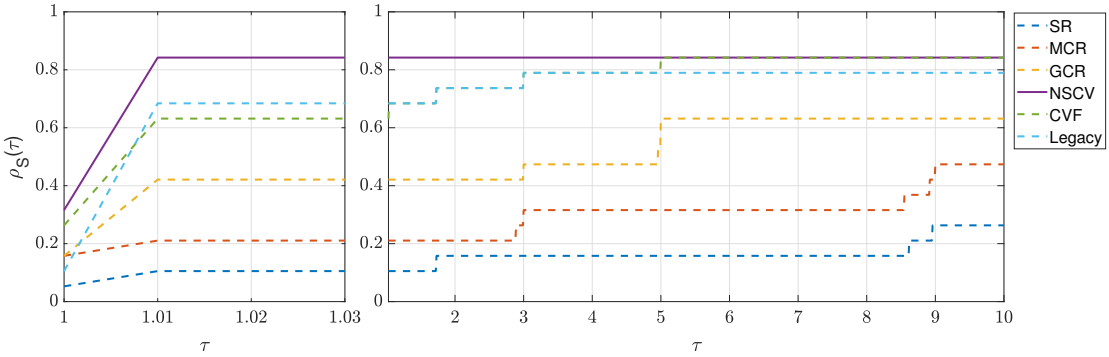


Figure 4.18: Profile performance of each CHT on RW-Problems set

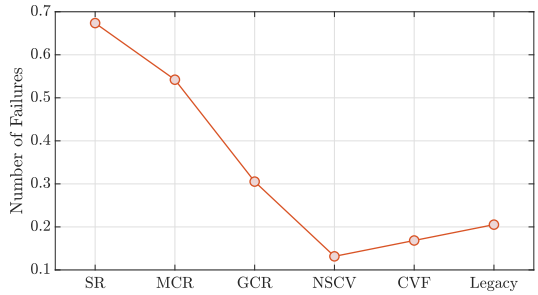


Figure 4.19: Ratio of failure of each CHT on RW-Problems set

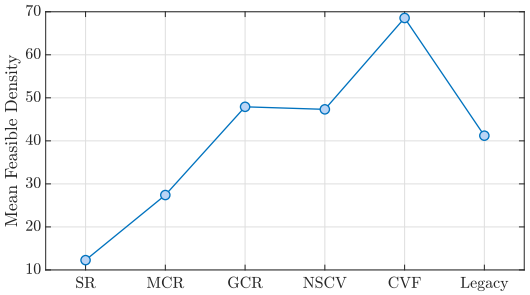


Figure 4.20: Overall mean feasible density on RW-Problems set

among other CHTs. This means that NSCV is capable to solve maximum number of problems (13 out of 19 problems) with a tolerance of 1%. The robustness of NSCV is confirmed in Figure 4.19, where NSCV has the lowest failure ratio (0.2). Figure 4.20 shows that VCH has again the maximum mean feasible density (0.7) in the last generation among all CHTs tested on RW-Problems set. While the next bests are GCR and NSCV respectively with density ≈ 0.5. NSCV

again proves its competitive performance among 5 state-of-art ranking based CHTs in terms of both accuracy and robustness.

CHTs that depend on feasible separation (NSCV, SF and VCH) are noticed to have generally better performance than other CHTs (GCR, MCR and SR). This observation may lead to a conclusion that feasible separation enhances both the robustness and the accuracy of a CHT.

4.5 Conclusion

In this chapter, we have examined the performance of (1) BnB-NSGAII in terms of quality and cost; (2) the proposed branching strategies in terms of MCBB convergence; (3) NSCV in terms of enhancing the performance of the genetic algorithm.

First, we have presented a benchmark experiment based on a statistical assessment to evaluate the performance of BnB-NSGAII algorithm comparatively to NSGAII. Both are tested on seven MO-MINLP problems from the literature. The true Pareto solutions of these problems are known, so the Pareto fronts resulting from both algorithms can be compared to the true ones. Metaheuristic optimization methods are often used as a compromise between optimization effort and accuracy (w.r.t. global optimality). The necessity of more accurate solutions with the toleration of more computational effort, a new metric that measures the potential improvement of the solution concerning computational cost - Investment Ratio (IR) - is defined. Both NSGAII and BnB-NSGAII performances (quality and cost) are affected by selecting different values of tuning parameters. In the purpose of impartial comparison between them, different tuning parameters are applied to each method on each of the 7 problems independently. The results generally verify the competitive performance of BnB-NSGAII with respect to NSGAII in terms of the quality of the solution.

Second, the performances of the branching by integer (BI), anchor points (BA) and pareto elements (BP) strategies are tested on the same benchmark problems. Ideally, best branching strategy is the one that can converge MCBB with minimum exploration effort, i.e. minimum number of total explored nodes. Moreover, the separation order (integer selection order) significantly affects the exploration effort of MCBB. Therefore, the numerical experiment that tests the effectiveness of branching strategy must consider the effect of the separation order. Hence, for impartial comparison in this experiment, each problem is solved repeatedly for each possible separation order by each strategy. The results show that for each separation order of each problem, there is a specific probability for each strategy to performs better than others in terms of total explored nodes. However, the results indicate that branching by anchor points and branching by Pareto elements has the highest fathoming force respectively among the branching by integer strategy. As a result, BA and BP outperform BI in terms of number of leaf nodes.

Next, NSCV is tested on two sets of mono-objectives problems. The performance of NSCV on these problems is compared to 5 ranking-based CHTs by implementing all of them in the same mono-objective genetic algorithm. Since we are instantaneously interested in both accuracy and robustness of the CHT, the performance of NSCV is considered the best comparatively to the other compared CHTs on the two sets of problems.

Finally, the results in this chapter show that investing the computational effort on a hybrid approach as BnB-NSGAII is more beneficial than investing in the metaheuristic itself as NSGAII in our case. Moreover, the results show that the new branching strategies enhance the performance of MCBB. On the other hand, NSCV enhances the performance of the genetic algorithm. In the next chapter, BnB-NSGAII, NSCV and the new branching strategies are implemented in one solver to be examined on a real-world complex mechanical design optimization problem.

Chapter 5

Applications

Contents

5.1 Introduction 106
5.2 3 stages Reducer Problem 107
5.2.1 Closure condition 108
5.2.2 Mechanical constraint for one stage of the mechanism 110
5.2.3 3SR Complexity 112
5.3 An Application of BnB-NSGAII: Search Feasibility 114
5.3.1 Numerical Experiment 114
5.4 NSCV Case Study 116
5.5 An Application of BnB-NSGAII: Seek Optimality 119
5.6 Conclusion 121

5.1 Introduction

In this chapter, BnB-NSGAII is tested on a real-world 3 stages reducer state of art problem whose true Pareto solution cannot be obtained. The problem has a huge combinatorial space which questions the capability of BnB-NSGAII to converge - at least - to a good Pareto front. The apprehension of high computational effort and the high exploration potential of BnB-NSGAII are the main motives to propose two applications of BnB-NSGAII on this problem and consequently on similar huge combinatorial space problems. The first application is to use BnB-NSGAII to search for individuals having specific criteria (such as feasible individuals as in this case) to inject them then in the initial population of another metaheuristic as NSGAII. Hence, this application is referred to as "Search feasibility application". In this application, BnB-NSGAII converges whenever at least one individual having this criteria is found. Therefore, the remaining question is: *can BnB-NSGAII converge within a short time (such as 30 minutes)?* The second application is the "Seek optimality application" that addresses the real potential of BnB-NSGAII to converge on this kind of problems. In this application, BnB-NSGAII is called to entirely solve the problem. Hence, the main question is: *can BnB-NSGAII converge to a good Pareto within an acceptable computational time (not more than 24 hours)?*

In this chapter, the effect of the legacy feature on BnB-NSGAII performance is also studied. In this aim, in the "search feasibility application", the traditional "branching by integer" strategy and Superiority of Feasible (SF) constraint handling technique (SF is the default CHT for NSGAII as proposed in [Deb & Jain, 2003]) are utilized in BnB-NSGAII to highlight the effect of the legacy feature only on the performance. Then, NSCV is solely examined comparatively to the five previously studied CHTs, as a case study, on the 3 stages reducer problem to study the effect of NSCV on this problem. Finally, in the "seek optimality application", all our proposed features are combined into one solver trying to solve the 3 stages reducer problem. Figure 5.1 shows the schematic of the benchmark of BnB-NSGAII on the 3 stages reducer problem.

This chapter is organized as follows. The 3 stages reducer problem is firstly presented in Section 5.2 showing the problem formulation and its complexity. Section 5.3 presents the "search feasibility application" showing the effect of the legacy feature on the performance of BnB-NSGAII. Section 5.4 demonstrates the effect of NSCV on the 3 stages reducer problem as a case study compared to five other CHTs. Finally, Section 5.5 presents the "seek optimality application" showing the overall performance of BnB-NSGAII on the real-world 3 stages reducer problem.

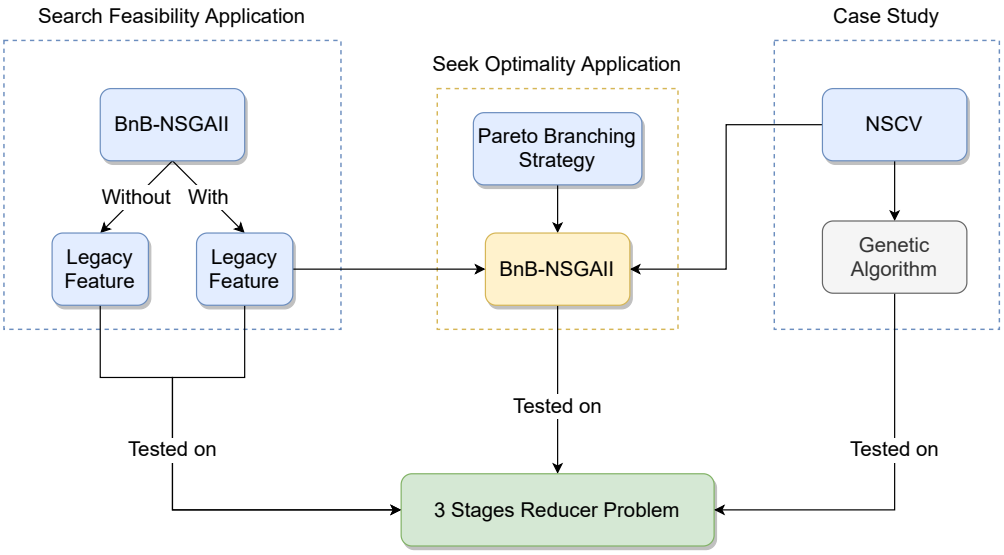


Figure 5.1: Benchmarking of BnB-NSGAII on the 3 stages reducer problem.

5.2 3 stages Reducer Problem

In Fauroux [1999], the design of the 3 stages Reducer (3SR) optimization problem has been introduced to illustrate the optimal design framework of the power transmission mechanism. This problem has been a point of interest for many researchers in different domains. Engineering researchers enhance the problem for mechanical engineering applications. In Fauroux & Lafon [2004], the problem is extended to a mixed variables optimization problem. And recently a similar problem is stated in Han *et al.* [2020] to illustrate the optimization of the volume and layout design of 3SR. Due to the problem complexity, optimization researchers are interested to test optimization methods on it. In Yvars & Zimmer [2018], the authors use the 3SR problem to examine the performance of the constraint propagation method.

In this thesis, the 3SR problem is reformulated to a bi-objective problem with additional constraints to meet the ISO mechanical standards. Those additional constraints increase the complexity of the problem, such that NSGAII performance is not sufficient.

The design problem consists in finding dimensions of main components (pinions, wheels and shafts) of the 3 stages reducer (Figure 5.2) to minimize the following bi-objective problem :

1. The volume of all the components of the reducer :

$$f_1(\boldsymbol{x}) = \pi \left(\sum_{s=0}^{s=3} l_{a_s} {r_{a,s}}^2 + \sum_{s=1}^{s=3} \left[b_s \frac{m_{\text{ns}}^2}{2} (Z_{s,1}^2 + Z_{s,2}^2) \right] \right) \tag{5.1}$$

2. The gap between the required reduction ratio $\bar{u}$ and the ratio of the reducer (tolerance):

$$f_2(\boldsymbol{x}) = \frac{1}{\bar{u}} \left| \bar{u} - \prod_{s=1}^{s=3} \frac{Z_{s,2}}{Z_{s,1}} \right|, \ \bar{u} > 1 \tag{5.2}$$

The problem is designed assuming the following are known:

- The power to be transmitted, P_t and the speed rotation of input shaft N_e.
- The total speed rotation reduction ratio $\bar{u}$, the position of the output shaft from the input shaft position (Figure 5.3).
- The dimension of the casing box.

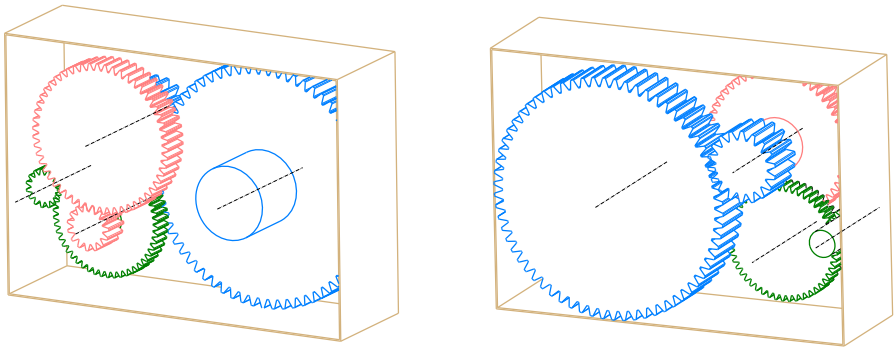


Figure 5.2: Front and back view of a 3 stages reducer with closure.

5.2.1 Closure condition

Interference and fitting constraints are adopted from Han *et al.* [2020]. In Fauroux & Lafon [2004], the closure condition was expressed with the distance between the terminal point O_3 shown in Figure 5.3 and required position of the center of the output shaft. The coordinate of O_3 can be easily computed with the center distance of each stage and the angle ξ_1, ξ_2 and ξ_3. But, if the center distance of each stage allow this closure condition, the value of ξ_2 and ξ_3 can be computed. By this way the number of variables in the optimization problem can be reduced.

For a given value of ξ_1 and $r_{1,1}$, $r_{1,2}$, the center distance of each stage allows a closure if:

$$\|\vec{O_1O_3}\| \leq \|\vec{O_1O_2}\| + \|\vec{O_2O_3}\|$$

Assuming the previous condition is true, the two intersections of the circle of center O_1 of $\|\vec{O_1O_2}\|$ radius and the circle of center O_3 of $\|\vec{O_2O_3}\|$ radius (Figure 5.4) can be computed.

With $a_2 = \|\vec{O_1O_2}\|$ and $a_3 = \|\vec{O_2O_3}\|$:

$$\begin{cases} a_2 \sin\alpha_1 - a_3 \sin\alpha_3 = 0 \\ a_2 \cos\alpha_1 + a_3 \cos\alpha_3 = \|\vec{O_1O_3}\| \end{cases}$$

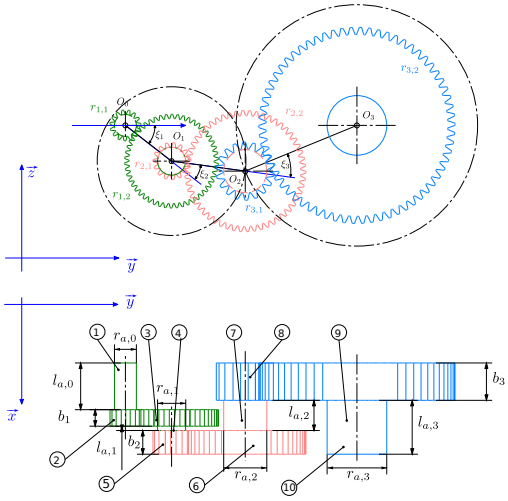


Figure 5.3: Detailed view of the 3 stages reducer.

which give :

$$\cos\alpha_1 = \frac{\overline{O_1H}}{\overline{O_1O_2}} = \frac{a_2^2 - a_3^2 + (O_1O_3)^2}{2a_2\overline{O_1O_3}}$$

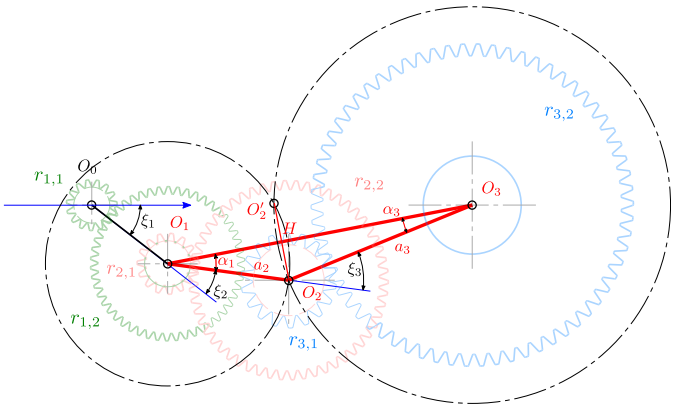


Figure 5.4: Gear mesh for each stage.

Knowing α_1, the computation of coordinate of O_2 and O_2' is straightforward. If the two positions O_2 and O_2' allow the wheel of the 2nd stage to fit in the casing box, then O_2' is preferred for lubrication reason.

5.2.2 Mechanical constraint for one stage of the mechanism

5.2.2.1 Constraints related to the gear pair

Following the recommendation of International Standard ISO 6336, International Organization for Standardization (Ginebra) [2019a], International Organization for Standardization (Ginebra) [2019b], International Organization for Standardization (Ginebra) [2019c] and knowing the geometry of gear pair, the material, the working conditions, the contact and σ_{H} the bending stress σ_{F} in the gear pair can be calculated. These stresses must be less of equal to the respective permissible value σ_{HP} and σ_{FP}, depending on the material and the working conditions.

From International Organization for Standardization (Ginebra) [2019c] the bending stress σ_{F} is given by (1 for the pinion and 2 for the wheel):

$$\sigma_{\mathrm{F}(1,2)} = \sigma_{\mathrm{F0}} \left(K_{\mathrm{A}} K_{\mathrm{V}} K_{\mathrm{F}\alpha} K_{\mathrm{F}\beta} \right),$$

with $\sigma_{\mathrm{F0}(1,2)}$, the nominal tooth stress :

$$\sigma_{\mathrm{F0}(1,2)} = \frac{F_t}{b m_{\mathrm{n}}} \left(Y_{\mathrm{F}} Y_{\mathrm{S}} Y_{\beta} Y_{\mathrm{B}} Y_{\mathrm{DT}} \right),$$

where :

- F_t: is the tangential load from International Organization for Standardization (Ginebra) [2019a].
- b: is the facewidth.
- m_{n}: is the normal module.

Factors K_{A}, K_{V}, $K_{\mathrm{F}\alpha}$, $K_{\mathrm{F}\beta}$ are related to dynamic and loading conditions in the gear. Factors Y_{F}, Y_{S}, Y_{β}, Y_{B}, Y_{DT} are related to the geometry effect on stress.

From International Organization for Standardization (Ginebra) [2019c], the permissible bending stress σ_{FP} is given by:

$$\sigma_{\mathrm{FP}} = \frac{\sigma_{\mathrm{FLim}}}{S_{\mathrm{Fmin}}} \left(Y_{\mathrm{ST}} Y_{\mathrm{NT}} Y_{\delta \mathrm{relT}} Y_{R\mathrm{relT}} Y_{\mathrm{X}} \right),$$

with σ_{FLim} is the nominal stress number (bending) from reference test gears International Organization for Standardization (Ginebra) [2019d] and S_{Fmin} the minimal required safety factor. Factors Y_{ST}, Y_{NT}, $Y_{\delta \mathrm{relT}}$, $Y_{R\mathrm{relT}}$, Y_{X} are related to the reference test gears and the geometry and material conditions of the gear pair.

From International Organization for Standardization (Ginebra) [2019b] the contact stress is given by (1 for the pinion and 2 for the wheel):

$$\sigma_{\mathrm{H}(1,2)} = Z_{(\mathrm{B,D})} \sigma_{\mathrm{H0}} \sqrt{K_{\mathrm{A}} K_{\mathrm{V}} K_{\mathrm{H}\alpha} K_{\mathrm{H}\beta}},$$

with σ_{H0} is the nominal contact stress :

$$\sigma_{\mathrm{H0}} = (Z_{\mathrm{H}} Z_{\mathrm{E}} Z_{\varepsilon} Z_{\beta}) \sqrt{\frac{F_t}{b d_1} \frac{u+1}{u}}.$$

Factors Z_{H}, Z_{E} ,Z_{ε}, Z_{β} are related to the Hertzian theory of contact. Geometry and material in the gear pair are considered in these factors.

From International Organization for Standardization (Ginebra) [2019b] the permissible contact stress σ_{HP} is :

$$\sigma_{\mathrm{HP}} = \frac{\sigma_{\mathrm{HLim}}}{S_{\mathrm{Hmin}}} \left(Z_{\mathrm{NT}} Z_{\mathrm{L}} Z_{\mathrm{V}} Z_{\mathrm{R}} Z_{\mathrm{W}} Z_{\mathrm{X}}\right),$$

where σ_{HLim} is the allowable contact stress number and S_{Hmin} is the minimum required safety factor for surface durability. Factors Z_{NT}, Z_{L}, Z_{V}, Z_{R} ,Z_{W}, Z_{X} are related to lubrication conditions, surface roughness and hardened conditions and size of the tooth.

So to respect the requirement specification of a given power to be transmitted, the gear pair must respect:

$$\sigma_{\mathrm{F}(1,2)} \leq \sigma_{\mathrm{FP}}$$
$$\sigma_{\mathrm{H}(1,2)} \leq \sigma_{\mathrm{HP}}.$$

Considering that σ_{F} is proportional to F_t and σ_{H} is proportional to $\sqrt{F_t}$ for a given gear pair, we can rewrite these 2 conditions with P_t the power to be transmitted :

$$\frac{\sigma_{\mathrm{FP}}}{\sigma_{\mathrm{F}(1,2)}} P_t \geq P_t$$
$$\left(\frac{\sigma_{\mathrm{HP}}}{\sigma_{\mathrm{H}(1,2)}}\right)^2 P_t \geq P_t$$

Usually, some factors are slightly different for the pinion and the wheel. So the transmitted power by the pinion (1) is different than by the wheel (2). Hence, the minimal value is considered.

Finally, for the stage number s on the reducer, the following conditions must be fulfilled:

$$\min\left(\frac{\sigma_{\mathrm{FP}s}}{\sigma_{\mathrm{F}(1,2)s}}\right) P_t \geq P_t \tag{5.3}$$
$$\min\left(\frac{\sigma_{\mathrm{HP}s}}{\sigma_{\mathrm{H}(1,2)s}}\right)^2 P_t \geq P_t. \tag{5.4}$$

Additionally, the following conditions must be respected :

- For the transverse contact ratio : $\varepsilon_{\alpha} \geq 1.3$.
- For the minimal face width : $b \geq 0.1 d_2$.
- For the maximal face width : $b \leq d_1$.

In order to use pinion with at least $Z_{\mathrm{min}} = 14$ teeth, the value of the profile shift coefficient must be adjusted to avoid gear meshing with the following relation:

$$Z_{\mathrm{min}} \geq \frac{2(1-x_1)}{\sin \alpha_{\mathrm{n}}^2} \Rightarrow x_1 \geq 1 - Z_{\mathrm{min}} \frac{\sin \alpha_{\mathrm{n}}^2}{2} \Rightarrow x_1 \geq 0.1812.$$

5.2.2.2 Constraint related to shaft's reducer

In each of the 4 shafts of the mechanism, the transmitted torque produce shear stress. This stress must not exceed the allowable shear of the material of shafts $\tau_{\max}$. We assume here that all the shaft are using the same steel and that all the shafts can be consider as beams. So, with $r_{a,0}$, the radius of input shaft, and $r_{a,s}$, $s = 1 \dots 3$ the radius of output shaft of the three stages, we have :

$$\tau_s = \frac{2C_s}{\pi {r_{a,s}}^3} \leq \tau_{\max} \text{ for } s = 1 \dots 3, \tag{5.5}$$

where C_s is the output torque of each stage, C_e the torque on the input shaft, $Z_{i,1}$ and $Z_{i,2}$ are the number of teeth for pinion $(_1)$ and wheel $(_2)$ of stage i:

$$C_s = C_e \prod_{i=1}^{i=s} \frac{Z_{i,2}}{Z_{i,1}}.$$

For the input shaft:

$$\tau_0 = \frac{2C_e}{\pi {r_{a,0}}^3} \leq \tau_{\max}. \tag{5.6}$$

The total rotation angle between the initial section of the input shaft and the final section of the output shaft is:

$$\theta = \frac{2C_e l_{a,0}}{G\pi {r_{a,0}}^3} + \sum_{s=1}^{s=3} \frac{2C_s l_{a,s}}{G\pi {r_{a,s}}^3}.$$

For better the dynamic behavior of the reducer, the total rotation angle should be limited by a maximal value $\theta_{\max}$.

$$\theta \leq \theta_{\max}. \tag{5.7}$$

5.2.3 3SR Complexity

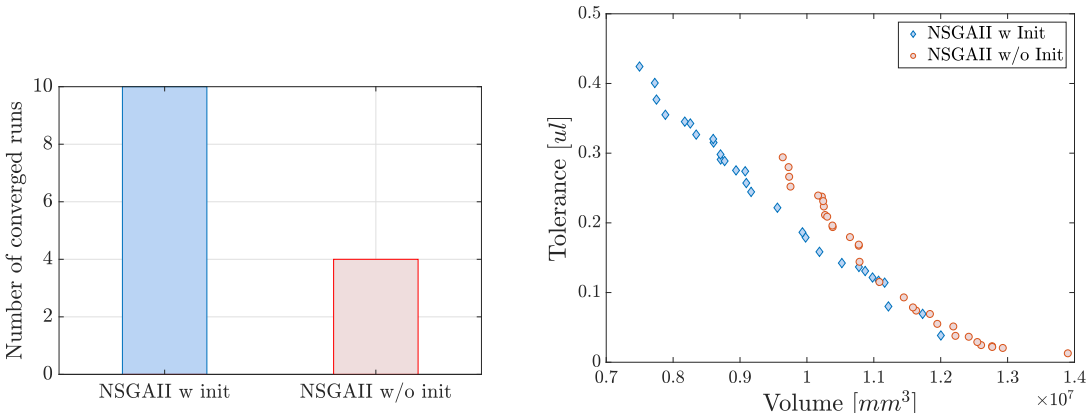
The 3SR problem is formulated with 2 objective functions, 41 constraints, 3 categorical variables (gears modules) which are indexed as non-relaxed integers, 6 integer variables (number of teeth), and 11 continuous variables. Gears modules have 41 possibilities, pinion number of teeth ranges from 14 to 30 and wheel number of teeth ranges from 14 to 150. Hence, the combinatorial space of the 3SR problem consists in $41^3 * (30 - 13)^3 * (150 - 13)^3 \simeq 8.7 \times 10^{14}$. Thus, the problem is considered a mid-sized problem concerning the number of variables and constraints, but, huge combinatorial space. Appendix B provides the data used in 3SR problem.

The additional constraints increase the complexity of the problem. This is noticed by solving the problem using NSGAII with different initial conditions as follows. In the first hand, NSGAII is initialized with 1 feasible individual. On the other hand, NSGAII is randomly initialized. Each was run 10 times with the same parameters shown in Table 5.1. Figure 5.5a shows how many run each method converged to a feasible solution out of 10. Figures 5.5a and 5.5b show that if the initial population contains at least 1 feasible individual, NSGAII converges to a good

approximated Pareto front every time. Whilst, if NSGAII is initialized with a random population, NSGAII either fails to converge to a feasible solution, or it converges to a low-quality Pareto front.

Table 5.1: Parameters used for NSGAII algorithm

Parameters	Value
Cross over probability	0.8
Mutation Probability	0.9
Population size	200
Allowable generations	500
Constraint handling	Superiority of Feasible Deb *et al.* [2002]
Crossover operator	Simulated Binary crossover (SBX) Maruyama & Tatsukawa [2017]
ETAC	100
Mutation operator	Partially-mapped crossover (PMX) Maruyama & Tatsukawa [2017]
ETAM	10



(a) Number of converged runs.

(b) Best Pareto front.

Figure 5.5: Results of 3SR problem solved by NSGAII with (blue) and without (red) initial feasible seed.

Figure 5.6 shows part of the domain of the 3SR problem explored by NSGAII with feasible initial population. The explored domain shows the complexity of the problem, where both feasible and infeasible solutions share the same domain on the projected objective domain. Moreover, all the feasible solutions are too near to the infeasible ones.

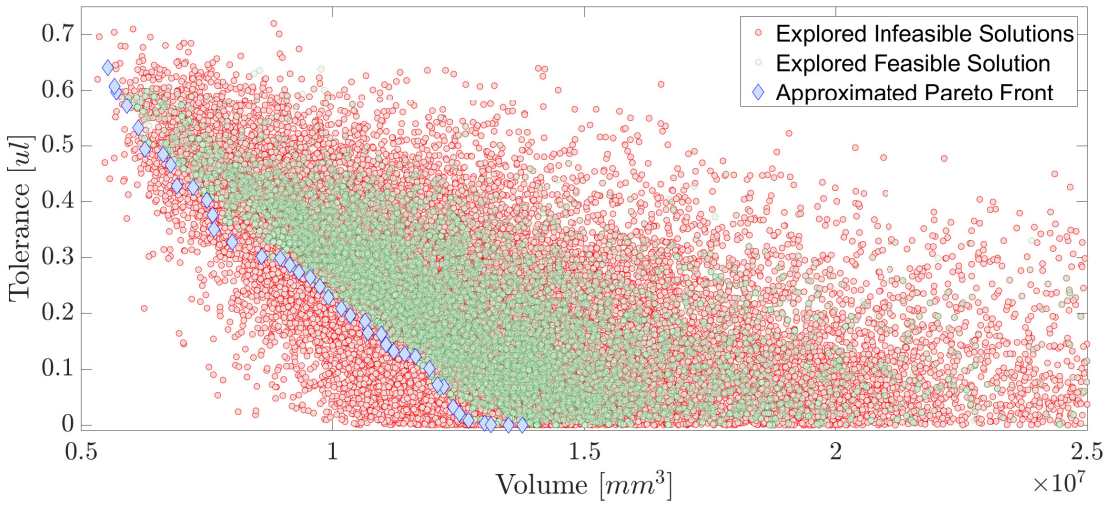


Figure 5.6: Explored portion of the domain, showing the 3SR problem complexity.

5.3 An Application of BnB-NSGAII: Search Feasibility

BnB-NSGAII is characterized by high exploration potential. Thus, in this section, BnB-NSGAII is used to search for at least one feasible solution for the 3SR problem. For this aim, BnB-NSGAII is properly modified to 1) continue enumeration of the combinatorial tree even if the root node is infeasible. 2) stop whenever a feasible solution(s) is found. Then, NSGAII is called to solve the 3SR problem by initializing it with the feasible solution(s) found as shown in Figure 5.7.

5.3.1 Numerical Experiment

NSGAII and BnB-NSGAII with and without the legacy feature were tested on the 3SR problem. Each method was run 10 times. The test was done using the same parameters for the 3 solvers. Table 5.1 shows the parameters used in this experiment.

5.3.1.1 Results and Discussion

In this experiment, the evaluation of the performance of each method is limited to how many times the method finds at least 1 feasible solution over the 10 runs. Figure 5.8 shows the number of times each method succeeded the test. It can be obviously concluded that BnB-NSGAII legacy method overcomes the performances of NSGAII and BnB-NSGAII. It should be noted that the computational effort is not regarded since all the runs converge within 30 minutes. Which is considered an acceptable time for such a problem.

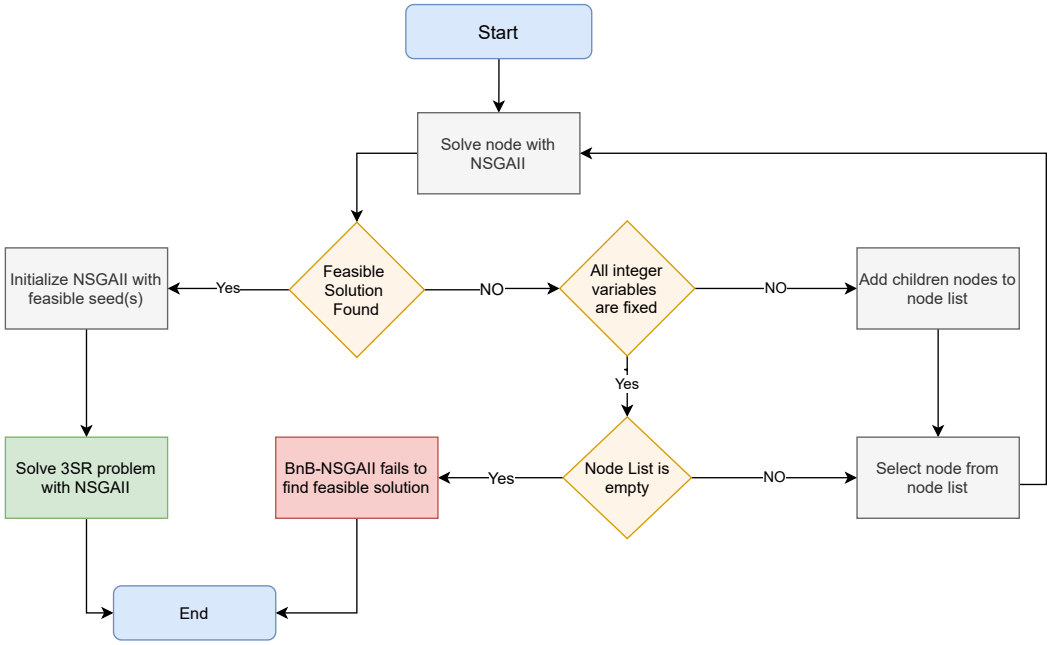


Figure 5.7: Flowchart of BnB-NSGAII "Search Feasibility" application.

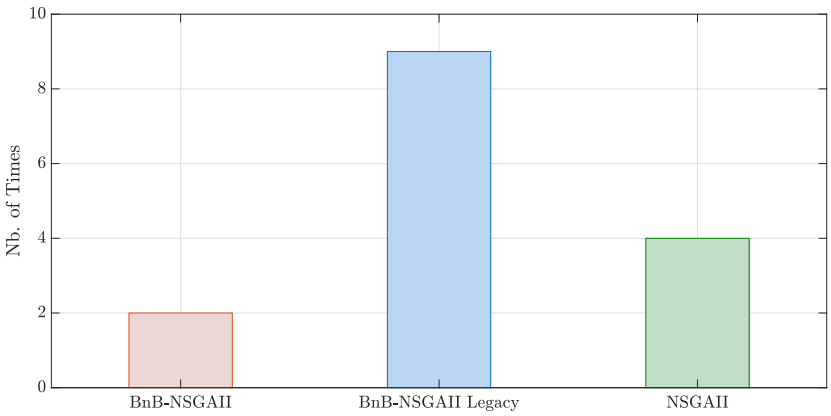
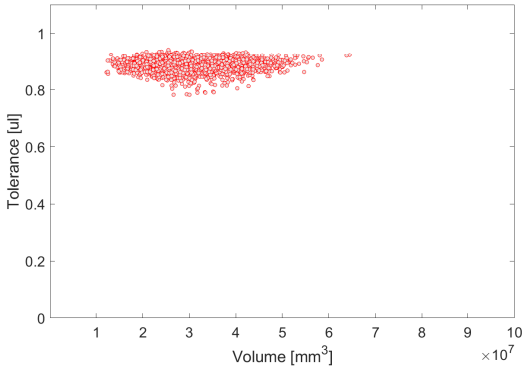
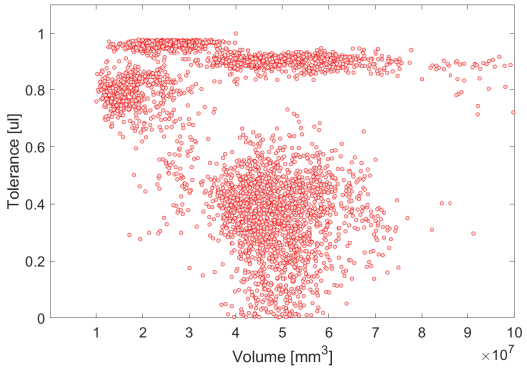


Figure 5.8: Number of converged runs

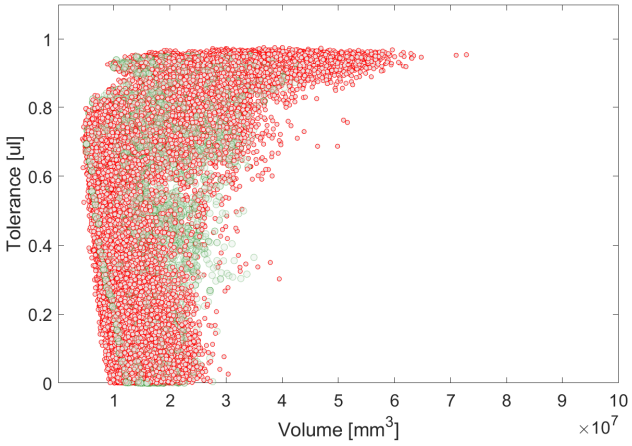
Figure 5.9a shows that NSGAII explored local space of the domain depending on the initial population. While Figure 5.9b shows that BnB-NSGAII explored random spaces of the domain. Figure 5.9c shows that the legacy feature guides the exploration force of BnB-NSGAII towards the feasible solutions.



(a) NSGAII



(b) BnB-NSGAII



(c) BnB-NSGAII Legacy

Figure 5.9: Explored domain by (5.9a) NSGAII, (5.9b) BnB-NSGAII and (5.9c) BnB-NSGAII legacy methods. Feasible and infeasible individuals are plotted in green and red respectively.

5.4 NSCV Case Study

In this section, the competitive performance of NSCV on benchmark problems is assessed on 3SR problem in the purpose of identification of the impact of NSCV on the solution of the 3SR problem. To preserve the same benchmark presented in Section 3.5.2, 3SR problem is solved as a mono-objective problem taking the volume objective only. The performance of NSCV is evaluated comparatively to other CHTs by solving the 3SR problem 30 times by each CHT starting from the same initial population using the same parameters defined in Table 5.1. Figure 5.10 shows the number of successful runs of each CHT, i.e the solver finds at least one feasible solution. NSCV records the highest number of successful runs with 29 out of 30 runs, which indicates that NSCV has the best robustness among other compared CHTs. Figure 5.11a shows

the distribution of the optimal solution found by each CHT along the 30 runs. The optimal solution is considered ∞ when the solver fails to find a feasible solution, this explains why some boxplots are not continuous. It is clearly observed that NSCV finds the best optimal solution among other CHTs. Figure 5.11b shows the density of feasible solution in the last generation of each CHT. NSCV has the highest density after VCH. Figure 5.11c shows the number of generations evolved before finding the first feasible solution. It is clearly shown that MCR is the fastest CHT to find the first feasible solution. Nevertheless, NSCV still has relatively quick performance, where its median is less than 100 of 500 generations.

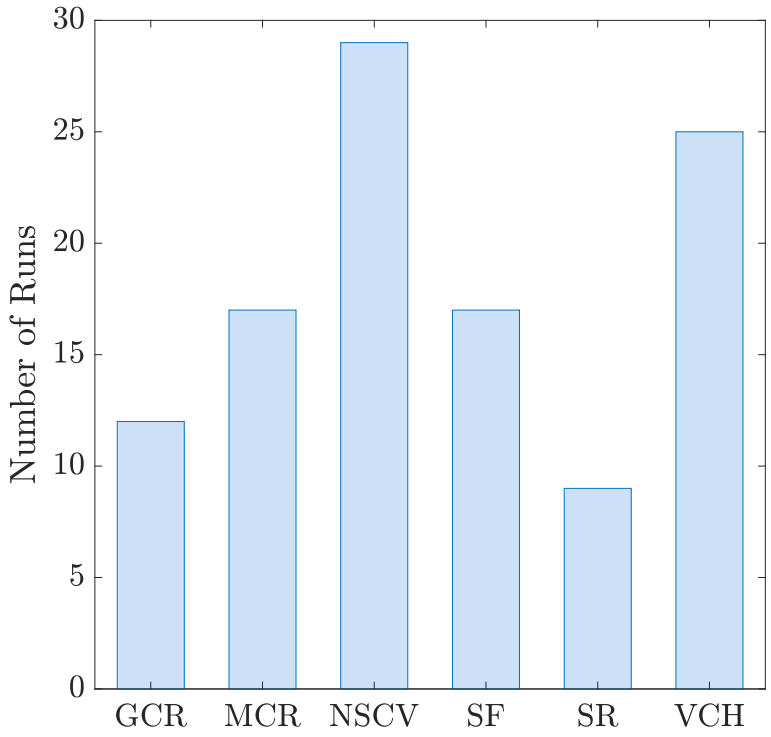
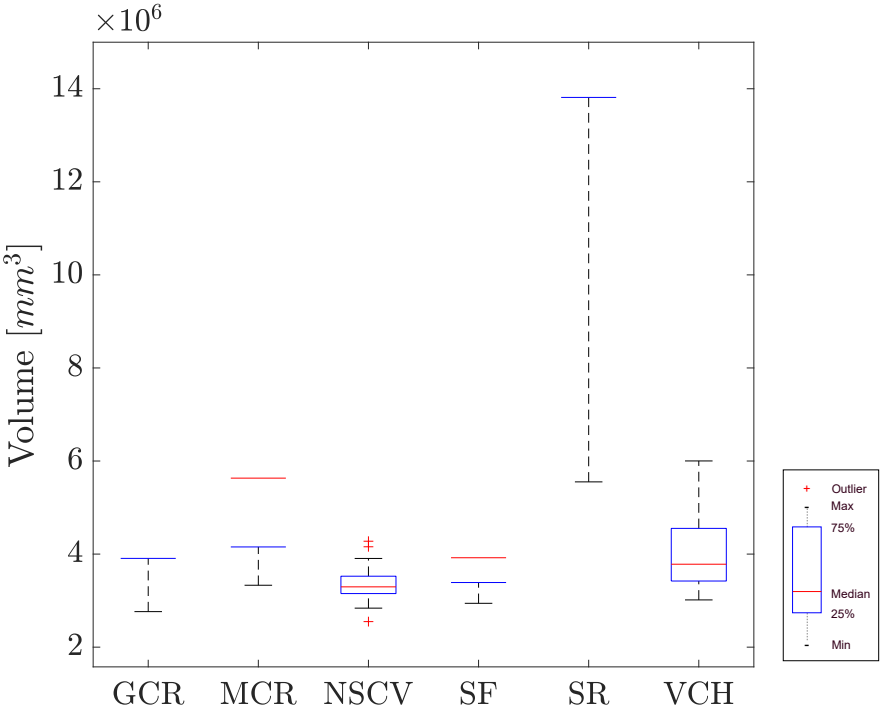
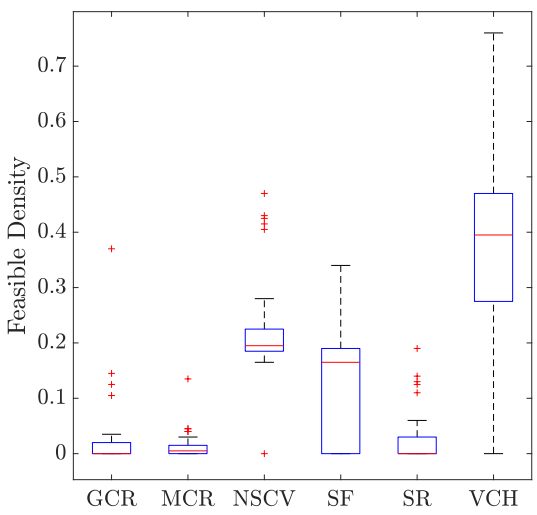


Figure 5.10: Number of successful runs of each CHT on 3SR problem

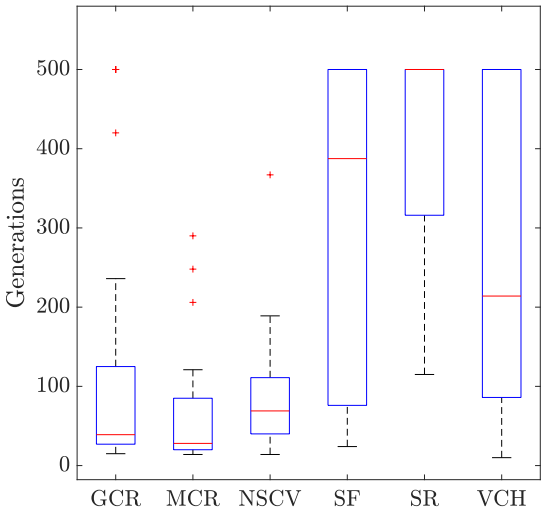
As a result, NSCV confirms its competitive performance on a real-world state-of-art COP regarding both accuracy and robustness. It is interesting to note that the results obtained on the 3SR problem match those obtained on the benchmark problems. In particular, VCH records significantly the highest feasibility density in the last generation. Moreover, CHTs based on feasible separation have generally better performance than other CHTs.



(a) Optimal solution obtained by each CHT for 3SR problem



(b) Mean feasible density in the last generation on 3SR problem



(c) Number of generations evolved to find first feasible solution for 3SR problem

Figure 5.11: Results of the 6 CHTs on the mono-objective version of the 3SR problem.

5.5 An Application of BnB-NSGAII: Seek Optimality

The performances of BnB-NSGAII, BA, BP and NSCV have been examined independently in Chapter 3. The impact of the Legacy feature on the solution is also examined in Section 5.3. In this section, the overall performance of all features implemented in one solver is examined on 3SR problem. Using the same parameters as in Table 5.1, BnB-NSGAII with NSCV as CHT and Pareto Branching (BP) as branching strategy with the legacy feature is run 10 times. The overall performance is compared to the performance of NSGAII with feasible initial population (see Section 5.3). Each run is limited to 24 hours which is considered an acceptable solving duration for such problem.

Starting from infeasible initial population, BnB-NSGAII converges to a feasible Pareto front for all runs. Hence, NSCV additionally enhances the performance of BnB-NSGAII with legacy feature in terms of (1) guiding the exploitation force of BnB-NSGAII; (2) increasing the robustness of BnB-NSGAII which lead to 100% successful runs. This enhancement is strictly due to the presence of NSCV, since, first it is proved in section 5.4 that NSCV enhances the performance of the genetic algorithm on the 3 SR problem; second, the branching strategy affects the number of nodes in MCBB search tree only. Hence, the additional 10% - the difference between successful runs in this experiment and the one in section 5.3 - are due to the presence of NSCV.

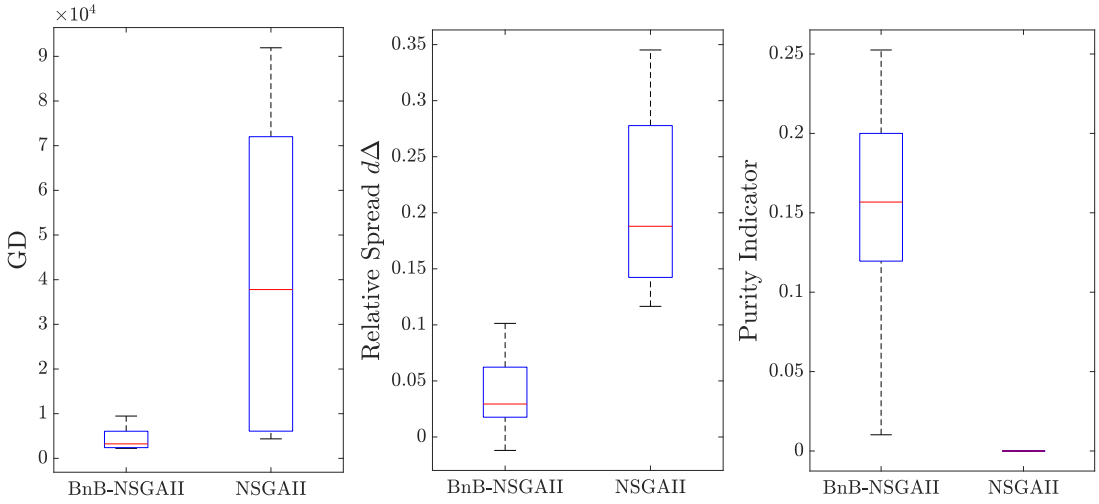


Figure 5.12: Distribution of the quality metric over the 10 runs.

The accuracy metrics defined in Section 4.2 are utilized in this experiment to quantitatively compare the performances of the solvers. However, those metrics require the true Pareto front which is not known for 3SR problem. Hence, the true Pareto front is considered the set which consists in the union of all Pareto elements obtained by both solvers during the 10 runs after

removing the non-dominated elements. Figure 5.12 shows the distribution of GD[1], relative spread $d\Delta$[2] and Purity indicator with respect to the 10 runs. The overall performance of the proposed features significantly overcomes the performance of NSGAII regarding the three accuracy metrics robustly. It should be noted that the value of the purity indicator of NSGAII is 0 in all the runs. Recall that the purity indicator represents the ratio of the number of the non-dominated solutions obtained by NSGAII with respect to the total number of solutions in the true Pareto set. This means that NSGAII failed to capture any solution that dominates the solutions obtained by the proposed algorithm for all runs.

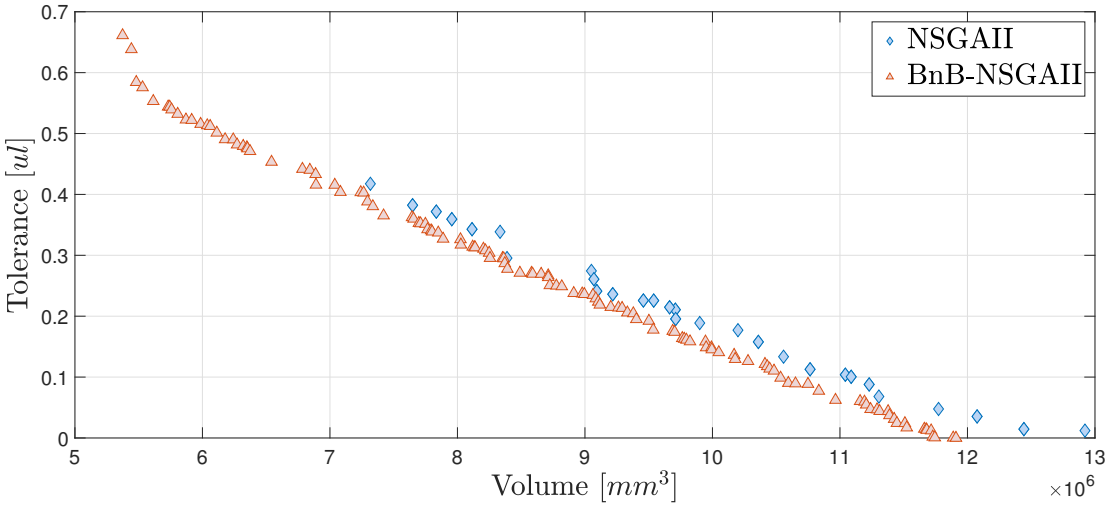


Figure 5.13: Best to Best comparison between the Pareto fronts obtained by BnB-NSGAII (red) and NSGAII (blue).

Figure 5.13 compares the best solution obtained by NSGAII to the best solution obtained by BnB-NSGAII. Table 5.2 shows the anchor points of the Pareto front obtained by each solver. The Pareto front obtained by BnB-NSGAII obviously dominates the Pareto front resulted by NSGAII. Moreover, BnB-NSGAII explores part of the Pareto that NSGAII failed to capture it. This verifies the high exploration potential of BnB-NSGAII.

The computational effort is excluded from this experiment since, to this end, it is well-known that BnB-NSGAII consumes more effort than NSGAII. However, in Section 4.2 we proved that the additional computation effort consumed by BnB-NSGAII is a good investment. As a result, the proposed algorithm, i.e. BnB-NSGAII with legacy feature and NSCV as CHT and the proposed branching strategies, shows significant enhancement of the solution of the real-world state-of-art 3SR problem compared to NSGAII that failed to capture part of the solution that might be important for the decision maker. Hence, BnB-NSGAII is competitive method to

[1]Generational distance measures the distance of the obtained solutions w.r.t. the ideal Pareto front.
[2]Spread measures the diversity of the obtained Pareto front.

solve complex real-world non-convex MO-MINLP problems with huge combinatorial space with a certain predefined time limit.

Table 5.2: Anchor points of the best Pareto front obtained by each solver for the 3SR problem.

Anchor	Objective	NSGAII	BnB-NSGAII
1	Volume $[mm^3]$	7.32×10^6	5.37×10^6
	Tolerance ratio [ul]	0.42	0.66
2	Volume $[mm^3]$	1.29×10^7	1.19×10^7
	Tolerance ratio [ul]	0.012	0

5.6 Conclusion

In this chapter, we have presented a state-of-art 3 stages reducer problem (3SR). In this thesis, the 3SR problem has been formulated to a bi-objective problem. We have added additional constraints on the 3SR problem to meet the requirement of ISO standards on calculation of load capacity of gears [ISO63361-ISO63366]. BnB-NSGAII has been tested on the 3SR problem with and without the legacy feature to examine the impact of this feature on the performance of BnB-NSGAII. The results show that the legacy feature guides the exploration force of BnB-NSGAII resulting in more efficient searching of the domain space.

Our proposed constraint handling technique - NSCV - is tested on a mono-objective version of the 3SR problem as a case study to examine the performance of NSCV on such complex problem.

In this chapter, we have proposed two applications of BnB-NSGAII. The first application - Search Feasibility - aims to reduce the computational effort needed for BnB-NSGAII by using it as an initializer for another metaheuristic (NSGAII in our case). The role of BnB-NSGAII in this application is to search for feasible individuals only. Whenever a feasible individual is found, BnB-NSGAII stops and injects this individual(s) into the initial population of NSGAII.

The second application - Seek Optimality - aims to solve the problem by BnB-NSGAII itself. In this application, NSCV is implemented in BnB-NSGAII as the constraint handling technique. And the branching by Pareto elements strategy is utilized in BnB-NSGAII. The results show that the performance of BnB-NSGAII on finding feasible individuals is furthermore enhanced, this is due to the existence of NSCV. Therefore, the competitive performance of NSCV is verified on a multi-objective problem. Moreover, the performance of BnB-NSGAII is compared to that of NSGAII. BnB-NSGAII significantly overcomes NSGAII in terms of the quality of the solution.

Chapter 6

Conclusion

The lack in literature of hybridization of multi-criteria branch and bound (MCBB) with meta-heuristics has been noticed, in contrary to the mono-objective branch and bound. We have proposed a general-purpose multi-criteria branch-and-bound based on genetic algorithm (NSGAII) for solving non-convex multi-objective Mixed Integer Non-Linear Programming problems (MINLPs), in particular mechanical design problems. The proposed algorithm and NSGAII were tested on 7 MO-MINLP problems from literature. Results show that the proposed algorithm obtain better solutions for all the problems, although in some cases the computational effort was higher than those for NSGAII. In this aspect, we have designed a new metric that relates the quality of the solution to the cost, which referred to as the investment ratio. The results show that the solution(s) has been enhanced by a higher ratio than the computational cost ratio for 6 out of 7 problems, which prove the effectiveness of the proposed hybrid.

BnB-NSGAII has been enhanced by adding the legacy feature. The legacy feature is a generic feature that can be implemented in any branch and bound algorithm. Any parameter that is tuned during the node solving process could be the legacy. In this work, the legacy was the last population in the father node in BnB-NSGAII. The latter was then used to initialize the child node.

We have proposed new branching strategies "Branching by Anchor points" (BA) and "Branching by Pareto elements" (BP) to be implemented in the Multi-criteria Branch and Bound algorithm (MCBB). The proposed strategies are based on interval branching concept that benefit from the solution of the parent node to create the children nodes. A numerical experiment has been designed to examine the performance of the proposed strategy with respect to the traditional strategy "Branching by Integer" (BI) on 5 mechanical engineering and 2 mathematical optimization problems. The experiment has also studied the effect of integer variable separation order on the performance of each strategy. Additionally, a new metric - Winning Ratio WR - is defined that measure the probability of each strategy to performs better than others for a

randomly selected separation order. Results have shown the positive impact of the proposed strategy on MCBB convergence. Moreover, it has been noticed that the proposed strategy are relatively less affected by the separation order than BI.

This work has proposed also a new ranking-based constraint handling technique based on the non-dominated sorting of constraint violation - NSCV. The technique has been specifically designed to handle engineering optimization problems with different orders of magnitude of constraints. NSCV is an uncoupled approach, hence, it can be implemented in any EA. NSCV is simple to implement and does not require any user intervention. NSCV was tested on two sets of benchmark problems. The results showed the competitive performance of NSCV against 5 state-of-art ranking-based CHTs.

The 3 stage reducer problem is a point of interest of many researchers, either to use/ enhance it for engineering applications, or to examine the performance of optimization methods. The 3SR problem is desirable for such experiments for its complexity. In this work, the 3SR problem was reformulated to a bi-objective problem with additional constraints to meet the ISO standards to demonstrate two applications of BnB-NSGAII. The first application is to use BnB-NSGAII as an initiator of NSGAII, where BnB-NSGAII initially seeks feasible individuals before injecting them into the initial population of NSGAII. The second application is to solve the 3SR problem using BnB-NSGAII.

The performances of NSGAII and BnB-NSGAII with and without the legacy feature were tested on the bi-objective version of the 3SR problem. Results show that the legacy feature guides the exploration force of BnB-NSGAII leading it to a better solution than that obtained by NSGAII and BnB-NSGAII.

6.1 Perspectives

BnB-NSGAII is hybrid approach that is based on MCBB and multi-objective genetic algorithm NSGAII. The work of Cacchiani & D'Ambrosio [2017] has been the main motive of BnB-NSGAII. In this thesis, we have utilized the same fathoming rules as Cacchiani & D'Ambrosio [2017]. However, these fathoming rules have been proposed by Mavrotas & Diakoulaki [1998] who proposed to solve the node repeatedly for each objective independently. Consequently, the upper bound is a set of anchor points in contrary to BnB-NSGAII and Cacchiani & D'Ambrosio [2017] where the upper bound is the resulted Pareto front itself. In [Mavrotas & Diakoulaki, 1998], the ideal point is the best known lower bound in contrast to the case of BnB-NSGAII. Therefore, future work is devoted to propose new fathoming rule(s) to tighten the lower bound. Figure 6.1 shows an example of a Pareto front obtained for a specific node (red) and the current incumbent list (black). In this example, all the Pareto elements obtained for this node are dominated by the incumbent list, nevertheless, the node is not fathomed because the ideal point of this front is not

dominated. Thus, a new fathoming rule can consider this scenario and tighten the lower bound to fathom this node and to accordingly reduce the exploration effort of BnB-NSGAII.

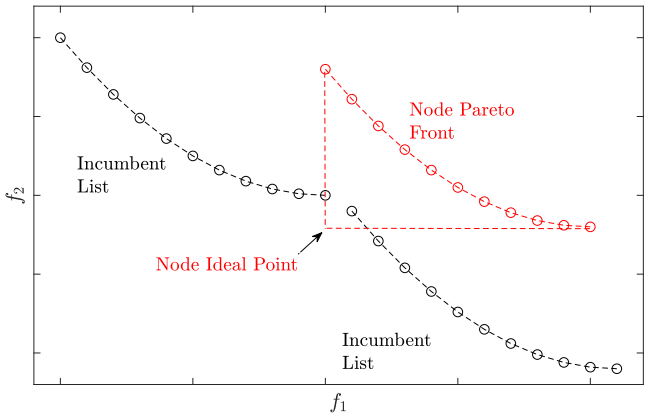


Figure 6.1: Example that show the need for new fathoming rule.

In this thesis, new interval branching strategies has been proposed for MCBB. Although these strategies enhance the performance of MCBB, they branch the integer domain iteratively until all the remaining nodes have single value for each integer variable. This exhaustive branching may considerably consume high effort. Therefore, future work is devoted to find a new fathoming rule that depends on (1) the node bounds and (2) the size of the intervals created by the proposed branching strategies. Hence, a node could be considered "fully explored" before it becomes a singleton node (leaf).

NSCV is proposed in this work as an "uncoupled" approach which could be implemented in any evolutionary algorithm (mono or multi-objective EA). In fact, in this thesis, we have implemented NSCV in both mono-objective and multi-objective genetic algorithms. However, the majority of the results have been obtained on a mono-objective benchmark. Moreover, NSCV is compared to 5 mono-objective CHTs in this experiment. Therefore, future work is devoted to compare the performance of NSCV to wider set of CHTs that not necessarily fall in "ranking-based methods" category. Moreover, future studies include the implementation of NSCV in multi-objective EAs comparing it with other multi-objective CHTs proposed in literature.

6.2 Publications

- A Branch and Bound Based on NSGAII Algorithm for Multi-Objective Mixed Integer Non-Linear Optimization Problems.Ahmed JABER, Pascal LAFON, Rafic YOUNES. Published in Engineering Optimization, April 2021.

- An Application of BnB-NSGAII: Initializing NSGAII to Solve 3 Stage Reducer Problem. Ahmed JABER, Pascal LAFON, Rafic YOUNES. Published in OLA2021 International Conference Proceeding, Sicilia, Italy, June 2021.
- Interval Branching Strategies for Multi-Objective Branch and Bound. Ahmed JABER, Pascal LAFON, Rafic YOUNES. Submitted to Computational Optimization and Applications, June 2021.
- A Ranking Based Constraint Handling Technique For Evolutionary Algorithms: Non-Dominated Sorting of Constraint Violation. Ahmed JABER, Rafic YOUNES, Pascal LAFON. Submitted to Transactions on Evolutionary Computation, May 2021.

Part II

French version

Résumé étendu en Français

Contents

1 Introduction 130
2 État de l'art 131
2.1 Techniques de manipulation des variables mixtes 132
2.2 Techniques pour manipuler les contraintes 137
3 Hybridation "Branch and Bound" et algo. génétique 139
3.1 Principes de l'algorithme BnB-NSGAII 139
3.2 Stratégies de Branchement 140
3.3 NSCV 142
4 Expériences numériques 142
4.1 Algorithme BnB-NSGAII 143
4.2 Stratégies de branchement 145
4.3 Technique NSCV 146
5 Applications : réducteur à engrenages à 3 étages 147
5.1 Introduction 147
5.2 Recherche de solutions faisables 148
5.3 Recherche de solutions optimales 149
5.4 Technique de gestion de contraintes NSCV 150
6 Conclusion 150

1 Introduction

Les problèmes d'optimisation issus du domaine de l'ingénierie mécanique et des matériaux sont souvent difficiles à résoudre. Cela est dû aux caractéristiques de ces problèmes d'optimisation. Par exemple, les problèmes d'optimisation mécanique sont généralement non linéaires et parfois non convexes. Ces problèmes se caractérisent par plusieurs objectifs contradictoires (antagonistes), ainsi que par un ensemble de contraintes qui peuvent rendre difficile pour l'algorithme d'optimisation la localisation du domaine de faisabilité du problème.

Une autre caractéristique qui distingue ces problèmes, est la présence de variables d'optimisation continues et discrètes à la fois. Avec systématiquement, l'impossibilité de relaxer les variables discrètes, c'est à dire de les traiter provisoirement comme des variables continues. Le problème d'optimisation est appelé alors : problème à variables mixtes. Les variables mixtes rendent l'espace de solution disjoint, et donc le problème d'optimisation plus difficile à résoudre.

La résolution de ce type de problèmes, c'est-à-dire les problèmes de programmation non linéaire multi-objectifs en variables mixtes (MO-MINLP), est un centre d'intérêt pour de nombreux chercheurs sachant qu' ils apparaissent dans plusieurs domaines applications, et en particulier dans le domaine du génie mécanique.

Dans ce travail, une approche hybride générale est proposée et basée sur une méthode de branchement et évaluation mutli-critères (MCBB : multi-criteria branch and bound) et l'algorithme Non-dominated Sorting Genetic Algorithm 2 (NSGAII) pour améliorer la recherche du front de Pareto approché de ces problèmes MO-MINLP. Cette approche hybride sera nommée : BnB-NSGAII. En outre, une nouvelle fonctionnalité est également proposée pour améliorer la méthode BnB-NSGAII, appelée fonctionnalité héritée. La fonctionnalité héritée permet l'héritage des meilleurs individus de la population entre les nœuds parents et enfants de la méthode branch and bound. Un des composants principal de tout algorithme branch and bound est la stratégie de branchement. La stratégie de branchement est une tâche cruciale qui affecte les performances d'un méthode MCBB. Dans ce travail, de nouvelles stratégies de branchement - "Branchement par points d'ancrage" et "Branchement par éléments de Pareto" - sont proposées pour être implémentées dans MCBB . Les stratégies proposées sont basées sur le concept de branchement par intervalles en utilisant la solution du nœud parent pour créer des nœuds enfants.

Les problèmes du monde réel, en particulier les problèmes d'ingénierie, sont généralement des problèmes contraints. Les contraintes peuvent être linéaires, non linéaires, convexes voire non convexes. Les techniques de gestion des contraintes (CHT) ont un impact élevé sur les performances des algorithmes évolutionnaires et par conséquent sur les algorithmes génétiques comme NSGAII. Un axe de recherche important consiste à étudier les techniques basées sur le classement. Dans ce travail, nous proposons une technique de gestion des contraintes (CHT) basée sur le classement de type "tri non dominé" des indicateurs de violation des contraintes

(NSCV Ranking). NSCV suit une approche « non couplée » où le CHT n'est pas intégré dans l'algorithme d'optimisation.

2 État de l'art

Les applications de l'optimisation sont innombrables. Chaque processus a un potentiel d'optimisation. En fait, Talbi [2009] a déclaré qu'"il n'y a pas d'entreprise qui ne soit impliquée dans la résolution de problèmes d'optimisation". Breitkopf & Coelho [2013] a déclaré que "toute conception de système est un processus d'optimisation". En effet, de nombreuses applications difficiles en science et en industrie peuvent être formulées comme des problèmes d'optimisation. L'optimisation se produit dans la minimisation du temps, des coûts et des risques ou la maximisation du profit, de la qualité et de l'efficacité.

Les méthodes d'optimisation peuvent être classées en fonction du type de problèmes qu'elles tentent de résoudre, ou elles peuvent être classées en fonction du type d'algorithme. Dans cette section, nous présentons une taxonomie qui aborde les classifications des problèmes d'optimisation (P dans la figure 2.4) en fonction de leurs caractéristiques, la classification des méthodes d'optimisation (Exact (E) et Approximative (A) Figures 2.1 et 2.3 respectivement) et la classification des techniques de transformation (T sur la figure 2.5) des problèmes d'optimisation. Dans le cadre de l'optimisation en génie mécanique et plus précisément dans le cadre de la conception de systèmes mécaniques, incluant éventuellement les problèmes d'optimisation des procédés mécaniques, il est à noter que :

- Le problème de conception contient des variables entières (P3.3) qui peuvent se référer soit à un nombre de composants, soit à un nombre d'entrée dans une table de données, permettant la manipulation de « catalogues » de composants mécaniques. Il est donc *parfois impossible de "relâcher" les variables entières en les considérant comme continues.*

- Ces problèmes font généralement intervenir des critères d'optimisation multi (P1.2) (souvent 2 *ou*3 et rarement plus), car ces critères - que le décideur cherche à minimiser - sont liés à la performance du système comme, par exemple, la masse, la rigidité, le coût (quand il peut être estimé de manière réaliste) ou etc. Des modèles physiques pour exprimer la résistance mécanique, la durabilité, la fiabilité, etc. sont normalement utilisés pour exprimer les fonctions contraintes du problème d'optimisation (d'où P2.2). Dans ce contexte, les fonctions objectifs et contraintes sont généralement non linéaires (P4.2) et non nécessairement convexes (parfois P4.2.2). Par conséquent, le problème résultant est un problème de programmation non linéaire multi-objectifs en nombres entiers (MO-MINLP).

- Une autre particularité de ce type de problèmes d'optimisation est qu'ils ont au moins un espace combinatoire de taille moyenne (couple de variables entières acceptant moins

d'une centaine de valeurs). Par conséquent, la solution exacte ne peut pas être calculée en utilisant une simple procédure de dénombrement explicite.

2.1 Techniques de manipulation des variables mixtes

2.1.1 Méthodes exactes

La méthode la plus populaire pour résoudre exactement un problème MO-MINLP consiste à utiliser une technique de scalarisation (T1) pour transformer le problème multi-objectif en un problème mono-objectif. Ensuite, le problème résultant est résolu itérativement à l'aide d'un solveur mono-objectif. La figure 2.7 montre les solveurs mono-objectifs les plus populaires dans la littérature classés par type de problèmes qu'ils ont l'intention de résoudre.

Mavrotas & Diakoulaki [1998] propose un schéma de conversion, autre que la technique de scalarisation, en utilisant le framework branch and bound (E2.1). Intitulé "Multi-Criteria Branch and Bound (MCBB)" [Mavrotas & Diakoulaki, 1998], ce schéma introduit un nouveau paradigme pour convertir MO-MINLP en mono-objectif en construisant un arbre combinatoire. Le concept principal de MCBB est de convertir $\mathcal{P}_{\text{MO-MINLP}}$ en plusieurs problèmes non linéaires en variables mixtes mono-objectifs $\mathcal{P}_{\text{MINLP}}$ et en problèmes non linéaires continus multi-objectifs $\mathcal{P}_{\text{MO-NLP}}$. Ceci est réalisé en construisant un arbre combinatoire qui vise à partitionner le problème du nœud racine, $\mathcal{P}_{\text{MO-MINLP}}$, en un nombre fini de sous-problèmes (nœuds) $N_1, \ldots, N_i, \ldots, N_n$, $i \in \{1, \ldots, n\}$, où n est le nombre total de nœuds. Le partitionnement (branchement) se fait en modifiant les bornes supérieure et inférieure des variables entières $\boldsymbol{y}$. Ensuite, chaque nœud est résolu à plusieurs reprises pour chaque objectif fonction indépendamment en tant que $\mathcal{P}_{\text{MINLP}}$. En résolvant N_i, l'un des éléments suivants est révélé :

- N_i est infaisable, signifie que le solveur n'a trouvé aucune solution qui satisfait toutes les contraintes, c'est-à-dire que le domaine réalisable est vide en raison de la modification des bornes supérieure et inférieure des variables entières. Par conséquent, N_i est élagué (*fathomed*) par *infaisabilité*.

- N_i est faisable, mais, la borne inférieure actuelle P_i^I est dominée par une borne supérieure précédemment trouvée $\mathcal{F}^*$. Par conséquent, N_i est compris par *optimalité*.

- N_i est faisable, et, P_i^I n'est pas dominé par $\mathcal{F}^*$, $P_i^I \leq \mathcal{F}^*$. $\mathcal{F}^*$ est ensuite mis à jour en y ajoutant la borne supérieure actuelle $\mathcal{F}_i^*$.

Si le nœud n'est pas sondé, l'arbre combinatoire est ramifié plus loin en divisant N_i en sous-problèmes plus éloignés, appelés nœuds enfants. Si un nœud ne peut plus être divisé, il est appelé nœud feuille. Les feuilles sont résolues comme $\mathcal{P}_{\text{MO-NLP}}$, puisque toutes les variables entières sont fixes. L'algorithme 3.2 démontre la boucle principale de MCBB.

Dans MCBB, il y a plusieurs composants qui ne sont pas spécifiés, mais qui peuvent avoir des impacts significatifs sur les performances de l'algorithme [Morrison *et al.*, 2016]. Ces composants sont illustrés dans la figure 2.6. A notre connaissance, il existe un manque dans la littérature concernant les stratégies de branchement pour le MCBB.

2.1.2 Inconvénients

Le principal inconvénient des méthodes exactes MINLP est d'abord que le coût de calcul augmente de façon prohibitive lorsque l'espace combinatoire est grand. Un autre aspect lié aux méthodes exactes qui résolvent les MINLP est que la plupart d'entre elles reposent sur le concept de relaxation entière, ce qui peut être indésirable dans certains cas. Si le problème ne peut pas être relaxé, la seule façon de le résoudre est d'énumérer toutes les combinaisons possibles des variables entières qui peuvent conduire à "l'explosion combinatoire" avec l'augmentation de la taille du problème. Dans ce cas, seuls les petits espaces combinatoires pourraient être résolus par de telles une approche. Par conséquent, bien qu'il existe des méthodes exactes pour résoudre les problèmes d'optimisation en nombres entiers mixtes, ces méthodes ont leurs inconvénients. Elles peuvent ne pas toujours être utilisées en raison de la formulation du problème ou du fait qu'elles auraient besoin d'un temps excessif pour résoudre les problèmes. C'est pourquoi les métaheuristiques et autres méthodes approximatives font l'objet d'une attention croissante.

2.1.3 Méthodes Approximatives

Contrairement aux méthodes exactes, les méthodes approximatives permettent d'aborder des instances de problèmes de grande taille en apportant des solutions satisfaisantes dans un délai raisonnable. Il n'y a aucune garantie de trouver des solutions optimales globales ou même des solutions bornées. Les métaheuristiques, les méthodes approchées les plus répandues, ont deux comportements contradictoires dont il faut tenir compte : l'exploration de l'espace de recherche et l'exploitation des meilleures solutions trouvées . En exploration, les régions non explorées doivent être visitées pour s'assurer que toutes les régions de l'espace de recherche sont explorées uniformément et que la recherche ne se limite pas à un nombre réduit de régions. Tandis que, l'exploitation est une recherche locale d'amélioration itérative [Talbi, 2009].

En plus des concepts communs des méta-heuristiques mono-objectifs, une méta-heuristique multi-objectifs contient trois composants de recherche principaux :

- L'attribution d'aptitude : Le rôle principal de cette procédure est de guider l'algorithme qui recherche des solutions optimales de Pareto pour une meilleure convergence. Elle attribue une aptitude à valeur scalaire à une fonction objectif vectorielle.
- Préservation de la diversité : l'accent est mis ici sur la génération d'un ensemble diversifié de solutions Pareto dans l'objectif et/ou l'espace de décision.

- Élitisme : La préservation et l'utilisation de solutions élites (par exemple, les solutions optimales de Pareto) permettent une amélioration robuste, rapide, monotone de la performance d'un méta-heuristique.

Les algorithmes évolutionnaires (EA) sont des méta-heuristiques stochastiques qui ont été appliquées à de nombreux problèmes réels et complexes . Ce sont les plus étudiés parmi les algorithmes basés sur la population (A2.2). Leur succès dans la résolution de problèmes d'optimisation difficiles dans divers domaines (optimisation continue ou combinatoire) a favorisé le domaine connu sous le nom de calcul évolutif.

L'algorithme génétique, présenté dans Algorithm 1, est l'algorithme évolutionnaire le plus populaire. Initialement, une population initiale est générée selon une règle heuristique ou aléatoirement. À chaque génération successive, un pourcentage de la population existante est « sélectionné » pour élever une nouvelle génération. Par conséquent, une fonction de sélection choisit les « parents » sur la base d'un processus de sélection basé sur la forme physique, où les solutions plus convenables sont plus susceptibles d'être sélectionnées. A chaque génération, l'AG utilise la génération actuelle pour créer la nouvelle progéniture qui définira la prochaine génération. L'algorithme appliquera un ensemble d'opérateurs génétiques (croisement et mutation) sur les parents sélectionnés par la fonction de sélection pour générer les enfants. La re-combinaison (ou croisement) est la combinaison d'une paire de parents, analogue à la reproduction biologique. Les enfants mutés sont créés par un changement aléatoire (ou une mutation) des gènes d'un seul parent. Les deux opérateurs génétiques sont essentiels pour le succès de la recherche d'optimisation. La re-combinaison permet à l'algorithme de préserver les meilleurs gènes de différents individus et les recombiner en enfants éventuellement plus en forme. Cela permet un meilleure « exploitation » de l'espace de recherche. Tandis que la mutation augmente la diversité de la population et permet une « exploration » plus poussée du domaine de recherche. Une fonction de sélection élitiste appropriée doit être employée pour éviter la perte de bonnes solutions. La sélection, la re-combinaison et la mutation sont appliqués de manière itérative à la population jusqu'à ce qu'une condition d'arrêt soit satisfaite.

Dans le cadre de l'algorithme génétique multi-objectif, nous nous concentrons sur l'algorithme Non-dominated Sorting Genetic Algorithm 2 (NSGAII), qui est un algorithme génétique multi-objectifs basé sur des critères. Srinivas & Deb [1994] a proposé la NSGA en 1994 pour l'optimisation multi-objectifs. NSGA est différent d'une AG à objectif unique dans la manière dont l'aptitude "fitness" est attribuée aux individus. Afin d'attribuer l'aptitude, la population est d'abord triée en fonction de son niveau de non-domination. Par la suite, toutes les solutions des meilleurs fronts non dominés seront attribuées une meilleure aptitude. Afin de préserver la diversité parmi les solutions de n'importe quel front, une technique de conservation de niche (niche-preservation technique) est utilisée pour dégrader l'aptitude brute des solutions qui sont encombrés par d'autres solutions du même front.

Sur un certain nombre de problèmes de test et sur un certain nombre de problèmes de conception technique, la NSGA aurait trouvé un nombre de solutions non dominées. Cependant, le même auteur améliore l'algorithme pour qu'il soit NSGAII. Bien que la population soit toujours triée sur la base de la non-domination, une procédure de comptabilité spéciale est utilisée afin de réduire la complexité de calcul. De plus, pour chaque génération, la population des parents est combinée avec la population des enfants. Cela permet de comparer les solutions des parents avec la population des enfants, ce qui assure l'élitisme.

2.1.4 Inconvénients

Dans le contexte de l'optimisation du génie mécanique, la littérature ne fournit pas de conclusion claire sur la métaheuristique ayant les meilleures performances. Yildiz *et al.* [2020] a conclu que la meilleure métaheuristique est l'optimisation de la flamme des mites (MFO) sur un benchmark de 6 problèmes de conception d'ingénierie mono-objectif à nombre entier mixte. Abderazek *et al.* [2019] a constaté que la meilleure métaheuristique est l'évolution différentielle mixte (NAMDE) sur un benchmark de 13 problèmes de conception d'ingénierie mono-objectifs variables mixtes. Dhiman *et al.* [2021] a observé qu'EMoSOA - un nouvel algorithme d'optimisation multi-objectifs évolutif de mouette pour l'optimisation globale - est l'algorithme le plus performant sur un benchmark de 4 problèmes de conception d'ingénierie multi-objectifs et variables mixtes. Assiri [2021] a conclu que la métaheuristique butter fly est la meilleure sur un benchmark de 4 problèmes de conception technique mono-objectif. Kaur *et al.* [2020] a développé Emperor Penguin Optimizer (EPO) en termes de capacité de résolution de problèmes multi-objectifs, qui s'intitule Multi-objectif Emperor Penguin Optimizer (MOEPO). Les auteurs ont observé que les résultats des analyses empiriques montrent que l'algorithme proposé est meilleur que les autres algorithmes existants sur 7 problèmes d'ingénierie multi-objectifs en nombres entiers. Par conséquent, il n'est pas clair quelle est la métaheuristique qui a les meilleures performances sur les problèmes d'ingénierie en variables mixtes mono-objectif/multi-objectif.

En fait, Giagkiozis *et al.* [2015] ont revu dans leur travaux récents, la capacité de sept métaheuristiques à traiter des problèmes de variables mixtes. Une échelle allant de 1 à 5 a été donnée pour mesurer les forces et faiblesses relatives des familles d'algorithmes pour les problèmes multi-objectifs avec des variables mixtes. Une valeur de 5 traduit que cette famille d'algorithmes particulière est très bien adaptée à ce type de problème, tandis qu'une valeur de 1 signifie que l'algorithme est mal adapté à ce type de problèmes. La comparaison a montré que si six algorithmes sur sept étaient capables de gérer l'optimisation de variables mixtes, tous ces six algorithmes ont obtenu un score de 1 dans leur test d'adéquation. Socha [2009] n'a mentionné dans sa revue des méthodes MINLP natives que 5 métaheuristiques pouvant gérer les entiers mixtes. L'un d'eux est la recherche génétique adaptative (GeneAS) introduite par Deb [1997]. GeneAS est suffisamment flexible pour gérer de tels problèmes de conception d'ingénierie

MINLP car il utilise une combinaison de GA codés en binaire et en code réel en fonction de la nature des variables de conception. Par conséquent, bien que la nature mixte du problème soit un problème grave en métaheuristique, les AG sont des sortes de métaheuristiques capables de gérer de tels problèmes en fonction du type de technique de codage de variable utilisée dans l'AG. Cette flexibilité est l'une des raisons qui ont valu aux GA leur popularité dans le domaine MO-MINLP/MINLP.

El Samrout [2019] dans sa thèse a examiné cinq métaheuristiques bien connues (A2.2) de la société scientifique d'optimisation combinatoire qui sont : (1) NSGAII ; (2) MOPSO ; (3) Recherche de coucou ; (4) algorithme de chauve-souris ; et (5) algorithme Firefly sur trois problèmes de conception technique MO-MINLP. En comparant la solution obtenue à partir des cinq métaheuristiques au vrai front de Pareto de chaque problème, l'auteur a conclu que « on peut supposer sans risque que les métaheuristiques traditionnelles étaient incapables de résoudre le problème des trois optimisations ». Même dans le cas de la meilleure métaheuristique, qui est NSGAII dans ce cas, toutes les solutions de Pareto n'ont pas été trouvées par la métaheuristique. Même avec l'utilisation de la meilleure métaheuristique pour les trois problèmes d'optimisation, le front de Pareto calculé est encore loin d'être une bonne solution. Le faible pourcentage d'intersection de solutions entre le front vrai et celui calculé pour les trois problèmes vérifie cette conclusion. De plus, les métaheuristiques "traditionnelles" ne sont pas capables, par nature, de traiter des problèmes à variables mixtes contraintes. Seules certaines d'entre elles peuvent être considérées comme des méthodes natives comme mentionné précédemment. Ainsi, la nécessité d'une meilleure approche pour résoudre ce genre de problèmes est maintenant un point d'intérêt pour de nombreux chercheurs. Récemment, une nouvelle ligne de recherche a vu le jour, celle de l'hybridation d'algorithmes distincts. Dans la section suivante, un aperçu de ces hybrides est présenté.

2.1.5 Hybridation

Au cours des dernières années, un nombre impressionnant d'algorithmes ont été signalés qui ne suivent pas purement le paradigme d'une seule méta-heuristique traditionnelle. Au contraire, ils combinent diverses composantes algorithmiques, souvent issues d'algorithmes d'autres domaines de recherche sur l'optimisation. Ces approches sont communément appelées méta-heuristiques hybrides [Blum *et al.*, 2011].

Plusieurs classifications ont été proposées dans la littérature pour les hybrides en fonction du type d'algorithmes combinés, en fonction du but de l'hybridation ou en fonction de leur ordre d'exécution . L'hybridation de méthodes approximatives avec des techniques de recherche arborescente est probablement l'une des lignes les plus populaires de combinaison de différents algorithmes pour l'optimisation. L'un des ingrédients de base d'une technique d'optimisation est un mécanisme d'exploration de l'espace de recherche, c'est-à-dire l'espace des solutions valides

au problème d'optimisation considéré. L'hybridation de méta-heuristiques avec des concepts branch and bound est assez récente [Blum *et al.*, 2008]. D'un côté, il est possible d'utiliser des concepts de branches et de bornes dans les méta-heuristiques basées sur la construction afin d'augmenter l'efficacité du processus de recherche des méta-heuristiques. D'un autre côté, les méta-heuristiques peuvent être utilisées au sein de branch and bound afin de réduire la consommation d'espace et de temps de branch and bound [Blum *et al.*, 2008].

De nombreux chercheurs ont proposé d'utiliser des hybrides de métaheuristiques avec une "branch and bound" mono-objectif et destinés à améliorer les performances des métaheuristiques. D'autres ont suggéré d'utiliser des heuristiques avec une "branch and bound" multi-objectif. Cependant, à notre connaissance, l'absence d'hybridation de la branche multicritères et liée avec les métaheuristiques dans la littérature est remarquée, contrairement à la "branch and bound" mono-objectif et liée comme illustré sur la figure 2.10. Ce manque de littérature peut être dû à l'appréhension d'un effort de calcul élevé. Par conséquent, toute conception d'un tel hybride doit prendre en compte "l'investissement" dans l'effort de calcul. Dans lequel, les hybrides qui combinent MCBB avec métaheuristique devraient consommer plus d'efforts que la métaheuristique elle-même, cependant, que se passe-t-il si cet investissement dans l'effort de calcul est ajouté à la métaheuristique elle-même (en augmentant les générations maximales ou la taille de la population en GA) ? En fait, il s'agit d'une question de recherche ouverte qui devrait être prise en compte lors de l'examen des performances de ces hybrides. De plus, Blum *et al.* [2011] a observé que la littérature montre qu'un certain hybride pourrait bien fonctionner pour des problèmes spécifiques, mais qu'il pourrait mal fonctionner pour d'autres. Néanmoins, il existe des types d'hybridation qui se sont avérés efficaces pour de nombreuses applications. Ils peuvent servir de guide pour de nouveaux développements.

2.2 Techniques pour manipuler les contraintes

Les algorithmes évolutionnaires ont été largement utilisés pour résoudre une grande variété de problèmes du monde réel, notamment dans les domaines de la science, de l'économie et de l'ingénierie Peng *et al.* [2020]. Bien que ces algorithmes aient été conçus à l'origine pour traiter des espaces de recherche sans contraintes Garcia *et al.* [2017]; Peng *et al.* [2020], ils ont été complétés avec succès par des techniques de gestion des contraintes (CHT) pour résoudre des problèmes sous contraintes. Les CHT guident le processus de recherche vers les régions réalisables et fournissent idéalement des solutions qui ne violent aucune contrainte.

Plusieurs méthodes ont été proposées pour gérer les contraintes par les algorithmes évolutionnaires pour résoudre des problèmes d'optimisation sous contraintes. Les méthodes de fonction de pénalité sont parmi les méthodes les plus courantes pour résoudre les COPs Cai & Wang [2006]. Cependant, en raison des difficultés bien connues associées à eux Coello Coello & Mezura Montes [2002], plusieurs chercheurs ont développé un nombre considérable d'approches alternatives pour

gérer les contraintes. Ces approches ont été classées dans Michalewicz & Schoenauer [1996] comme des méthodes basées sur une recherche de solutions réalisables et des méthodes basées sur la préservation de la faisabilité des solutions.

Récemment, un axe de recherche important a vu le jour consistant à étudier les CHTS par classement Peng *et al.* [2020]. Les CHT basés sur le classement trient et classent la population en fonction de la combinaison de la valeur de la fonction objectif et des violations de contraintes. Dans ce contexte, Runarsson & Yao [2000] a proposé la technique de classement stochastique (SR) qui équilibre les fonctions de pénalité et celles objectifs par un paramètre P_f, produisant un classement en comparant des individus adjacents. Plus tard, les mêmes auteurs ont proposé le classement golbal compétitif (Global Competitive Ranking - GCR) [Runarsson & Yao, 2002] qui équilibre toujours l'objectif et la fonction de violation avec le paramètre P_f, et définit une fonction d'aptitude (fitness function) selon deux classements : la fonction objectif et la somme des violations. Ho & Shimizu [2007] a proposé un schéma de classement où les individus sont triés à l'aide d'une fonction définie selon trois classements respectivement: la fonction objectif; les valeurs de violation de contrainte et le nombre de violations contraintes. Plus récemment, Garcia *et al.* [2017] a proposé le classement à contraintes multiples (Multiple Constraint Ranking - MCR). Celui-ci étend l'approche basée sur le classement de nombreux CHT, en construisant plusieurs files d'attente séparées basées sur les valeurs de la fonction objectifs et la violation de chaque contrainte. Chehouri *et al.* [2016] a proposé le Violation Constraint-Handling (VCH). La population est divisée en un ensemble réalisable et un ensemble non-réalisable. L'ensemble réalisable est trié en fonction de R_f. Alors que l'ensemble non-réalisable est trié en fonction de R_{NV}. Ensuite, la population globale est triée par comparaison par paires, où l'individu réalisable est préféré à l'individu non-réalisable. La méthode la plus populaire est la Supériorité du réalisable (Superiority of Feasable - SF) proposée par Deb [2000]. SF classe séparément les solutions en fonction de leurs valeurs de fonction objective ou de leurs valeurs de violation de contrainte. Elle comprend une sélection de tournois binaires, c'est-à-dire une comparaison par paires qui préfère fondamentalement une solution réalisable à toute solution non-réalisable.

Habituellement, la solution optimale globale est située à la limite de la région réalisable d'un COP, où au moins une des contraintes est active. La solution optimale peut exister à la frontière de plusieurs contraintes, c'est-à-dire que plusieurs contraintes sont actives. Ainsi, le nombre de contraintes violées de rang R_{NV} utilisé dans les méthodes Ho et Shimizu, MCR et VCH n'est pas théoriquement un indicateur du meilleur individu.

Bien que la distance entre l'individu et la limite réalisable soit un indicateur plus sensible, dans le cas de contraintes multiples qui peuvent avoir des ordres de grandeurs différents, la fusion de toutes les violations en une seule somme n'est pas numériquement stable. Puisque l'influence des contraintes de magnitude inférieure peut devenir insignifiante Peng *et al.* [2020]. Normaliser ces valeurs est une bonne approche, mais ce n'est pas toujours applicable (voir Section 3.5.1).

La nécessité d'un CHT sans paramètres (contrairement aux méthodes de pénalité, SR et GCR) en plus des préoccupations mentionnées précédemment sont les principales motivations pour proposer un nouveau CHT basé sur le classement.

3 Hybridation "Branch and Bound" et algorithme génétique

3.1 Principes de l'algorithme BnB-NSGAII

Pour déterminer une bonne solution approchée du front de Pareto d'un problème $\mathcal{P}_{\text{MO-MINLP}}$, nous proposons une approche hybride générale basée sur une méthode MCBB et un algorithme génétique de type NSGAII, pour laquelle, à notre connaissance, il s'agit de la première tentative de combinaison. Le succès de l'utilisation des méta-heuristiques avec "branch and bound" en mono-objectif est le principal motif pour les utiliser avec le branch and bound multicritères.

Dans notre proposition, MCBB est utilisée pour améliorer la force d'exploration de NSGAII de l'espace du domaine à variables mixtes en le séparant sur des sous-domaines, puis NSGAII sera utilisé pour "évaluer" chacun d'eux. De cette façon, MCBB guide la recherche en utilisant les bornes inférieures obtenues par NSGAII. Nous appelons cette approche BnB-NSGAII.

$\mathcal{P}_{\text{MO-MINLP}}$ est un problème complexe et coûteux à résoudre. L'idée générale est donc de résoudre plusieurs problèmes plus simples à la place. Une approche triviale consisterait à énumérer toutes les configurations possibles de variables entières. Ainsi pour chaque combinaison, le problème $\mathcal{P}_{\text{MO-MINLP}}$ est converti en $\mathcal{P}_{\text{MO-NLP}}$. Cette technique est applicable lorsque le nombre de configurations est relativement petit, c'est-à-dire lorsque l'espace combinatoire des variables entières est petit. Mais celle-ci est vite limitée, et le temps de calcul augmente de façon prohibitive lorsque l'espace combinatoire est grand. En effet le nombre de combinaisons croît de façon exponentielle en fonction du nombre de variables entières. Pour éviter cela, MCBB est utilisée pour diviser $\mathcal{P}_{\text{MO-MINLP}}$ en partitionnant le problème du nœud racine, $\mathcal{P}_{\text{MO-MINLP}}$, en un nombre fini de sous-problèmes. On peut représenter ce partitionnement par un graphe arborescent où chaque nœud N_i représente un problème $\mathcal{P}_{\text{MO-MINLP}}$.

Le concept principal de MCBB est d'évaluer chaque nœud N_i en résolvant chaque fonction objectif du problème $\mathcal{P}_{\text{MO-MINLP}}$ de N_i indépendamment, c'est-à-dire que MCBB convertit $\mathcal{P}_{\text{MO-MINLP}}$ en p problème MINLP mono-objectif pour chaque nœud.

Dans notre proposition, BnB-NSGAII divise le domaine de $\mathcal{P}_{\text{MO-MINLP}}$ en plus petits sous-domaines en construisant un arbre exploratoire qui vise à augmenter la force d'exploitation de NSGAII. Ici chaque noeud N_i répresentant un problème $\mathcal{P}_{\text{MO-MINLP}}$ "plus petit" est évalué avec l'algorithme NSGAII en résolvant un problème multi-objectif.

Dans NSGAII, la meilleure population est celle de la dernière génération, car elle contient les meilleurs individus parmi toutes les générations précédentes. Dans BnB-NSGAII, chaque nœud est résolu indépendamment.

De manière à conserver la dernière population du nœud précédent, nous proposons de permettre l'héritage entre les nœuds où chaque nœud enfant hérite de la dernière population de son nœud parent. Le nœud fils initialise alors NSGAII par cette population.

Les nœuds enfants sont des sous-problèmes de leur nœud parent. Ainsi, les bornes sur les variables entières du nœud parent sont différentes de celles des nœuds enfants, $\boldsymbol{Y}_{parent} \neq \boldsymbol{Y}_{child}$. Par conséquent, la population initiale est modifiée pour tenir compte des nouvelles bornes du nœud enfant pour initialiser NSGAII. Cette opération sur la population peut entraîner la perte de certains des meilleurs individus.

3.2 Stratégies de Branchement

L'un des principaux composants de MCBB est la stratégie de branchement qui est appelée à la ligne 18 dans l'algorithme 2. La stratégie de branchement sépare chaque nœud en plusieurs nœuds enfants tout en garantissant que l'union des domaines des nœuds enfants est égale au domaine du nœud parent. Bien que toutes les stratégies fonctionnent sur la modification des bornes inférieures et supérieures des variables entières, chacune a sa façon de construire l'arbre combinatoire. Idéalement, la meilleure stratégie consiste à faire converger MCBB avec un effort d'exploration minimal (c'est-à-dire examiner un nombre minimal de nœuds).

La stratégie la plus utilisée dans les algorithmes MCBB (par exemple Cacchiani & D'Ambrosio [2017]; Mavrotas & Diakoulaki [2005]) consiste à fixer consécutivement les variables entières dans un ordre spécifique appelé "ordre de séparation", nous appelons cette approche "branchement par entier" (Branching by Integer - BI). Cependant, les stratégies utilisées dans le cas mono-objectif ne se transposent pas aussi facilement au cas multi-objectif. Des difficultés apparaissent même avec les stratégies les plus basiques. Plus de détails sont disponibles dans Przybylski & Gandibleux [2017].

D'autres stratégies de branchement existent dans la littérature mais elles ne sont pas aussi populaires que BI. On trouve par exemple des stratégies de branchement par intervalles (Interval branching strategies). Le branchement par intervalles est couramment utilisé dans les branches à objectif unique et est utilisé pour la résolution de problèmes d'optimisation non linéaires Araya *et al.* [2019]. Cette stratégie commence par un nœud initial (nœud racine) puis construit l'arbre de recherche en le divisant en plusieurs sous-intervalles. Cela se fait en divisant le domaine d'une variable entière et générant les nœuds enfants dans l'arbre de recherche. La procédure se répète jusqu'à ce qu'un critère de terminaison soit atteint. Généralement, les nœuds ne sont plus séparés lorsque leurs tailles atteint une précision donnée. Dans Araya *et al.* [2019], une stratégie de branchement par intervalles est proposée pour MCBB qui vise à obtenir une bonne enveloppe pour les vecteurs non dominés en divisant chaque domaine du milieu en deux sous-domaines. Bien que cette stratégie de branchement ait été conçue pour des cas multi-objectifs, le point de division statique (point médian) reste le principal problème dans cette approche.

Dans cette thèse, nous proposons deux stratégies de branchement par intervalles multi-objectifs qui divisent le domaine en fonction de la solution obtenue pour chaque nœud dans MCBB. Nous les appelons "Branching by Anchor Points" et "Branching by Pareto Elements".

MCBB résout le nœud courant N_i en déterminant les p points d'ancrage, où p est le nombre d'objectifs dans $\mathcal{P}_{\text{MO-MINLP}}$. Dans la stratégie de branchement par points d'ancrage (Branching by Anchor points - BA), $p+1$ intervalles sont ensuite créés en divisant l'intervalle d'une des variables entières selon l'ordre de séparation défini. L'intervalle de la variable entière sélectionné j^{th} dans le nœud i, $\boldsymbol{y}_j^i$ est divisé comme suit :

$$\begin{aligned}\boldsymbol{y}_j^i = \{\underline{y}_j^1, (y_j^1)_1\} \cup \{(y_j^2)_1, (y_j^2)_2\} \\ \cdots \cup \{(y_j^{i'})_k, (y_j^{i'})_{k+1}\} \cdots \cup \{(y_j^s)_p, \bar{y}_j^s\},\end{aligned} \tag{6.1}$$

où k est la solution de la k^{th} fonction objectif $k \in \{1, 2, \ldots, p\}$. Chaque intervalle $y_j^{i'}$ $\forall$ $i' \in \{1, \ldots, s\}$ représente un nœud enfant avec un total de s nœuds enfants, où $s \leq p+1$. Ensuite, les intervalles sont correctement décalés pour éviter les intersections, c'est-à-dire les solutions redondantes. Par exemple, supposons que $\boldsymbol{y}_j^i = \{0,1\} \cup \{1,2\} \cup \{2,3\}$, limites inférieures de l'intervalle seront décalées de 1 à partir du deuxième intervalle pour obtenir $\boldsymbol{y}_j^i = \{0,1\} \cup \{2,2\} \cup \{3,3\}$. La figure 3.7 montre un exemple de stratégie BA (BA strategy) pour un problème bi-objectif, où les vecteurs variables entiers des points d'ancrage du nœud racine sont $(\boldsymbol{y}_0^0)_1 =$ [1 1 1 1] and $(\boldsymbol{y}_0^0)_2 =$ [2 2 2 2].

Deux types de solveurs peuvent être utilisés pour évaluer (borner) un nœud dans MCBB, un solveur mono-objectif ou un solveur multi-objectif. Le solveur mono-objectif résout le problème pour chaque fonction objectif indépendamment, ce qui donne uniquement les points d'ancrage du front de Pareto du nœud. Alors que dans le cas de BnB-NSGAII, le solveur multi-objectifs renvoie le front de Pareto du nœud.

Le branchement par éléments de Pareto (Branching by Pareto elements - BP) est plus général que la stratégie BA. Cependant, il ne peut être implémenté dans MCBB qu'avec un solveur multi-objectifs. BP est similaire à BA, mais il sépare le domaine en fonction des solutions de Pareto du front de Parto (Définition 1) au lieu des points d'ancrage comme indiqué dans la figure 3.9, où PE est le nombre d'éléments dans $\mathcal{F}_i^*$. Par conséquent, le nombre de nœuds s' issu de BP **dans chaque niveau** est toujours supérieur ou égal à celui de BA, c'est-à-dire $s \leq s'$ puisque les points d'ancrage sont un sous-ensemble de l'ensemble de Pareto. Ainsi, BP offre un potentiel d'exploration plus élevé *à chaque niveau*. Ce potentiel d'exploration est intéressant lorsque MCBB est utilisé avec des solveurs heuristiques ou méta-heuristiques (e.g. BnB-NSGAII, Adiche & Aider [2019]; Cacchiani & D'Ambrosio [2017]). Cependant, cela ne signifie pas que le nombre total de nœuds de BP est plus élevé, puisque tous les intervalles (nœuds enfants) dans les deux stratégies sont ensuite divisés en singletons. Par conséquent, il ne peut pas être prédit quelle stratégie permet d'obtenir une convergence avec un nombre total de nœuds inférieur.

3.3 NSCV

La nécessité d'une nouvelle technique de gestion des contraintes (CHT) a été discutée précédemment. Dans cette thèse nous proposons une nouvelle CHT basée sur le classement en se basant sur le tri non dominé des violations de contraintes (NSCV). Dans NSCV, la population est divisée en ensembles faisables (qui respectent donc les contraintes du problème d'optimisation) et infaisables (qui ne respectent toutes ou partie de ces contraintes). Les individus de l'ensemble faisables sont triés et classés en fonction des valeurs des fonctions objectifs. Ensuite, chaque individu faisable prend un rang R_f. D'autre part, l'ensemble des individus infaisables est traité comme un problème d'optimisation multi-objectifs où les contraintes sont considérées comme les fonctions objectifs. Ensuite, la population infaisables est triée et classée en fonction de la non-domination des individus du vecteur $\boldsymbol{G}$, où,

$$\boldsymbol{G} = g_j(\boldsymbol{x}) \quad j = 1, \ldots, m. \tag{6.2}$$

Par conséquent, par comparaison par paire (pair-wise comparison), chaque individu de l'ensemble infaisables prend un rang R_C basé sur son $\boldsymbol{G}$. Cette comparaison vectorielle entre les violations de contraintes élimine le besoin de normalisation lorsque différents ordres de grandeurs de violations de contraintes existent.

A cette fin, NSCV ne favorise aucune solution sur une autre dans le même front (c'est à dire le même ensemble d'individus non dominé entre eux). Cependant, pour augmenter la pression vers une meilleure solution optimale, nous proposons de trier les solutions qui ont le même rang R_C en fonction de leur rang calculé avec les valeur des fonctions objectifs R_f. Par conséquent, l'algorithme NSCV peut être résumé comme suit. Pour n'importe quelle paires de solutions :

- Si les deux solutions sont faisables, la solution avec un meilleur rang R_f d'aptitude est conservée.
- Si un seul d'entre elles est faisables, la solution faisable est sélectionnée.
- Si les deux sont infaisables, celle avec le rang de violation de contrainte non dominé le plus bas est sélectionnée.
- Si les deux sont infaisables et dans le même front non dominé, celle qui a le meilleur rang R_f est sélectionnée.

4 Expériences numériques

Le théorème « No Free Lunch (NFL) » et sa généralisation sur les performances d'une méthode de recherche d'une solution en optimisation et en apprentissage Schaffer [1994]; Wolpert & Macready

[1997] impliquent que si un algorithme fonctionne bien sur un ensemble de problèmes, il doit être moins performants sur tous les autres problèmes.

Alternativement, pour le paramétrage de l'optimisation : la performance moyenne de tous les algorithmes de non-rééchantillonnage est identique lorsqu'on calcule la moyenne de leur performance sur tous les problèmes possibles. Ces théorèmes et leurs implications soulignent les limites de la généralisation des résultats obtenus sur des "bancs d'essais" de problèmes à des problèmes différents de ceux du banc d'essai. Par ailleurs, il a parfois été avancé que les hypothèses des théorèmes NFL ne sont pas pertinentes pour les applications pratiques issues de problème "industriels". Ces problèmes "industriels" sont susceptibles de posséder une structure exploitable, les théorèmes impliquent également que si la connaissance de cette structure du problème n'est pas incorporée dans un algorithme particulier, aucune assurance formelle n'existe pour indiquer que l'algorithme sera efficace Sala & Müller [2020]. D'un point de vue pratique, le défi pour l'optimisation métaheuristique est de faire correspondre des algorithmes d'optimisation spécifiques avec des problèmes spécifiques ou des classes de problèmes pour lesquels ces algorithmes sont bien adaptés et fonctionnent relativement bien. Pour les problèmes d'optimisation multimodaux non convexes, les méthodes d'analyse des performances des algorithmes théoriques sont très limitées. Par conséquent, des évaluations comparatives ≪ empiriques ≫ - utilisant des expériences numériques sur des problèmes de test ou des fonctions "banc d'essais" - doivent être utilisées Borenstein & Poli [2004]. Ainsi, une étude de généralisation implique la quantification statistique des performances attendues des algorithmes sur une "classe" spécifique de problèmes.

4.1 Algorithme BnB-NSGAII

Pour évaluer les performances de l'algorithme BnB-NSGAII, nous présentons une expérience basée sur une évaluation statistique pour comparer ses performances avec celles de l'algorithme NSGAII seul. Les deux algorithmes sont testés sur sept problèmes MO-MINLP de la littérature. Les solutions "théoriques" de Pareto de ces problèmes sont connues, de sorte que les fronts de Pareto résultant des deux algorithmes peuvent être comparés aux solutions "théoriques". Nous présentons des critères d'évaluation issus de la littérature pour une comparaison quantitative des performances.

Les performances NSGAII et BnB-NSGAII sont affectées par la sélection de différentes valeurs de paramètres de réglage. Dans le but d'une comparaison impartiale entre eux, différents paramètres de réglage sont appliqués à chaque méthode sur chacun des sept problèmes indépendamment. Dans cette expérience, l'effet du réglage des paramètres sur NSGAII et BnB-NSGAII est testé de manière détaillée, en particulier pour la taille de la population et le nombre de générations en raison de leur effet direct sur la qualité de la solution et l'effort de calcul nécessaire. Ce test est réalisé en faisant varier ces paramètres de manière itérative pour des meilleures performances

de NSGAII et BnB-NSGAII. Le test considère les trois niveaux de réglage de BnB-NSGAII mentionnés précédemment. Le tableau 4.2 présente les valeurs des paramètres distincts utilisés pour les deux algorithmes.

Les méthodes d'optimisation méta-heuristiques sont souvent utilisées pour trouver un compromis entre le coût de l'optimisation (puisque généralement celui-ci est déterminé avant l'optimisation, par exemple en fixant le nombre de générations pour un algorithme génétique) et la précision (en référence à l'optimalité globale que l'on souhaiterait obtenir). Bien qu'une grande variété d'algorithmes d'optimisation ait été développée, il reste difficile d'identifier les algorithmes les plus efficaces pour un nouvel exemple de problème d'optimisation de type "boîte noire" Sala & Müller [2020]. L'évaluation de ce compromis (coût de calcul/qualité de la solution) nécessite une nouvelle métrique qui mesure l'amélioration potentielle de la solution pour un certain supplément de coût de calcul. Dans notre travail, la qualité est définie par la précision de la solution, tandis que le coût est défini par l'effort de calcul. De ce fait, nous proposons un indicateur nommé "ratio d'investissement (IR)" afin d'évaluer l'augmentation de la qualité par rapport à l'augmentation du coût. Cet indicateur est conçu pour considérer divers cas et calculé suivant l'équation 4.7. Une performance de IR$\geq$ 1 indique un bon investissement, c'est à dire un gain en qualité de la solution plus important que le surcoût nécessaire.

La distance générationnelle (GD), l'écart relatif (dΔ) et l'indicateur de pureté (voir la section 4.2) sont utilisés comme indicateurs de qualité. Le coût est défini par le nombre d'évaluations de fonctions objectifs. Pour examiner la robustesse des résultats, chaque problème est résolu 20 fois par chaque méthode et pour chaque combinaison.

Les figures 4.2 à 4.8 montrent la distribution de GD, l'écart relatif (dΔ), l'indicateur de pureté ainsi que le nombre d'évaluations sur les 20 itérations pour chaque combinaison de paramètres, ce dernier étant représenté par le numéro ID. Comme prévu, les résultats montrent que le réglage des paramètres a un effet significatif sur les performances de NSGAII et BnB-NSGAII. En comparaison à NSGAII, BnB-NSGAII offre de meilleures performances en termes d'indicateurs de GD et de propagation pour tous les problèmes, à l'exception du problème de treillis pour lequel BnB-NSGAII donne des solutions plus dispersées. En ce qui concerne l'indicateur de pureté, BnB-NSGAII surpasse NSGAII pour tous les problèmes à l'exception du problème Mela qui a un taux légèrement inférieur.

Par conséquent, pour évaluer les performances de BnB-NSGAII avec l'indicateur IR, les meilleures performances de NSGAII en termes de convergence et d'uniformité des solutions sont sélectionnées pour chaque problème. Le ratio d'investissement IR sera calculé pour chacune des combinaisons donnant les meilleurs performances de NSGAII. Les résultats du tableau 4.5 montrent que BnB-NSGAII est capable de fournir un "bon investissement" (IR $>$ 1) pour tous les problèmes sauf pour le problème de treillis. Par conséquent, l'efficacité de BnB-NSGAII est justifiée pour six problèmes contre sept.

La figure 4.9 montre la meilleure solution obtenue par les deux méthodes par rapport au vrai front de Pareto pour chaque problème. Nous remarquons que BnB-NSGAII est capable de converger vers le vrai front de Pareto pour tous les problèmes. Par contre, NSGAII n'a pas réussi à capturer toutes les solutions de Pareto pour les problèmes d'engrenages et de roulements, comme indiqué dans les figures 4.9a et 4.9b, respectivement. Un examen plus détaillé du front de Pareto du problème de Tong, Figure 4.9g, nous constatons que NSGAII ne capture pas la section du front de Pareto proche des points d'ancrage. On peut cependant conclure que les résultats vérifient généralement la bonne performance de BnB-NSGAII par rapport à NSGAII en termes de qualité de la solution.

4.2 Stratégies de branchement

Les performances des stratégies BI, BA et BP sont testées sur les mêmes 7 problèmes de référence. Les problèmes en variables mixtes binaires ne sont pas adaptés pour cette expérience car les performances de toutes les stratégies de branchement sont évidemment les mêmes sur ce type de problèmes. De ce fait, les problèmes en variables mixtes binaires (problèmes Mela et Tong) sont convertis en problèmes mixtes entiers suivant $\boldsymbol{y} = \{0, 1, 2, 3\}$ pour toutes les variables entières des deux problèmes (Mela-v2 et Tong-v2).

Idéalement, la meilleure stratégie de branchement est celle qui peut faire converger MCBB avec un effort d'exploration minimum, c'est-à-dire un nombre minimum de nœuds explorés au total Belotti *et al.* [2009]. Par conséquent, dans cette expérience, les stratégies de branchement sont comparées en se basant sur l'effort de calcul de MCBB qui est évalué par (1) le nombre total de nœuds explorés ; (2) le nombre total de nœuds feuilles obtenus. Le nombre de nœuds feuilles obtenus est indicatif de la force de "stérilisation" d'une méthode MCBB, (i.e un nœud qui ne sera plus séparé car son évaluation montre qu'il ne contiendra pas de solution optimale), est un critère essentiel pour tout algorithme de branchement et de séparation. La force de "stérilisation" d'autant plus forte que le nombre de nœuds feuilles obtenus est faible, ce qui signifie que plus de nœuds sont sondés.

L'ordre de séparation (ordre de sélection des variables entières) affecte de manière significative l'effort d'exploration de MCBB Ghasemi Saghand *et al.* [2019]. Par conséquent, l'expérience numérique qui teste l'efficacité de la stratégie de branchement doit prendre en considération l'effet de l'ordre de séparation. Ainsi, pour une comparaison impartiale dans cette expérience, chaque problème est résolu à plusieurs reprises pour chaque ordre de séparation possible par chaque stratégie. Ensuite, pour chaque problème, une stratégie est considérée comme meilleure lorsque ses performances sont élevées sur un plus grand nombre d'ordres de séparation. Les performances seront évaluées par une métrique normalisée appelée Winning Ratio (WR), qui est le nombre de fois où chaque stratégie gagne (ayant les meilleures performances) sur le nombre total d'ordres de séparation possibles $\mathcal{O}$ pour chaque problème. Pour des résultats plus tangibles,

le nombre de nœuds est normalisé par la taille de l'espace combinatoire $\mathcal{M}$ (nombre total de combinaisons des valeurs possibles des variables entières) dans chaque problème.

La figure 4.10 montre la distribution du ratio total d'exploration de nœuds obtenus pour chaque stratégie selon toutes les combinaisons possibles d'ordres de séparation. Les cases étroites de BA indiquent que BA est comparativement moins dépendant de l'ordre de séparation. BA surpasse la BI dans quatre problèmes sur sept, au contraire, elle est surpassée par la BI dans le reste. Cependant, il reste difficile de déterminer quelle stratégie est la meilleure en termes d'effort d'exploration. Néanmoins, pour chaque ordre de séparation de chaque problème, on peut calculer une probabilité spécifique pour qu'une stratégie soit plus performante que les autres.

La figure 4.11 montre le rapport gagnant (i.e. winning ratio WR) de chaque stratégie sur chaque problème en fonction du ratio d'exploration total des nœuds. Bien que les résultats indiquent qu'aucune stratégie ne peut être considérée comme la meilleure, le WR illustré dans la figure 4.11 montre les performances plus compétitives pour BA et BP par rapport à BI.

La figure 4.12 montre la distribution du taux d'exploration des nœuds feuilles obtenus pour chaque stratégie selon toutes les combinaisons possibles d'ordres de séparation. BA a le ratio d'exploration le plus bas sur tous les problèmes sauf sur le problème Mela-v2. Cela indique que BA a la force d'exploration la plus élevée parmi les stratégies comparées. De plus, dans BA, le nombre de nœuds feuilles est légèrement affecté par l'ordre de séparation sur tous les problèmes par comparaison à BI. BP dépasse BI pour 5 problèmes sur 7, ce qui fait de BP la deuxième meilleure stratégie en termes de potentiel d'exploration. La performance élevée de BA peut évidemment être observée sur la figure 4.13 qui montre le rapport gagnant (WR) de chaque stratégie sur chaque problème en fonction du ratio d'exploration des nœuds feuilles. Par conséquent, BA et BP surpassent la BI en termes de nombre de nœuds feuilles.

4.3 Technique NSCV

La technique de gestion des contraintes (CHT) NSCV est testée sur deux ensembles de problèmes mono-objectifs. Le premier ensemble se compose de 13 problèmes de référence bien connus Runarsson & Yao [2002] et largement adoptés pour la validation des techniques de gestion des contraintes dans les EA Cai & Wang [2006]; Ho & Shimizu [2007]; Runarsson & Yao [2002]. Cet ensemble est appelé G-Problèmes dans ce travail. D'autre part, le second ensemble se compose de 19 problèmes d'ingénierie mécanique du monde réel (RW-Problems) introduits dans Kumar *et al.* [2020]. Les tableaux 4.7 et 4.8 montrent les propriétés distinctes des ensembles G-Problems et RW-Problems, respectivement. Les performances de NSCV sur ces problèmes sont comparées à cinq technique CHT basées sur le classement (mentionnés dans la section 3.5.1) en les implémentant toutes dans le même algorithme génétique mono-objectif. Toutes les techniques CHT sont initialisées par la même population initiale aléatoire. Le tableau 4.9 montre les caractéristiques de l'algorithme génétique utilisé dans cette expérience. Pour évaluer

la robustesse des techniques CHT, chaque solveur est exécuté 10 fois sur chaque problème. Les performances des techniques CHT sont ensuite évaluées sur la base de (1) la meilleure solution optimale obtenue par chaque technique CHT, (2) la densité de solutions réalisables dans la dernière génération (moyenne des 10 exécutions) et (3) le taux d'échec qui est le nombre d'échecs d'un CHT sur le nombre d'exécutions globales de chaque CHT sur chaque ensemble.

La figure 4.15 montre le profil de performances (une méthodologie spécifiquement développée pour évaluer et comparer les performances d'un ensemble de solveurs $\mathcal{S}$ pour un ensemble donné de problèmes d'optimisation $\mathcal{P}$ pour une métrique spécifique μ) des 6 techniques CHT sur l'ensemble G-Problems. Les résultats montrent que la technique NSCV fournit la meilleure solution de 10 problèmes sur 13 avec une tolérance de 1%. De plus, la technique NSCV est finalement capable de résoudre un maximum de problèmes par rapport aux autres techniques CHT avec 11 problèmes sur 13. Cette robustesse de la technique NSCV est en outre indiquée dans la figure 4.16 qui montre le taux d'échec de chaque CHT sur l'ensemble des G-Problèmes. La technique NSCV a le taux d'échec minimum ($< 0,3$), tandis que le reste des techniques CHT ont des taux d'échec entre 0,4 (SF) et 0,7 (SR). Un autre critère intéressant est la densité de solution réalisable dans la dernière génération. La densité moyenne réalisable de toutes les exécutions pour chaque technique CHT est illustrée à la figure 4.17. La technique VCH a la densité maximale réalisable (45%). Le second maximum appartient à la technique NSCV avec une densité réalisable de 40%. Étant donné que nous nous intéressons à la précision ainsi qu'à la robustesse des techniques CHT, la performance de la technique NSCV est considérée comme la meilleure par comparaison aux autres techniques CHT examinées sur l'ensemble G-Problems.

La figure 4.18 montre le profil de performances de chaque technique CHT sur l'ensemble de problèmes RW. Les résultats montrent que la technique NSCV est capable de résoudre un nombre maximum de problèmes (13 problèmes sur 19) avec une tolérance de 1%. La robustesse de la technique NSCV est confirmée dans la figure 4.19, où NSCV a le taux d'échec le plus faible (0,2). La figure 4.20 montre que la technique VCH a de nouveau la densité maximale moyenne réalisable (0,7) dans la dernière génération parmi tous les techniques CHT testées sur l'ensemble de problèmes RW. Toutefois, les meilleures suivantes sont les techniques GCR et NSCV respectivement avec une densité de ≈ 0.5. La technique NSCV démontre une fois de plus ses performances compétitives parmi les 5 techniques CHT basées sur le classement de l'état de l'art en termes de précision et de robustesse.

5 Applications : réducteur à engrenages à 3 étages

5.1 Introduction

Dans Fauroux [1999], la proposition du problème d'optimisation d'un réducteur à 3 étages (3SR) a été introduite pour illustrer la démarche de conception optimale d'un mécanisme de transmission

de puissance. Plusieurs chercheurs dans différents domaines ont abordé ce problème surtout les chercheurs en génie, dans le cadre des applications de génie mécanique. Dans Fauroux & Lafon [2004], le problème est étendu à un problème d'optimisation à variables mixtes. Et récemment, un problème similaire est proposé dans Han *et al.* [2020] pour illustrer l'optimisation du volume et de la conception de la disposition des éléments d'un réducteur à engrenages à 3 étages. Plusieurs méthodes d'optimisation ont été proposées pour traiter ce problème. Dans Yvars & Zimmer [2018], les auteurs utilisent le problème 3SR pour examiner les performances de la méthode de propagation de contraintes Dans Han *et al.* [2020] ce problème est résolu avec un algorithme génétique.

Dans cette thèse, le problème 3SR est formulé avec 2 fonctions objectifs, 41 contraintes, 3 variables catégorielles (modules d'engrenages), 6 variables entières (nombre de dents) et 11 variables continues. Les modules d'engrenages ont 41 possibilités, le nombre de dents du pignon va de 14 à 30 et le nombre de dents de la roue va de 14 à 150. Par conséquent, la taille de l'espace combinatoire du problème 3SR est : $41^3 * (30-13)^3 * (150-13)^3 \simeq 8.7 \times 10^{14}$. Le problème est considéré comme un problème de taille moyenne concernant le nombre de variables et de contraintes, mais avec un espace combinatoire très conséquent, pour lequel une exploration systématique n'est pas envisageable.

Les 41 fonctions contraintes de ce problème augmentent la complexité du problème en rendant plus difficile l'identification du domaine des solutions. Ceci est observé en résolvant le problème avec l'algorithme NSGAII avec différentes conditions initiales. D'une part, NSGAII est initialisé avec un individu faisable. D'autre part, NSGAII est initialisé de manière aléatoire. Chaque configuration a été exécuté 10 fois avec les mêmes paramètres indiqués dans le tableau 5.1. La figure 5.5a montre le nombre d'exécutions nécessaire pour que chaque méthode converge vers une solution réalisable sur 10. Les figures 5.5a et 5.5b montrent que si la population initiale contient au moins un individu faisable, NSGAII converge à chaque fois vers un bon front de Pareto approché. Alors que, si NSGAII est initialisé avec une population aléatoire, NSGAII ne parvient pas à converger vers une solution réalisable, ou il converge vers un front de Pareto de faible qualité.

5.2 Recherche de solutions faisables

L'algorithme BnB-NSGAII se caractérise par un fort potentiel d'exploration. Ainsi, dans cette section, BnB-NSGAII est utilisé pour rechercher au moins une solution faisable pour le problème 3SR. Dans cet objectif, BnB-NSGAII est convenablement modifié pour 1) continuer l'énumération de l'arbre combinatoire même si le nœud racine est infaisable, 2) arrêter chaque fois qu'une ou plusieurs solutions faisables sont identifiées. Ensuite, NSGAII est appelé pour résoudre le problème 3SR en l'initialisant avec la ou les solutions faisables identifiées comme indiqué sur la figure 5.7.

Les algorithmes NSGAII et BnB-NSGAII, avec et sans la fonctionnalité héritée, ont été testés sur le problème 3SR. Chaque algorithme a été exécutée 10 fois. Le test a été fait en utilisant les mêmes paramètres pour les 3 solveurs (NSGAII, Bnb-NSGAII et BnB-NSGAII hérité). Le tableau 5.1 montre les paramètres utilisés dans cette expérience.

Dans cette dernière, l'évaluation des performances de chaque méthode est limitée au nombre de fois que la méthode trouve au moins une solution faisable sur les 10 exécutions. La figure 5.8 montre le nombre de fois où chaque méthode a réussi le test. On peut évidemment conclure que la méthode héritée BnB-NSGAII surpasse les performances de NSGAII et BnB-NSGAII. Il convient de noter que l'effort de calcul n'est pas pris en compte puisque toutes les exécutions convergent dans les 30 minutes. Ce qui est considéré comme un délai acceptable pour un tel problème.

La figure 5.9a montre que NSGAII a exploré un espace relativement restreint du domaine en fonction de la population initiale. Tandis que la figure 5.9b montre que BnB-NSGAII a exploré une partie plus large du domaine. La figure 5.9c montre que la fonctionnalité héritée guide la force d'exploration de BnB-NSGAII vers les solutions réalisables.

5.3 Recherche de solutions optimales

Les performances de BnB-NSGAII, BA, BP et de la technique NSCV ont été examinées indépendamment dans le chapitre 3. L'impact de la fonctionnalité "héritage" sur la solution a également été examiné dans la section 5.3 où les performances globales de toutes les fonctionnalités implémentées dans un solveur ont été évaluées sur le problème 3SR.

En utilisant les mêmes paramètres indiqués dans le tableau 5.1, BnB-NSGAII avec NSCV comme technique de CHT et BP comme stratégie de branchement ajouté à la fonctionnalité héritée est exécuté 10 fois. La performance globale est comparée avec celle de NSGAII avec une population initiale réalisable (voir Section 5.3). Chaque exécution est limitée à 24 heures, ce que nous considérons comme une durée de résolution acceptable pour un tel problème.

En partant d'une population initiale infaisable, BnB-NSGAII converge vers un front de Pareto réalisable pour toutes les analyses. Par conséquent, la technique NSCV améliore en outre les performances de BnB-NSGAII avec une fonctionnalité héritée en termes de (1) guidage de la force d'exploitation de BnB-NSGAII ; (2) augmentation de la robustesse de BnB-NSGAII ce qui conduit à des exécutions réussies de 100 %.

La figure 5.12 montre la distribution de GD, l'écart relatif $d\Delta$ et l'indicateur de pureté par rapport aux 10 exécutions. Les performances globales des fonctionnalités proposées surpassent de manière significative les performances de NSGAII en ce qui concerne les trois métriques de mesure. Il est à noter que la valeur de l'indicateur de pureté de NSGAII est de 0 dans toutes les séries. Rappelons que cet indicateur de pureté représente le rapport du nombre de solutions non dominées obtenues par NSGAII par rapport au nombre total de solutions dans le véritable

ensemble de Pareto. Cela signifie que NSGAII n'a réussi à capturer aucune solution qui domine les solutions obtenues par l'algorithme que nous proposons pour toutes les exécutions.

La figure 5.13 compare la meilleure solution obtenue par NSGAII à celle obtenue par BnB-NSGAII. Le front de Pareto obtenu par BnB-NSGAII domine clairement le front de Pareto résultant de NSGAII. De plus, BnB-NSGAII explore une partie du Pareto que NSGAII n'a pas réussi à capturer. Ceci vérifie le fort potentiel d'exploration de BnB-NSGAII.

5.4 Technique de gestion de contraintes NSCV

La performance compétitive de la technique NSCV sur les problèmes de référence est évaluée sur le problème 3SR dans le but d'identifier l'impact de celle-ci sur la solution du problème 3SR. Pour conserver le même cas test présenté dans la section 3.5.2, le problème 3SR est résolu comme un problème mono-objectif ne prenant que l'objectif de volume. La performance de la technique NSCV est évaluée comparativement à d'autres techniques CHT en résolvant le problème 3SR 30 fois par chaque technique CHT à partir de la même population initiale en utilisant les mêmes paramètres définis dans le tableau 4.2. La figure 5.10 montre le nombre d'exécutions réussies de chaque CHT, c'est-à-dire que le solveur trouve au moins une solution réalisable. La technique NSCV enregistre le plus grand nombre d'analyses réussies avec 29 analyses sur 30, ce qui indique que cette technique possède la meilleure robustesse parmi les autres techniques CHT de la comparaison.

Il est clairement observé sur la figure 5.11a que la technique NSCV trouve la meilleure solution optimale parmi les autres techniques CHT. La figure 5.11b montre la densité de solution faisable dans la dernière génération de chaque technique CHT : NSCV a la densité la plus élevée après VCH. La figure 5.11c montre le nombre de générations nécessaires avant de trouver la première solution faisable. Il est clairement démontré que la technique MCR est la technique CHT la plus rapide pour en trouver une. Néanmoins, la technique NSCV présente toujours des performances relativement bonnes, où sa médiane est inférieure à 100 de générations sur 500.

6 Conclusion

Il a été remarqué dans la littérature le manque de propositions d'hybridation de méthode de séparation/évaluation multi-critères (MCBB) avec des algorithmes méta-heuristiques, contrairement aux méthodes de séparation/évaluation mono-objectif. Nous avons proposé une méthode de séparation/évaluation multi-critères à usage général basée sur un algorithme génétique (NSGAII) pour résoudre des problèmes de programmation non-linéaire en variables mixtes non convexes multi-objectifs (MO-MINLP), en particulier des problèmes de conception mécanique. L'algorithme proposé et NSGAII ont été testés sur 7 problèmes MO-MINLP de la littérature. Les résultats montrent que l'algorithme proposé obtient de meilleures solutions pour tous les

problèmes, bien que dans certains cas, l'effort de calcul a été supérieur à celui de NSGAII. Dans ce travail, nous avons également conçu une nouvelle métrique qui relie la qualité de la solution à son coût d'obtention, que nous avons appelée ratio d'investissement. Il a été observé que les résultats sont améliorés ce qu'atteste un ratio plus élevé que le ratio de coût de calcul pour 6 problèmes sur 7, ce qui prouve l'efficacité de la méthode proposée.

L'algorithme BnB-NSGAII a été amélioré en ajoutant la fonctionnalité "héritage". Cette dernière est une fonctionnalité générique qui peut être implémentée dans n'importe quel algorithme de séparation/évaluation. Toute information générée au cours du processus de résolution d'un nœud pourrait représenter l'héritage. Dans ce travail, l'héritage était la dernière population du nœud père dans BnB-NSGAII. Ce dernier a été ensuite utilisé pour initialiser le nœud fils.

Nous avons proposé de nouvelles stratégies de branchement (séparation) "Branching by Anchor points" (BA) et "Branching by Pareto elements" (BP) qui ont été testées et implémentées dans l'algorithme Multi-criteria Branch and Bound (MCBB). Les stratégies proposées sont basées sur le concept de branchement par intervalles qui utilise de la solution du nœud parent pour créer les nœuds enfants. Une expérience numérique a été conçue pour examiner les performances de la stratégie proposée par rapport à la stratégie traditionnelle "Branching by Integer" (BI) sur 5 problèmes d'ingénierie mécanique et 2 problèmes d'optimisation mathématique.

Nous avons également étudié l'effet de l'ordre de séparation des variables entières sur les performances de chaque stratégie. De plus, une nouvelle métrique - Winning Ratio WR - est définie pour mesurer la probabilité que chaque stratégie soit plus performante que les autres pour un ordre de séparation sélectionné au hasard. Les résultats ont montré l'impact positif de la stratégie proposée sur la convergence de l'algorithme BnB-NSGAII. De plus, il a été remarqué que la stratégie proposée est relativement moins affectée par l'ordre de séparation que BI. Des travaux futurs sont consacrés à la recherche d'un nouvelle règle d'exploration qui dépend (1) des limites des nœuds et (2) de la taille des intervalles créés par les stratégies de branchement proposées.

Ce travail a également proposé une nouvelle technique de gestion des contraintes reposant sur le classement par le tri non dominé des violations de contraintes ; la technique NSCV. La technique a été spécialement conçue pour traiter des problèmes d'optimisation d'ingénierie avec des fonctions contraintes de différents ordres de grandeur numérique. La technique NSCV est une approche non couplée, elle peut donc être mise en œuvre dans n'importe quelle stratégie évolutionnaire (EA) puisqu' elle ne nécessite aucune intervention de l'utilisateur. La technique NSCV a été testée sur deux ensembles de problèmes de référence. Les résultats ont montré la performance compétitive de la technique NSCV contre 5 techniques CHT conventionnelles basées sur un classement. Les travaux futurs seront consacrés à comparer les performances de la technique NSCV à un ensemble plus large de techniques CHT qui ne relèvent pas nécessairement

de la catégorie des « méthodes basées sur le classement ». De plus, les études futures incluront la mise en œuvre de la technique NSCV dans des EA multi-objectifs.

Le problème du réducteur à 3 étages est représentatif des problèmes de conception optimale. Ce type de problème peut être utilisé pour de nombreux objectifs soit pour l'utiliser/l'améliorer les meilleures solutions connues pour des applications d'ingénierie, soit pour examiner les performances des méthodes d'optimisation. Le problème 3SR est intéressant pour de telles expériences de part sa complexité. Dans ce travail, le problème 3SR a été reformulé en un problème bi-objectif avec des contraintes supplémentaires pour intégrer les normes ISO de calcul sur les capacités de charge des engrenages, afin d'illustrer deux applications de notre algorithme BnB-NSGAII. La première application consiste à utiliser BnB-NSGAII comme initiateur de NSGAII, où BnB-NSGAII recherche initialement des individus faisables avant de les injecter dans la population initiale de NSGAII. La deuxième application consiste à résoudre le problème 3SR en utilisant BnB-NSGAII.

Les performances de NSGAII et BnB-NSGAII, avec et sans la fonctionnalité "héritage", ont été testées sur la version bi-objectif du problème 3SR. Les résultats montrent que la fonctionnalité "héritage" guide la force d'exploration de BnB-NSGAII en la conduisant vers de meilleures solutions que celles obtenues par NSGAII et BnB-NSGAII sans cette fonction "héritage".

Bibliography

Abderazek, H., Yildiz, A.R. & Sait, S.M. (2019). Mechanical engineering design optimisation using novel adaptive differential evolution algorithm. *International Journal of Vehicle Design*, **80**, 285. 31, 135

Adiche, C. & Aider, M. (2019). A hybrid method to solve the multi-objective combinatorial auctions chahrazad adiche and meziane aider. *Advanced Studies in Contemporary Mathematics (Kyungshang)*, **29**, 125–146. 55, 141

Androulakis, I.P., Maranas, C.D. & Floudas, C.A. (1995). αBB: A global optimization method for general constrained nonconvex problems. *J Glob Optim*, **7**, 337–363. 24

Araya, I., Campusano, J. & Aliquintui, D. (2019). Nonlinear biobjective optimization: improvements to interval branch & bound algorithms. *J Glob Optim*, **75**, 91–110. 51, 140

Assiri, A.S. (2021). On the performance improvement of Butterfly Optimization approaches for global optimization and Feature Selection. *PLOS ONE*, **16**, e0242612, publisher: Public Library of Science. 31, 135

Audet, C., Le Digabel, S., Cartier, D., Bigeon, J. & Salomon, L. (2018). Performance indicators in multiobjective optimization. Tech. Rep. G-2018-90, GERAD, iSSN: 0771-2440 Publication Title: Les Cahiers du GERAD. 76

Backer, B.D., Furnon, V., Shaw, P., Kilby, P. & Prosser, P. (2000). Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics. *Journal of Heuristics*, **6**, 501–523. 37

Barreto, A.M., Bernardino, H.S. & Barbosa, H.J. (2010). Probabilistic performance profiles for the experimental evaluation of stochastic algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, 751, ACM Press, Portland, Oregon, USA. 97

Beale, E. & Tomlin, J. (1969). Special_facilities_in_a_general_mathematical_progr.pdf. In *Operational Research*, vol. 69, 447–454. 24

Belotti, P., Lee, J., Liberti, L., Margot, F. & Wächter, A. (2009). Branching and bounds tighteningtechniques for non-convex MINLP. *Optimization Methods and Software*, **24**, 597–634. 20, 90, 145

Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J. & Mahajan, A. (2013). Mixed-integer nonlinear optimization. *Acta Numerica*, **22**, 1–131. 22

Ben Hadj-Alouane, A. & Bean, J.C. (1997). A Genetic Algorithm for the Multiple-Choice Integer Program. *Operations Research*, **45**, 92–101, publisher: INFORMS. 60

Berthold, T. (2014). RENS. *Mathematical Programming Computation*, **6**, 33–54. 23

Berthold, T. & Gleixner, A.M. (2014). Undercover: a primal MINLP heuristic exploring a largest sub-MIP. *Mathematical Programming: Series A and B*, **144**, 315–346. 23

Blum, C., Cotta, C., Fernández, A.J., Gallardo, J.E. & Mastrolilli, M. (2008). Hybridizations of Metaheuristics With Branch & Bound Derivates. In C. Blum, M.J.B. Aguilera, A. Roli & M. Sampels, eds., *Hybrid Metaheuristics: An Emerging Approach to Optimization*, Studies in Computational Intelligence, 85–116, Springer, Berlin, Heidelberg. 6, 34, 36, 37, 137

Blum, C., Puchinger, J., Raidl, G.R. & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, **11**, 4135–4151. 32, 33, 34, 37, 38, 136, 137

Boix, M., Montastruc, L., Pibouleau, L., Azzaro-Pantel, C. & Domenech, S. (2010). Multiobjective optimization of industrial water networks with contaminants. In S. Pierucci & G.B. Ferraris, eds., *Computer Aided Chemical Engineering*, vol. 28 of *20 European Symposium on Computer Aided Process Engineering*, 859–864, Elsevier. 19

Bonami, P., Cornuéjols, G., Lodi, A. & Margot, F. (2009). A Feasibility Pump for mixed integer nonlinear programs. *Mathematical Programming*, **119**, 331–352. 23

Bonami, P., Lee, J., Leyffer, S. & Waechter, A. (2011-09). More branch-and-bound experiments in convex nonlinear integer programming. Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division. 91

Borenstein, Y. & Poli, R. (2004). Fitness Distributions and GA Hardness. In X. Yao, E.K. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J.E. Rowe, P. Tiňo, A. Kabán & H.P. Schwefel, eds., *Parallel Problem Solving from Nature - PPSN VIII*, Lecture Notes in Computer Science, 11–20, Springer, Berlin, Heidelberg. 72, 143

Breitkopf, P. & Coelho, R.F. (2013). *Multidisciplinary Design Optimization in Computational Mechanics*. Wiley-ISTE. 10, 131

Burachik, R.S., Kaya, C.Y. & Rizvi, M.M. (2021). Algorithms for Generating Pareto Fronts of Multi-objective Integer and Mixed-Integer Programming Problems. *arXiv:1903.07041 [math]*, arXiv: 1903.07041. 18

Burer, S. & Letchford, A.N. (2012). Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, **17**, 97–106. 23

Cacchiani, V. & D'Ambrosio, C. (2017). A branch-and-bound based heuristic algorithm for convex multi-objective MINLPs. *European Journal of Operational Research*, **260**, 920–933. 19, 20, 37, 45, 47, 50, 55, 124, 140, 141

Cai, Z. & Wang, Y. (2006). A Multiobjective Optimization-Based Evolutionary Algorithm for Constrained Optimization. *IEEE Trans. Evol. Computat.*, **10**, 658–675. 56, 59, 96, 137, 146

Chehouri, A., Younes, R., Perron, J. & Ilinca, A. (2016). A Constraint-Handling Technique for Genetic Algorithms using a Violation Factor. *Journal of Computer Science*, **12**, 350–362. 30, 59, 60, 61, 63, 138

Coello Coello, C.A. (1999). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, **1**, 269–308. 61

Coello Coello, C.A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, **41**, 113–127. 60

Coello Coello, C.A. & Mezura Montes, E. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, **16**, 193–203. 56, 64, 65, 137

Cotta, C. & Troya, J.M. (2003). Embedding Branch and Bound within Evolutionary Algorithms. *Applied Intelligence*, **18**, 137–153. 36

Custódio, A.L., Madeira, J.F.A., Vaz, A.I.F. & Vicente, L.N. (2011). Direct Multisearch for Multiobjective Optimization. *SIAM J. Optim.*, **21**, 1109–1140, publisher: Society for Industrial and Applied Mathematics. 76

Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection*. Murray, London, or the Preservation of Favored Races in the Struggle for Life. 29

De Santis, M., Eichfelder, G., Niebling, J. & Rocktäschel, S. (2020). Solving Multiobjective Mixed Integer Convex Optimization Problems. *SIAM J. Optim.*, **30**, 3122–3145, publisher: Society for Industrial and Applied Mathematics. 19

Deb, K. (1997). GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In D. Dasgupta & Z. Michalewicz, eds., *Evolutionary Algorithms in Engineering Applications*, 497–514, Springer, Berlin, Heidelberg. 32, 135

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, **186**, 311–338. 59, 61, 138

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, google-Books-ID: OSTn4GSy2uQC. 16, 77

Deb, K. & Jain, S. (2003). Multi-Speed Gearbox Design Using Multi-Objective Evolutionary Algorithms. *Journal of Mechanical Design*, **125**, 609. 106

Deb, K., Pratap, A. & Moitra, S. (2000). Mechanical Component Design for Multiple Ojectives Using Elitist Non-dominated Sorting GA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo & H.P. Schwefel, eds., *Parallel Problem Solving from Nature PPSN VI*, Lecture Notes in Computer Science, 859–868, Springer, Berlin, Heidelberg. 73

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182–197. 66, 79, 113

Dhiman, G., Singh, K.K., Slowik, A., Chang, V., Yildiz, A.R., Kaur, A. & Garg, M. (2021). EMoSOA: a new evolutionary multi-objective seagull optimization algorithm for global optimization. *International Journal of Machine Learning and Cybernetics*, **12**, 571–596. 31, 135

Dimkou, T.I. & Papalexandri, K.P. (1998). A parametric optimization approach for multiobjective engineering problems involving discrete decisions. *Computers & Chemical Engineering*, **22**, S951–S954. 75

Dréo, J. & Candan, C. (2007). Different classifications of metaheuristics. xv, 26, 27

Duran, M.A. & Grossmann, I.E. (1986). An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, **36**, 307–339. 22

Ehrgott, M. (2006). A discussion of scalarization techniques for multiple objective integer programming. *Ann Oper Res*, **147**, 343–360. 18

Ehrgott, M. & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, **22**, 425–460. 13, 34

Ehrgott, M. & Gandibleux, X. (2008). Hybrid Metaheuristics for Multi-objective Combinatorial Optimization. In C. Blum, M.J.B. Aguilera, A. Roli & M. Sampels, eds., *Hybrid Metaheuristics: An Emerging Approach to Optimization*, Studies in Computational Intelligence, 221–259, Springer, Berlin, Heidelberg. 33, 35, 37

Ehrgott, M., Gandibleux, X. & Hillier, F.S., eds. (2002). *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, vol. 52 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA. 13, 18

El Samrout, A. (2019). *Hybridization of multicriteria metaheuristic optimization methods for mechanical problems*. Ph.D. thesis, Université de technologie de Troyes, Troyes, France. 6, 32, 33, 34, 36, 45, 73, 92, 136, 167, 169

Fang, Y. & Li, J. (2010). A Review of Tournament Selection in Genetic Programming. In Z. Cai, C. Hu, Z. Kang & Y. Liu, eds., *Advances in Computation and Intelligence*, Lecture Notes in Computer Science, 181–192, Springer, Berlin, Heidelberg. 99

Fauroux, J.C. (1999). *Conception optimale de structures cinématiques tridimensionnelles. Application aux mécanismes de transmission en rotation*. Ph.D. thesis, INSA de Toulouse, Toulouse, France. 107, 147

Fauroux, J.C. & Lafon, P. (2004). Optimization of a Multi-Stage Transmission Mechanism. In *IFToMM*, vol. 2, 813–817. 107, 108, 148

Festa, P. (2014). A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In *2014 16th International Conference on Transparent Optical Networks (ICTON)*, 1–20, iSSN: 2161-2064. 26

Fischetti, M. & Lodi, A. (2011). Heuristics in Mixed Integer Programming. In *Wiley Encyclopedia of Operations Research and Management Science*, American Cancer Society, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470400531.eorms0376. 22

Fletcher, R. & Leyffer, S. (1994). Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, **66**, 327–349. 22

Florios, K., Mavrotas, G. & Diakoulaki, D. (2010). Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. *European Journal of Operational Research*, **203**, 14–21. 38

Garcia, R.d.P., de Lima, B.S.L.P., Lemonge, A.C.d.C. & Jacob, B.P. (2017). A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. *Computers & Structures*, **187**, 77–87. 56, 57, 61, 137, 138

Ghasemi Saghand, P., Charkhgard, H. & Kwon, C. (2019). A Branch-and-Bound Algorithm for a Class of Mixed Integer Linear Maximum Multiplicative Programs: A Bi-objective Optimization Approach. *Computers & Operations Research*, **101**, 263–274. 91, 145

Giagkiozis, I., Purshouse, R.C. & Fleming, P.J. (2015). An overview of population-based algorithms for multi-objective optimisation. *International Journal of Systems Science*, **46**, 1572–1599. 31, 135

Giraud, L. & Lafon, P. (1999). Optimal Design of Mechanical Components with Genetic Algorithm. In J.L. Batoz, P. Chedmail, G. Cognet & C. Fortin, eds., *Integrated Design and Manufacturing in Mechanical Engineering '98*, 93–100, Springer Netherlands. 73

Giraud-Moreau, L. & Lafon, P. (2002). Comparison of evolutionary algorithms for mechanical design components. *Journal of Engineering Optimization*, **34**, 307–322. 73

Gomory, R.E. (1960). Solving linear programming problems in integers. In *Proc. Sympos. Appl. Math.*, vol. 10, 211–215, Providence, R.I. 10

Han, L., Liu, G., Yang, X. & Han, B. (2020). Dimensional and Layout Optimization Design of Multistage Gear Drives Using Genetic Algorithms. *Mathematical Problems in Engineering*, **2020**, 3197395, publisher: Hindawi. 107, 108, 148

Ho, P.Y. & Shimizu, K. (2007). Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. *Information Sciences*, **177**, 2985–3004. 62, 96, 138, 146

Holland, J.H. (1992). Genetic algorithms. *Scientific American*. 29

Homaifar, A., Qi, C.X. & Lai, S.H. (1994). Constrained Optimization Via Genetic Algorithms. *SIMULATION*, **62**, 242–253, publisher: SAGE Publications Ltd STM. 60

International Organization for Standardization (Ginebra) (2019a). *ISO 6336-1: Calculation of Load Capacity of Spur and Helical Gears. Part 1*. ISO. 110

International Organization for Standardization (Ginebra) (2019b). *ISO 6336-2: Calculation of Load Capacity of Spur and Helical Gears. Part 2*. ISO. 110, 111

International Organization for Standardization (Ginebra) (2019c). *ISO 6336-3: Calculation of Load Capacity of Spur and Helical Gears. Part 3*. ISO. 110

International Organization for Standardization (Ginebra) (2019d). *ISO 6336-5: Calculation of Load Capacity of Spur and Helical Gears. Part 5*. ISO. 110

Joines, J.A. & Houck, C.R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, 579–584 vol.2. 60

Jozefowiez, N., Semet, F. & Talbi, E.G. (2007). The bi-objective covering tour problem. *Computers & Operations Research*, **34**, 1929–1942. 37

Kaur, H., Rai, A., Bhatia, S.S. & Dhiman, G. (2020). MOEPO: A novel Multi-objective Emperor Penguin Optimizer for global optimization: Special application in ranking of cloud service providers. *Engineering Applications of Artificial Intelligence*, **96**, 104008. 31, 135

Kiziltan, G. & Yucaoğlu, E. (1983). An Algorithm for Multiobjective Zero-One Linear Programming. *Management Science*, **29**, 1444–1453, publisher: INFORMS. 19

Koziel, S. & Michalewicz, Z. (1999). Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, **7**, 19–44. 57

Kronqvist, J., Lundell, A. & Westerlund, T. (2016). The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. *Journal of Global Optimization*, **64**, 249–272. 22

Kronqvist, J., Bernal, D.E., Lundell, A. & Grossmann, I.E. (2019). A review and comparison of solvers for convex MINLP. *Optim Eng*, **20**, 397–455. 21, 22, 23

Kumar, A., Wu, G., Ali, M.Z., Mallipeddi, R., Suganthan, P.N. & Das, S. (2020). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, **56**, 100693. 96, 146

Land, A.H. & Doig, A.G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, **28**, 497–520, publisher: [Wiley, Econometric Society]. 19

Li, H. & Zhang, Q. (2009). Multiobjective optimization problems with complicated pareto sets, MOEA/ D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, **13**, 284–302. 77

Long, Q. (2014). A constraint handling technique for constrained multi-objective genetic algorithm. *Swarm and Evolutionary Computation*, **15**, 66–79. 61

Maruyama, S. & Tatsukawa, T. (2017). A parametric study of crossover operators in pareto-based multiobjective evolutionary algorithm. In *Advances in Swarm Intelligence - 8th International Conference, ICSI 2017, Proceedings*, 3–14, Springer Verlag. 79, 99, 113

Mavrotas, G. & Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, **107**, 530–541. 19, 124, 132

Mavrotas, G. & Diakoulaki, D. (2005). Multi-criteria branch and bound: A vector maximization algorithm for Mixed 0-1 Multiple Objective Linear Programming. *Applied Mathematics and Computation*, **171**, 53–71. 50, 140

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming*, **10**, 147–175. 23

Mela, K., Koski, J. & Silvennoinen, R. (2007). Algorithm for generating the pareto optimal set of multiobjective nonlinear mixed-integer optimization problems. In *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, vol. 2, 2026–2042, American Institute of Aeronautics and Astronautics, Honolulu, Hawaii. 74

Messac, A. (2015). *Optimization in practice with MATLAB for engineering students and professionals*. Cambridge University Press, New York, NY. 16

Mezura-Montes, E. & Coello Coello, C.A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, **1**, 173–194. xv, 57, 58

Michalewicz, Z. & Janikow, C.Z. (1996). GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints. *Commun. ACM*, **39**, 175–es. 58

Michalewicz, Z. & Schoenauer, M. (1996). Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, **4**, 1–32. 56, 57, 60, 138

Montgomery, D.C. (2017). *Design and Analysis of Experiments*. John Wiley & Sons, googleBooks-ID: Py7bDgAAQBAJ. 78

Morrison, D.R., Jacobson, S.H., Sauppe, J.J. & Sewell, E.C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, **19**, 79–102. 20, 133

Nagar, A., Heragu, S.S. & Haddock, J. (1996). A combined branch-and-bound and genetic algorithm based approach for a flowshop scheduling problem. *Ann Oper Res*, **63**, 397–414. 36

Osyczka, A. & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, **10**, 94–99. 74

Pascoletti, A. & Serafini, P. (1984). Scalarizing vector optimization problems. *Journal of Optimization Theory and Applications*, **42**, 499–524. 19

Peng, C., Liu, H.L. & Goodman, E.D. (2020). Handling multi-objective optimization problems with unbalanced constraints and their effects on evolutionary algorithm performance. *Swarm and Evolutionary Computation*, **55**, 100676. 56, 62, 63, 64, 137, 138

Pomrehn, L.P. & Papalambros, P.Y. (1995). Discrete Optimal Design Formulations With Application to Gear Train Design. *J. Mech. Des*, **117**, 419–424. 16

Przybylski, A. & Gandibleux, X. (2017). Multi-objective branch and bound. *European Journal of Operational Research*, **260**, 856–872. 18, 19, 20, 51, 92, 140

Puchinger, J. & Raidl, G.R. (2005). Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. In J. Mira & J.R. Álvarez, eds., *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, Lecture Notes in Computer Science, 41–53, Springer Berlin Heidelberg. 33, 34, 37

Puchinger, J., Raidl, G.R. & Koller, G. (2004). Solving a Real-World Glass Cutting Problem. In J. Gottlieb & G.R. Raidl, eds., *Evolutionary Computation in Combinatorial Optimization*, Lecture Notes in Computer Science, 165–176, Springer, Berlin, Heidelberg. 37

Rao, R.V. & Savsani, V.J. (2012). *Mechanical Design Optimization Using Advanced Optimization Techniques*. Springer Series in Advanced Manufacturing, Springer London, London. 10

Runarsson, T. & Yao, X. (2002). Constrained Evolutionary Optimization. In R. Sarker, M. Mohammadian & X. Yao, eds., *Evolutionary Optimization*, International Series in Operations Research & Management Science, 87–113, Springer US, Boston, MA. 62, 96, 138, 146

Runarsson, T.P. & Yao, X. (2000). Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, **4**, 11. 59, 61, 138

Ryoo, H.S. & Sahinidis, N.V. (1996). A branch-and-reduce approach to global optimization. *J Glob Optim*, **8**, 107–138. 23

Sala, R. & Müller, R. (2020). Benchmarking for Metaheuristic Black-Box Optimization: Perspectives and Open Challenges. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. 72, 143, 144

Schaffer, C. (1994). A Conservation Law for Generalization Performance. In W.W. Cohen & H. Hirsh, eds., *Machine Learning Proceedings 1994*, 259–265, Morgan Kaufmann, San Francisco (CA). 72, 142

Simionescu, P.A., Beale, D.G. & Dozier, G.V. (2004). Constrained optimization problem solving using estimation of distribution algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, 296–302 Vol.1. 59

Simpson, T.W. & Martins, J.R.R.A. (2011). Multidisciplinary Design Optimization for Complex Engineered Systems: Report From a National Science Foundation Workshop. *Journal of Mechanical Design*, **133**. 16

Socha, K. (2009). *Ant Colony Optimisation for Continuous and Mixed-Variable Domains*. VDM Verlag, Saarbrücken, DEU. 14, 15, 32, 135

Srinivas, N. & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, **2**, 221–248. 30, 42, 64, 134

Surry, P.D., Radcliffe, N.J. & Boyd, I.D. (1995). A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA method. In T.C. Fogarty, ed., *Evolutionary Computing*, Lecture Notes in Computer Science, 166–180, Springer, Berlin, Heidelberg. 64, 65

Talbi, E.G. (2009). *Metaheuristics: from design to implementation*. Wiley, Hoboken, NJ, oCLC: 230183356. 10, 12, 26, 29, 78, 131, 133

Talukdar, S., Baerentzen, L., Gove, A. & De Souza, P. (1998). Asynchronous Teams: Cooperation Schemes for Autonomous Agents. *Journal of Heuristics*, **4**, 295–321. 34

Tong, W., Chowdhury, S. & Messac, A. (2016). A multi-objective mixed-discrete particle swarm optimization with multi-domain diversity preservation. *Struct Multidisc Optim*, **53**, 471–488. 74, 75

Van Veldhuizen, D. & Lamont, G. (2000). On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, 204–211, IEEE, La Jolla, CA, USA. 77

Wang, S. & Liu, M. (2013). A heuristic method for two-stage hybrid flow shop with dedicated machines. *Computers & Operations Research*, **40**, 438–450. 38

Westerlund, T. & Pettersson, F. (1995). An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, **19**, 131–136. 21

Wolpert, D. & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67–82, conference Name: IEEE Transactions on Evolutionary Computation. 72, 142

Woodruff, D.L. (1999). A Chunking Based Selection Strategy for Integrating Meta-Heuristics with Branch and Bound. In S. Voß, S. Martello, I.H. Osman & C. Roucairol, eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, 499–511, Springer US, Boston, MA. 36

Yildiz, A.R., Abderazek, H. & Mirjalili, S. (2020). A Comparative Study of Recent Non-traditional Methods for Mechanical Design Optimization. *Archives of Computational Methods in Engineering*, **27**, 1031–1048. 31, 135

Yvars, P.A. & Zimmer, L. (2018). System Sizing with a Model-Based Approach: Application to the Optimization of a Power Transmission System. *Mathematical Problems in Engineering*, **2018**, 1–14. 107, 148

Črepinšek, M., Liu, S.H. & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, **45**, 1–33. 27

Appendix A

Benchmark Problems

The formulations of the benchmark problems used in this article and their solutions are presented here. The true Pareto solution of these problems is obtained in the following manner, first, the problem is solved by discretization the continuous variables, then the combinations of integer variables that gives part of the Pareto are recorded. By fixing those combinations iteratively, the problem is solved exactly using the scalarization method as $\mathcal{P}_{\text{MO-NLP}}$. Cubic interpolation is then used to obtain a uniform discretization step for each continuous part of the Pareto front. Finally, Pareto refining is applied to remove all dominated solutions.

For the convergence metric GD in the case of continuous/discontinuous Pareto, the value of d_i is affected by the discretization step used to generate the true Pareto front. Thus although the tested solver might find a solution that lies on the theoretical front, the distance d_i might be greater than 0 because of the discretization step. To avoid this, first we chose $\epsilon = 10^{-4}$ as discretization step, then for each n non-dominated solution $d_n \leq \epsilon$ since n must be between ideal and nadir points. Thus we can assume that $d_n = 0$ for a more reasonable approximation of GD. However this assumption might lead to inaccurate GD value, but since we are using this assumption on both compared solvers, we can consider it impartial. For the relative spread metric, to minimize Δ_{True}, the true Pareto is discretized uniformly, i.e. the euclidean distance between every two consecutive solutions laying on a continuous section of the front is constant.

1 Problems Formulation

1.1 Gear Problem

The gear train problem is unconstrained. Since the number of teeth must be integers, all 4 variables are strictly integers. The problem is illustrated in figure A.1 and formulated as following:

$$\begin{aligned} \text{minimize} \quad & f_1 = \frac{1}{6.931} - \left(\frac{z_1 z_3}{z_2 z_4}\right)^2 \\ & f_2 = \max\{z_1, z_2, z_3, z_4\} \\ \text{subject to} \quad & \\ & 12 \leq z_i \leq 60; \quad i = 1, 2, 3, 4 \\ & z_i \in \mathbb{N}; \qquad\quad i = 1, 2, 3, 4 \end{aligned} \tag{A.1}$$

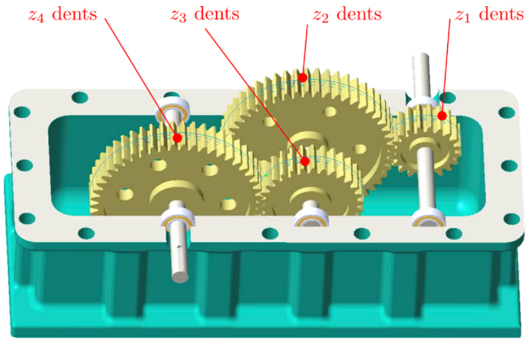


Figure A.1: Illustration of gear problem

1.2 Bearing Problem

In the ball bearing pivot link, the ball bearings were chosen from a standardized table of prefabricated sizes. The formulated optimization problem contains 4 variables, 2 continuous/ 2 integer, 12 discrete parameters and 10 inequality constraints. Figure A.2 illustrates the problem that is formulated as following:

$$\begin{aligned} \underset{i_{r1}, i_{r2}, x_1, x_2}{\text{minimize}} \quad & F_1 = \frac{\pi \rho_a}{4} \left[\phi_3(i_{r1})^2 \phi_5(i_{r1}) + \phi_3(i_{r2})^2 \phi_5(i_{r2}) + \dots \right. \\ & \dots \left(x_1 + x_2 - \frac{\phi_5(i_{r1}) + \phi_5(i_{r2})}{2} \right) \dots \\ & \left. \dots * \max\{\phi_6(i_{r2}), \phi_6(i_{r2})\}^2 + \phi_7(i_{r1}) + \phi_7(i_{r2}) \right) \\ & F_2 = \phi_1(i_{r1}) + \phi_1(i_{r2}) \\ \text{subject to} \quad & \end{aligned} \tag{A.2}$$

$$
\begin{aligned}
C_1 &= (XF_{r1}(x_1, x_2) + YF_{a1}) \left(\frac{60L_v\omega}{10^6}\right)^{\frac{1}{3}} - \phi_1(i_{r1}) \leq 0 \\
C_2 &= F_{r2}(x_1, x_2) \left(\frac{60L_v\omega}{10^6}\right)^{\frac{1}{3}} - \phi_1(i_{r2}) \leq 0 \\
C_3 &= \left(x_{1\text{Min}} + \frac{\phi_5(i_{r1})}{2}\right) - x_1 \leq 0 \\
C_4 &= x_1 - \left(x_{1\text{Max}} - \frac{\phi_5(i_{r1})}{2}\right) \leq 0 \\
C_5 &= \left(x_{2\text{Min}} + \frac{\phi_5(i_{r2})}{2}\right) - x_2 \leq 0 \\
C_6 &= x_2 - \left(x_{2\text{Max}} - \frac{\phi_5(i_{r2})}{2}\right) \leq 0 \\
C_7 &= d_{1\text{Min}} - \phi_3(i_{r1}) \leq 0 \\
C_8 &= d_{2\text{Min}} - \phi_3(i_{r2}) \leq 0 \\
C_9 &= \max\{\phi_4(i_{r1}), \phi_4(i_{r2})\} - D_{\text{Max}} \leq 0 \\
C_{10} &= \phi_4(i_{r1}) - \phi_4(i_{r2}) \leq 0
\end{aligned}
$$

Where:

$$i_{r1}, i_{r2} \in \{1, \ldots, 61\} \times \{1, \ldots, 61\}$$

Given :

$$\{x_{1,2\text{Min}}, x_{1,2\text{Max}}, d_{1,2\text{Min}}, D_{\text{Max}}, \ \ldots$$
$$\ldots \rho_a, \omega, L_v, X_O, Y_O, Z_O, M_O, N_O\}$$

With :

i_{r1}, i_{r2}**:** Index of 61 references in bearing catalog for bearing parameters values [El Samrout, 2019].

L_{10} **:** Lifespan expressed in millions of turns [].

C **:** Dynamic load bearing capacity, intrinsic to rolling bearing [N].

P_{eq} **:** Radial equivalent load [N].

F_a **:** Axial load in [N].

F_r **:** Radial load in [N].

X**,**Y **:** A dimensionless coefficient that depends on the bearing type and on the ratio $\frac{F_a}{C_0}$.

C_0 **:** Static load bearing capacity, intrinsic to the rolling bearing in [N].

$B_{1,2}$ **:** Bearing width (R1) and (R2) [mm].

$D_{1,2}$ **:** Outside diameter of bearings (R1) and (R2) [mm].

$d_{1,2}$ **:** Bearing inner diameter (R1) and (R2) [mm].

$x_{1,2\mathbf{Min}}$ **:** Minimum limit of bearings positions (R1) and (R2) [mm].

$x_{1,2\mathbf{Max}}$ **:** Maximum limit of bearing positions (R1) and (R2) [mm].

$D_{\mathbf{Max}}$ **:** Maximum bearing mounting diameter [mm].

$m_{1,2}$ **:** Mass of bearings (R1) and (R2) [g].

$d_{a1,2}$ **:** Diameter of the necessary shoulders on the shaft for bearings (R1) and (R2) [mm].

ρ_a **:** Shaft density [g/mm^3].

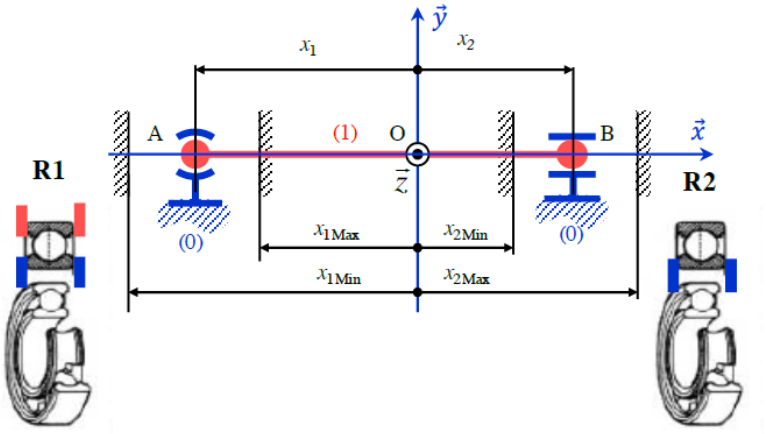


Figure A.2: Illustration of bearing problem

1.3 Coupling Problem

The coupling with bolted rim contains 1 discrete variable, 1 integer variable, 2 continuous variables, 11 inequality constraints and 5 discrete bolt parameters. Figure A.3 illustrates the problem that is formulated as following:

$$\text{minimize} \quad \begin{aligned} f_1(\boldsymbol{x}) &= \frac{\pi}{2} e_p \left[\rho_J \left(16 R_b \phi_3(i_d) - N \phi_4(i_d)^2\right) + \rho_b N \phi_1(i_d)^2\right] \\ f_2(\boldsymbol{x}) &= \frac{\phi_1(i_d)}{d_{\text{Maxi}}} + \frac{N}{N_{\text{Maxi}}} + \frac{M}{M_{\text{Maxi}}} \end{aligned}$$

subject to (A.3)

$$
\begin{aligned}
c_1(\boldsymbol{x}) =& \frac{\alpha_S M}{N R_b K(i_d)} - 1 && \leq 0 \\
c_2(\boldsymbol{x}) =& \frac{\phi_3(i_d) N}{k_s \pi R_b} - 1 && \leq 0 \\
c_3(\boldsymbol{x}) =& (R_{\text{Mini}} + \phi_3(i_d)) - R_b && \leq 0 \\
c_4(\boldsymbol{x}) =& R_b - (R_{\text{Maxi}} - \phi_3(i_d)) && \leq 0
\end{aligned}
$$

where:

$$
\begin{aligned}
&\boldsymbol{x} = \{i_d, N, R_b, M\}^T \\
&i_d \in \{1 \dots 15\} \\
&N \in [N_{\text{Mini}}, N_{\text{Maxi}}] \times \mathbb{N} \\
&R_b \in [R_{\text{Mini}}, R_{\text{Maxi}}] \\
&M \in [M_T, M_{\text{Maxi}}]
\end{aligned}
$$

given:

$$
\begin{aligned}
&K(i_d) = \frac{0.9 R_e f_m \pi (\phi_1(i_d) - 0.93815 \phi_2(i_d))^2}{4\sqrt{1 + 3\left(\frac{4(0.16\phi_2(i_d) + 0.583(\phi_1(i_d) - 0.6495\phi_2(i_d)) f_1)}{\phi_1(i_d) - 0.93815\phi_2(i_d)}\right)}} \\
&k_s = \frac{\sin\left(\frac{\pi}{N_{\text{Mini}}}\right) - \sin\left(\frac{\pi}{N_{\text{Maxi}}}\right)}{\frac{\pi}{N_{\text{Mini}}} - \frac{\pi}{N_{\text{Maxi}}}} \\
&\{N_{\text{Mini}}, N_{\text{Maxi}}, k_s, R_{\text{Mini}}, R_{\text{Maxi}}, e_p, M_T, \ \dots \\
&\dots M_{\text{Maxi}}, \alpha_S, f_1, f_m, R_e, \rho_J, \rho_b\}
\end{aligned}
$$

With:

i_d**:** Index of 15 references in bolts catalog for bolts parameters values [El Samrout, 2019].

C_1 **:** Torsion torque in the bolt body due to tightening [N mm].

F_0 **:** Tension force in the bolt body [N].

d **:** Nominal thread diameter [mm].

p **:** bolt thread pitch [mm].

d_2 **:** Average diameter at the thread end of the thread [mm].

f_1 **:** Coefficient of friction thread [ul].

$F_{0\textbf{Maxi}}$ **:** Maximum tension in the bolt body [N].

$F_{0\mathbf{Mini}}$: Minimum tension in the bolt body [N].

α_S : Uncertainty coefficient of the clamping tool > 1 [ul].

d_s : Diameter of the "resistant" section of the thread [mm].

d_3 : Diameter of the thread kernel [mm].

C_1 : Torsion torque calculated for $F_0 = F_{0\mathrm{Maxi}}$.

R_e : Elastic limit of the bolt depending on its quality class [MPa].

M : Torque transmitted by coupling [N m].

N : Number of bolts [ul].

R_b : Position radius of N bolts [mm].

f_m : Coefficient of friction between the plates of the coupling [ul].

s : Deviation on the circumference between two bolts [mm].

b_m : Radial size of the clamping tool, depending on the nominal diameter of the bolt d [mm].

s_m : Circumference of the clamping tool, depending on b_m and the installation radius of the N bolt[mm].

s_m : Diameter of the holes in the rims, depending on the nominal diameter of the bolt d [mm].

e_p : Thickness of the coupling rims [mm].

ρ_J : Density of the rim material [kg/mm^3].

ρ_b : Density of the bolt material [kg/mm^3].

$d_{\mathbf{Maxi}}$: Diameter of the largest bolts [mm].

$N_{\mathbf{Maxi}}$: Maximum number of bolts [ul].

$M_{\mathbf{Maxi}}$: Maximum torque that can be transmitted by adhesion [N m].

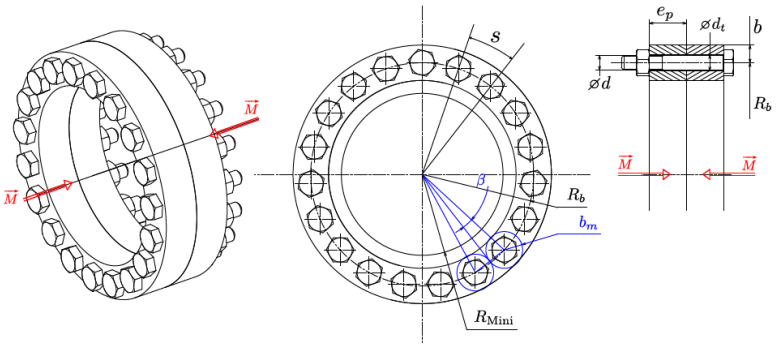


Figure A.3: Illustration of coupling problem

1.4 Disk Brake Problem

In the disk brake design problem, there are five inequality constraints that relate to the surface area, length of the brake, pressure, torque, and temperature. The aim of this problem is to minimize the mass of the brake and the stopping time. Four design variables are considered, namely the inner radius of the discs (x_1), the outer radius of the discs (x_2), the engaging force (x_3), and the number of friction surfaces (y_1), where the latter is an integer variable. This problem is formulated as following:

$$\begin{aligned}
&\text{minimize} && f_1 = 4.9 \times 10^{-5}(x_1^2 - y_1^2)(x_3 - 1) \\
&&& f_2 = \frac{9.82 \times 10^6 (x_1^2 - y_1^2)}{x_2 x_3 (x_1^3 - y_1^3)} \\
&\text{subject to} \\
&&& c_1 = 20 - (x_1 - y_1) && \leq 0 \\
&&& c_2 = 2.5(x_3 + 1) - 30 && \leq 0 \\
&&& c_3 = \frac{x_2}{\pi(x_1^2 - y_1^2)} - 0.4 && \leq 0 \\
&&& c_4 = \frac{2.22 \times 10^{-3} x_2 (x_1^3 - y_1^3)}{(x_1^2 - y_1^2)^2} - 1 && \leq 0 \\
&&& c_5 = 900 - \frac{2.66 \times 10^{-2} x_2 x_3 (x_1^3 - y_1^3)}{(x_1^2 - y_1^2)} && \leq 0 \\
&&& 2 \leq y_1 \leq 20; \quad y_1 \in \mathbb{N} \\
&&& 55 \leq x_1 \leq 80 \\
&&& 75 \leq x_2 \leq 100 \\
&&& 1000 \leq x_3 \leq 3000 \\
&&& x_i \in \mathbb{R} \quad i = 1, 2, 3
\end{aligned} \tag{A.4}$$

1.5 Truss Problem

The nine bar truss problem is unconstrained and contains 3 continuous variables and 9 discrete ones. Figure A.4 illustrates the problem that is formulated as following:

$$\begin{aligned} &\text{minimize} && f_1 = (A_1 + A_2 + A_3 + \sqrt{2}A_4 + A_5 + \sqrt{2}A_6 + A_7 + \sqrt{2}A_8 + A_9) \\ & && f_2 = \frac{4}{A_1} + \frac{1}{A_2} + \frac{1}{A_3} + \frac{8\sqrt{2}}{A_4} + \frac{4}{A_5} + \frac{2\sqrt{2}}{A_6} + \frac{4}{A_7} + \frac{2\sqrt{2}}{A_8} \\ &\text{subject to} \\ & && c_i \leq A_i \leq 10; \quad i = 1, 2, 3 \\ & && A_i \in \{1, 5, 10, 15\}; \quad i = 4, \ldots, 9 \end{aligned} \tag{A.5}$$

where,

$$c_1 = \frac{2}{3}, c_2 = \frac{1}{3}, c_3 = \frac{1}{3}, c_4 = \frac{2\sqrt{2}}{3}, c_5 = \frac{2}{3}, c_6 = \frac{\sqrt{2}}{3}, c_7 = \frac{2}{3}, c_8 = \frac{\sqrt{2}}{3}, c_9 = 0$$

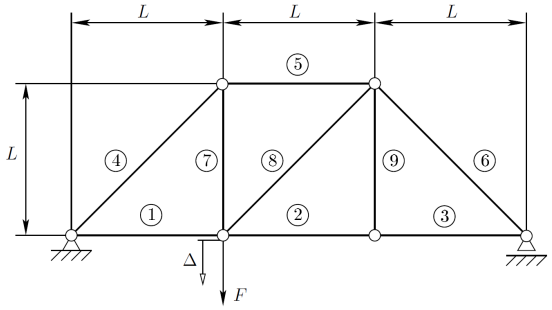


Figure A.4: Illustration of truss problem

1.6 Mela Mathematical Problem

Mela problem is unconstrained with 8 binary variables and 2 continuous variables that are bounded. This problem is formulated as following:

$$\begin{aligned} &\text{minimize} && f_1 = \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}^\top \mathbf{G} \begin{bmatrix} x \\ y \end{bmatrix} + \boldsymbol{c}^{1\top} \begin{bmatrix} x \\ y \end{bmatrix} \\ & && f_2 = \mathbf{c}^{2\top} \begin{bmatrix} x \\ y \end{bmatrix} \\ &\text{subject to} \\ & && -1 \leq x_i \leq 1; \quad i = 1, 2 \\ & && y_j \in \{0, 1\}; \quad j = 1, \ldots, 8 \end{aligned} \tag{A.6}$$

where, $\mathbf{G}, \mathbf{c}^1$ and $\mathbf{c}^2$ are given by:

$$\mathbf{G} = \begin{bmatrix} 1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 4 & 0 & 2 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

$$\boldsymbol{c}^1 = \begin{bmatrix} -1 & -1 & 1 & -10 & 0 & 1 & -2 & 0 & 3 & 0 \end{bmatrix}^\top$$

$$\boldsymbol{c}^2 = \begin{bmatrix} 1 & 2 & -1 & 1 & 5 & -2 & 0 & 6 & 0 & -3 \end{bmatrix}^\top$$

1.7 Tong Mathematical Problem

In Tong problem, it involves 3 continuous variables, 3 binary ones and 9 constraints. The problem is formulated as following:

$$\begin{aligned} \text{minimize} \quad & f_1 = x_1^2 - x_2 + x_3 + 3y_1 + 2y_2 + y_3 \\ & f_2 = 2x_1^2 + x_2 - 3x_3 - 2y_1 + y_2 - 2y_3 \\ \text{subject to} \quad & \\ & c_1 = 3x_1 - x_2 + x_3 + 2y_1 && \leq 0 \\ & c_2 = 4x_1^2 + 2x_1 + x_2 9x_3 + y_1 + 7y_2 && \leq 40 \\ & c_3 = -x_1 - 2x_2 + 3x_3 + 7y_3 && \leq 0 \\ & c_4 = -x_1 + 12y_1 && \leq 10 \\ & c_5 = x_1 - 2y_1 && \leq 5 \\ & c_6 = -x_2 + y_2 && \leq 20 \\ & c_7 = x_2 - y_2 && \leq 40 \\ & c_8 = -x_3 + y_3 && \leq 17 \\ & c_9 = x3 - y_3 && \leq 25 \\ & -100 \leq x_i \leq 100;\ i = 1, 2, 3;\ x_i \in \mathbb{R} \\ & y_j \in \{0, 1\};\ j = 1, 2, 3 \end{aligned} \tag{A.7}$$

Appendix B

3 Stages Reducer Problem

1 Variables

Variable		Signification	Type C/I	Initial Design	Lower Bound	Upper Bound
x_1	$r_{a,0}$	Radius of the input shaft.	C	30.5 mm	10 mm	500 mm
x_2	$l_{a,0}$	Length of the input shaft.	C	130.0 mm	10 mm	1000 mm
x_3	ξ_1	Angular position of 2nd stage.	C	$-17.9°$	$-90°$	$90°$
x_4	im_1	Index of normal module of 1st stage.	I	21	1	41
x_5	$Z_{1,1}$	Number of teeth of 1st stage pinion.	I	15	14	45
x_6	$Z_{1,2}$	Number of teeth of 1st stage wheel.	I	50	20	150
x_7	b_1	Facewidth of 1st stage gear pair.	C	46.0 mm	10 mm	200 mm
x_9	$r_{a,1}$	Radius of the output shaft of 1st stage.	C	39.0 mm	10 mm	500 mm
x_9	$l_{a,1}$	Length of output shaft of 1st stage	C	12.0 mm	10 mm	1000 mm
x_{10}	im_2	Index of normal module of 2nd stage.	I	23	1	41
x_{11}	$Z_{2,1}$	Number of teeth of 2nd stage pinion.	I	15	14	45
x_{12}	$Z_{2,2}$	Number of teeth of 2nd stage wheel.	I	54	20	150
x_{13}	b_2	Facewidth of 2nd stage gear pair.	C	66.0 mm	10 mm	200 mm
x_{14}	$r_{a,2}$	Radius of the output shaft of 2nd stage.	C	60.0 mm	10 mm	500 mm
x_{15}	$l_{a,2}$	Length of output shaft of 2nd stage	C	84.0 mm	10 mm	1000 mm
x_{16}	im_3	Index of normal module of 3rd stage.	I	25	1	41
x_{17}	$Z_{3,1}$	Number of teeth of 3rd stage pinion.	I	18	14	45
x_{18}	$Z_{3,2}$	Number of teeth of 3rd stage wheel.	I	66	20	150
x_{19}	b_3	Facewidth of 2nd stage gear pair.	C	104.0 mm	10 mm	200 mm
x_{20}	$r_{a,3}$	Radius of the output shaft of 3rd stage.	C	83.0 mm	10 mm	500 mm

Table B.1: List of the 20 variables of the optimization problem

2 Reducer Data

Name	Signification	Value
P_t	Power transmitted by the reducer	8.8 kW
N_e	Rotation speed of input shaft	1500 tr/min
$u_{\max}$	Maximal speed reduction ratio per stage	8
$\theta_{\max}$	Allowable torsional angle	0.5°
$\tau_{\max}$	Allowable shear stress	35 MPa
G	Coulomb elasticity modulus for shafts (cast iron)	40 000 MPa
$r_{\min}$	Minimal radius of shafts	10.0 mm
$r_{\max}$	Maximal radius of shafts	500.0 mm
$l_{\min}$	Minimal length of shafts	10.0 mm
$l_{\min}$	Maximal length of shafts	1000.0 mm
$O_0(x, y, z)$	Coordinate of center of input shaft	(100, 144, 300) mm
$O_3(x, y, z)$	Coordinate of center of output shaft	(354, 651, 350) mm
$B_0(x, y, z)$	Coordinate of the bottom corner of casing box	(100, 100, 75) mm
$B_1(x, y, z)$	Coordinate of the top corner of casing box	(355, 930, 660) mm
δ_r	Radial clearance for interference checking	5.0 mm
δ_a	Axial clearance for interference checking	0.0 mm
$Z_{1\min}$	Minimal number of teeth for pinions	14
$Z_{1\max}$	Maximal number of teeth for pinions	45
$Z_{2\min}$	Minimal number of teeth for wheels	20
$Z_{2\max}$	Maximal number of teeth for wheels	150
$b_{\min}$	Minimal face width of teeth	10 mm
$b_{\max}$	Maximal face width of teeth	200 mm
β	Helix angle	0°
x_1, x_2	Profile shift coefficient for pinions and wheels	0.19,−0.19
$\bar{u}$	Global speed reduction ration	44

Table B.2: Input data for the 3SR problem used in this thesis.

3 Gear Data

Name	Signification	Value
S_{Fmin}	Bending stress safety factor	1.5
S_{Hmin}	Contact stress safety factor	1.0
K_{A}	Application factor	1.5
L_{h}	Life duration of the transmission	10 000 h
Q	ISO Quality of gears	6
	ISO data for XC48/XC55 IF, material of pinions and wheels :	
$E_{1,2}$	Young modulus	208 000 MPa
$\sigma_{\mathrm{Flim1,2}}$	Allowable bending stress number	360 MPa
$\sigma_{\mathrm{Hlim1,2}}$	Allowable contact stress number	1170 MPa
$\mathrm{HB}_{1,2}$	Hardness	560 HB
$\nu_{1,2}$	Poisson coefficient	0.272
$\rho_{1,2}$	Bulk density	$7800\,\mathrm{kg/m^3}$
$\mathrm{R}_{\mathrm{Z}(1,2)}$	Mean relative roughness	3.5 µm
$\nu_{40(1,2)}$	Nominal viscosity at 40 °C of lubricant	$32\,\mathrm{mm^2/s}$
$\nu_{50(1,2)}$	Nominal viscosity at 50 °C of lubricant	$21\,\mathrm{mm^2/s}$
	Geometry of the rack tool :	
α_{n}	Normal pressure angle	20°
$\rho_{a(1,2)}$	Root fillet radius coeff. of the basic rack for cylinder gears	0.375
$h_{a(1,2)}$	Addendum coeff. of the basic rack for cylinder gears	1
$h_{f(1,2)}$	Dedendum coeff. of the basic rack for cylinder gears	1.25
s_{pr}	Residual fillet undercut	0 mm

Table B.3: Gears Data.

4 Standard Modules

index im_s	1	2	3	4	5	6	7	8	9	10	11	12
m_n	0.5	0.55	0.6	0.7	0.8	0.9	1	1.125	1.25	1.375	1.5	1.75
index im_s	13	14	15	16	17	18	19	20	21	22	23	24
m_n	2	2.25	2.5	2.75	3	3.5	4	4.5	5	5.5	6	7
index im_s	25	26	27	28	29	30	31	32	33	34	35	36
m_n	8	9	10	11	12	14	16	18	20	22	25	28
index im_s	37	38	39	40	41							
m_n	32	36	40	45	50							

Table B.4: List of the 41 standard values of normal module in [mm], $s = \{1, 2, 3\}$

Appendix C

Numerical Experiment Data

1 BnB-NSGAII Tuning Parameters

Table C.1: Population size, allowed generations and maximum stall generations parameters for BnB-NSGAII.

Problem	**Root**			**Node**			**Leaf**			**ID**
	Pop. Size	*Gen.s*	*Stall*	*Pop. Size*	*Gen.s*	*Stall*	*Pop. Size*	*Gen.s*	*Stall*	
gear	100	800	500	100	50	20	50	20	20	18
	100	800	500	50	100	100	50	20	20	19
	100	800	500	50	100	20	50	20	20	20
	100	800	500	50	20	20	50	20	20	21
	100	800	500	50	50	20	50	20	20	22
	100	800	500	50	800	20	50	20	20	23
	20	100	100	20	100	100	20	100	100	24
	50	20	20	50	100	20	50	20	20	25
	50	20	20	50	20	20	50	20	20	26
Bearing	100	5000	1000	10	200	200	10	800	800	10
	10	200	200	10	200	200	10	5000	2000	11
	10	200	200	10	800	800	10	5000	2000	12
	20	800	800	50	800	800	20	1000	1000	13
	20	800	800	50	800	800	50	3000	1000	14
	50	50	50	20	3000	2000	10	10	10	15
	50	50	50	20	3000	2000	20	3000	2000	16
	50	800	800	20	800	800	50	800	500	17
	50	800	800	50	800	100	50	800	800	18
	50	800	800	50	800	300	50	800	300	19
	50	800	800	50	800	300	50	800	800	20
	50	800	800	50	800	500	50	800	500	21
	50	800	800	50	800	800	100	800	800	22
	50	800	800	50	800	800	50	100	100	23
	50	800	800	50	800	800	50	800	500	24

Table C.1: Population size, allowed generations and maximum stall generations parameters for BnB-NSGAII.

Problem	Root			Node			Leaf			ID
	Pop. Size	*Gen.s*	*Stall*	*Pop. Size*	*Gen.s*	*Stall*	*Pop. Size*	*Gen.s*	*Stall*	
	50	800	800	50	800	800	50	800	800	25
Coupling	100	1000	800	10	200	200	10	200	200	9
	100	1000	800	20	100	100	20	100	100	10
	100	1000	800	20	100	50	100	100	50	11
	100	1000	800	20	100	50	20	100	100	12
	100	1000	800	20	100	50	20	100	50	13
	100	1000	800	20	200	100	20	200	100	14
	100	1000	800	20	200	200	20	200	200	15
	100	1000	800	50	100	50	50	100	50	16
	100	1000	800	50	50	20	50	100	100	17
	100	1000	800	50	50	20	50	100	20	18
	100	1000	800	50	50	50	50	100	100	19
	100	1000	800	50	50	50	50	50	50	20
	10	1000	800	10	100	50	10	400	400	21
	10	1000	800	10	200	200	10	200	200	22
	10	100	100	10	100	100	10	800	200	23
	10	100	100	10	100	100	10	800	800	24
	10	100	100	10	150	100	10	800	200	25
	10	400	400	10	100	100	10	400	200	26
	10	500	500	10	100	100	10	200	100	27
	10	500	500	10	100	100	10	500	500	28
	10	500	500	10	100	50	10	200	100	29
	10	800	800	10	100	100	10	400	400	30
Brake	200	5000	5000	50	20	20	100	20	20	4
	200	5000	5000	50	20	20	100	300	300	5
	200	5000	5000	50	20	20	100	800	500	6
	200	5000	5000	50	20	20	50	200	200	7
	200	5000	5000	50	20	20	50	20	20	8
	200	5000	5000	50	20	20	50	300	300	9
	200	5000	5000	50	20	20	50	800	800	10
	200	5000	5000	50	800	100	50	800	100	11
	50	200	200	50	200	200	100	200	200	12
	50	200	200	50	200	200	50	200	200	13
	50	400	200	50	400	200	100	400	200	14
	50	800	100	50	100	100	100	200	200	15
	50	800	100	50	20	20	100	200	200	16
	50	800	100	50	20	20	100	50	50	17
	50	800	100	50	800	100	50	20	20	18
	50	800	100	50	800	100	50	800	100	19
Truss	1000	1000	1000	100	20	20	100	20	20	7
	20	20	20	20	20	20	200	20	20	8
	20	20	20	20	20	20	20	200	200	9
	20	20	20	20	20	20	20	20	20	10

Table C.1: Population size, allowed generations and maximum stall generations parameters for BnB-NSGAII.

Problem	**Root**			**Node**			**Leaf**			**ID**
	Pop. Size	*Gen.s*	*Stall*	*Pop. Size*	*Gen.s*	*Stall*	*Pop. Size*	*Gen.s*	*Stall*	
	50	1000	100	10	20	20	100	20	20	11
	50	1000	100	10	20	20	100	40	20	12
	50	1000	100	20	20	20	100	20	20	13
	50	1000	100	20	20	20	20	20	20	14
	50	1000	100	50	20	20	100	800	500	15
	50	1000	100	50	20	20	50	20	20	16
	50	20	20	50	20	20	50	20	20	17
Mela	1000	1000	1000	50	20	20	50	200	200	7
	200	800	100	50	20	20	100	200	200	8
	200	800	100	50	20	20	100	800	500	9
	200	800	100	50	20	20	50	200	200	10
	200	800	100	50	20	20	50	500	500	11
	200	800	100	50	50	20	50	200	200	12
Tong	200	1000	1000	100	2000	1200	100	2000	1200	8
	200	1000	1000	200	1000	1000	200	1000	1000	9
	200	1000	1000	50	20	20	100	800	500	10
	200	1000	1000	50	800	500	100	800	500	11
	200	1000	1000	50	800	800	20	10000	10000	12
	200	1000	1000	50	800	800	50	2000	1200	13
	200	1000	1000	50	800	800	50	800	800	14
	200	1200	100	200	1200	100	200	1200	100	15
	200	1200	1200	200	1200	1200	200	1200	1200	16
	20	100	100	50	800	800	20	10000	10000	17
	20	200	200	20	200	200	100	2000	2000	18
	20	200	200	20	200	200	200	1200	1200	19
	20	200	200	20	200	200	200	2000	2000	20
	20	400	400	20	400	400	200	1200	1200	21
	50	1200	1200	50	1200	1200	200	1200	1200	22
	50	400	400	50	400	400	200	1200	1200	23
	50	400	400	50	400	400	300	1200	1200	24
	50	800	800	50	800	800	200	2000	2000	25

2 NSGAII Parameters Tuning

Table C.2: Population size, allowed generations and maximum stall generations parameters for NSGAII.

Problem	ID	Pop. Size	Gen.s	Stall
Gear	1	1000	1000	1000
	2	100	1000	1000
	3	100	1000	800
	4	100	100	500
	5	100	800	100
	6	100	800	300
	7	100	800	400
	8	100	800	500
	9	100	800	600
	10	100	800	800
	11	100	800	900
	12	200	800	800
	13	500	1000	1000
	14	500	800	800
	15	50	800	100
	16	50	800	300
	17	50	800	800
Bearing	1	1000	1000	1000
	2	1000	800	800
	3	100	1000	1000
	4	100	5000	1000
	5	100	5000	2000
	6	100	5000	800
	7	100	800	800
	8	50	800	100
	9	50	800	800
Coupling	1	1000	1000	1000
	2	100	1000	500
	3	100	1000	800
	4	100	800	100
	5	100	800	500
	6	100	800	800
	7	50	800	100
	8	50	800	300
Brake	1	100	800	800
	2	200	5000	5000
	3	50	800	100
Truss	1	1000	1000	1000
	2	200	2000	100
	3	50	1000	100
	4	50	2000	100
	5	50	500	100

Table C.2: Population size, allowed generations and maximum stall generations parameters for NSGAII.

Problem	ID	Pop. Size	Gen.s	Stall
	6	50	800	100
Mela	1	1000	1000	1000
	2	100	800	100
	3	200	1800	100
	4	200	500	100
	5	200	800	100
	6	50	1800	100
Tong	1	200	1000	1000
	2	200	1200	100
	3	200	1200	1200
	4	200	2000	2000
	5	200	5000	2000
	6	200	5000	5000
	7	50	1200	100

Ahmed JABER

Doctorat : Matériaux, Mécanique, Optique, Nanotechnologie

Année 2021

Algorithme hybride pour l'optimisation multi-objectif non convexe en variables mixtes en conception mécanique

Les problèmes d'optimisation sous contraintes non linéaires non convexes multi-objectifs en variables mixtes (discrètes et continues) apparaissent dans de nombreux domaines de l'ingénierie et notamment dans les applications de conception en mécaniques. Cette thèse vise à développer une nouvelle méthode pour résoudre ces problèmes d'optimisation. Notre proposition est une hybridation de l'algorithme multicritère « Branch-and-Bound » (MCBB) avec l'algorithme évolutionnaire de type NSGAII. L'approche proposée est en outre renforcée par de nouvelles stratégies de branchement conçues pour l'algorithme MCBB. Les contraintes du problème d'optimisation sont gérées à l'aide d'une nouvelle technique dédiée aux algorithmes évolutionnaires. Les performances de cette nouvelle approche sont évaluées et comparées à l'existant par une étude statistique sur un ensemble de problèmes tests. Les résultats montrent que les performances de notre algorithme sont compétitives face à l'algorithme NSGAII seul. Nous proposons deux applications de notre algorithme : les applications "Recherche de solutions faisables" et "Recherche de solutions optimales". Celles-ci sont appliquées sur un problème industriel réel d'un réducteur à engrenages à 3 étages formulé comme un problème bi-objectif. Dans ce problème des contraintes sont incluses pour satisfaire aux exigences de normes ISO sur le calcul de la capacité de charge des engrenages.

Mots clés : optimisation mathématique – décision multicritère – mécanique – métaheuristiques – programmation évolutionnaire – conception technique.

Hybrid Algorithm for Multi-objective Mixed-integer Non-convex Mechanical Design Optimization Problems

Multi-objective mixed-integer non-convex non-linear constrained optimization problems that appears in several fields especially in mechanical applications. This thesis aims to develop a new method to solve such problems. Our proposal is a hybridization of the Multi-Criteria Branch-and-Bound (MCBB) algorithm with the Non-dominated Sorting Genetic Algorithm 2 (NSGAII). The proposed approach is furthermore enhanced by new branching strategies designed for MCBB. The constraints are handled using a new proposed constraint handling technique for evolutionary algorithms. Numerical experiments based on statistical assessment are done in this thesis to examine the performance of the new proposed approach. Results show the competitive performance of our algorithm among NSGAII. We propose two applications of our proposed approach: "Search Feasibility" and "Seek Optimality" applications. Both are applied on a real-world state of art 3 stages reducer problem which is formulated in this thesis to a bi-objective problem to meet the requirement of ISO standards on calculation of load capacity of gears.

Keywords: mathematical optimization – multiple criteria decision making – mechanics – metaheuristics – evolutionary programming (computer science) – engineering design.

Thèse réalisée en partenariat entre :

Université Libanaise — l'Europe s'engage en Champagne-Ardenne avec le FEDER — Union Européenne, Fonds Européen de Développement Régional — utt Université de Technologie Troyes

Ecole Doctorale "Sciences pour l'Ingénieur"