AN ADVANCED EVOLUTIONARY PROGRAMMING METHOD FOR MECHANICAL SYSTEM DESIGN: FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET SİNAN HASANOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING

JANUARY 2019

Approval of the thesis:

AN ADVANCED EVOLUTIONARY PROGRAMMING METHOD FOR MECHANICAL SYSTEM DESIGN: FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION

submitted by **MEHMET SİNAN HASANOĞLU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. M. A. Sahir Arıkan
Head of Department, **Mechanical Engineering** ________________

Assoc. Prof. Dr. Melik Dölen
Supervisor, **Mechanical Engineering Dept., METU** ________________

Examining Committee Members:

Assoc. Prof. Dr. E. İlhan Konukseven
Mechanical Engineering Dept., METU ________________

Assoc. Prof. Dr. Melik Dölen
Mechanical Engineering Dept., METU ________________

Prof. Dr. Göktürk Üçoluk
Computer Engineering Dept., METU ________________

Assoc. Prof. Dr. Can Ulaş Doğruer
Mechanical Engineering Dept., Hacettepe University ________________

Assist. Prof. Dr. Kutluk Bilge Arıkan
Mechanical Engineering Dept., TED University ________________

Date: 16/01/2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Mehmet Sinan Hasanoğlu

Signature :

ABSTRACT

AN ADVANCED EVOLUTIONARY PROGRAMMING METHOD FOR MECHANICAL SYSTEM DESIGN: FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION

Hasanoğlu, Mehmet Sinan
Ph.D., Department of Mechanical Engineering
Supervisor: Assoc. Prof. Dr. Melik Dölen

January 2019, 205 pages

Constrained optimization problems constitute an important fraction of optimization problems in mechanical engineering domain. It is not rare for these problems to be highly-constrained where a specialized approach that aims to improve constraint satisfaction level of the whole population as well as finding the optimum is deemed useful especially when the objective functions are very costly.

This dissertation introduces a new algorithm titled Feasibility Enhanced Particle Swarm Optimization (FEPSO) to handle highly-constrained optimization problems. FEPSO, which is based on particle swarm optimization technique, treats feasible and infeasible particles differently. Infeasible particles do not need to evaluate objective functions and fly only based on social attraction depending on a single violated constraint, called the activated constraint (AC), which is selected in each iteration based on constraint priorities and flight occurs only along dimensions of the search space to which the AC is sensitive. To ensure progressive improvement of constraint satisfaction, particles are not allowed to violate a satisfied constraint in FEPSO. Unlike its counterparts, FEPSO does not require any feasible solutions in the initialized swarm.

A modified version of the new method called the multi-objective FEPSO (MOFEPSO) is also introduced. MOFEPSO, which is capable of handling highly-constrained multi-objective optimization problems, employs repositories of non-dominated and feasible positions (or solutions) to guide feasible particle flight.

In this study, several constrained optimization problems are described. For the given problems, the performance of FEPSO- and MOFEPSO are comparatively evaluated against a number of popular optimization algorithms found in the literature. The results suggest that FEPSO- and MOFEPSO are effective and consistent in obtaining feasible points, finding good solutions, and improving the constraint satisfaction level of the swarm as a whole.

Keywords: mechanical design, constrained problems, multi-objective optimization, particle swarm optimization, evolutionary algorithms

ÖZ

MEKANİK SİSTEM TASARIMI İÇİN GELİŞMİŞ EVRİMSEL PROGRAMLAMA YÖNTEMİ: OLURLUĞU ARTTIRILMIŞ PARÇACIK SÜRÜ OPTİMİZASYONU

Hasanoğlu, Mehmet Sinan
Doktora, Makina Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Melik Dölen

Ocak 2019 , 205 sayfa

Kısıtlı eniyileme problemleri makine mühendisliği alanındaki eniyileme problemlerinin önemli bir kısmını oluşturur. Bu problemlerin aşırı kısıtlı olması da az rastlanır bir durum değildir. Amaç fonksiyonunun zor hesaplandığı aşırı kısıtlı problemlerde amaç fonksiyonunu eniyilemenin yanında popülasyonun kısıtları sağlama durumunu bütüncül olarak iyileştirecek bir yaklaşımın önemli olduğu değerlendirilmektedir.

Bu çalışmada aşırı kısıtlı problemler için olurluğu geliştirilmiş parçacık sürüsü eniyilemesi (FEPSO) isimli yeni bir yöntem sunulmaktadır. Parçacık sürüsü optimizasyonu tekniğine dayanan FEPSO, kısıtlara uyan ve uymayan parçacıkları farklı bir şekilde ele alır. FEPSO'da kısıtlara uymayan parçacıklar sadece etkin kısıt adı verilen tek bir kısıta dayalı sosyal çekim kuralları ile hareket ederler. Etkin kısıt her yinelemede kısıt önceliklerine göre seçilir ve parçacık hareketi sadece etkin kısıtın duyarlı olduğu karar değişkeni boyutlarında gerçekleşir. Olurluk seviyesinin sürekli iyileşmesi amacıyla FEPSO, sağlanmış bir kısıtın tekrar ihlal edilmesine izin vermez. Benzerlerinden farklı olarak FEPSO, ilklendirilmiş sürüde olurlu parçacıkların bulunmasını gerektirmez.

Ayrıca bu yeni yaklaşımın çok amaçlı FEPSO (MOFEPSO) adı verilen bir tipi de tanımlanmaktadır. Aşırı kısıtlı çok amaçlı problemlerin çözümünde kullanılabilen MOFEPSO, olurlu parçacığa kılavuz olarak basılgın olmayan ve olurlu çözümlerden oluşan veri havuzlarından faydalanır.

Bu çalışmada bazı kısıtlı eniyileme problemleri de tarif edilmiştir. Bu problemler kullanılarak FEPSO ve MOFEPSO, literatürde bulunan bazı yaygın eniyileme algoritmaları ile karşılaştırılmıştır. Elde edilen sonuçlar değerlendirildiğinde FEPSO ve MOFEPSO'nun etkin ve istikrarlı olarak olurlu noktalar bulabildiği, iyi çözümler elde edebildiği ve sürünün kısıtları sağlama durumunu bütüncül olarak geliştirebildiği görülmektedir.

Anahtar Kelimeler: mekanik tasarım, kısıtlı problemler, çok amaçlı eniyileme, parçacık sürü optimizasyonu, evrimsel algoritmalar

This work is dedicated to my wife, the spouse extraordinaire,
and our wonderful children.

ACKNOWLEDGMENTS

First and foremost I want to thank my advisor Associate Professor Melik Dölen. Without his vision, guidance, and encouragement I wouldn't have come so far. His immense knowledge helped me in the time of research and writing of this dissertation. I can not imagine having a better mentor.

Appreciation is extended to members of my thesis monitoring committee, Professor Erhan İlhan Konukseven and Assistant Professor Can Ulaş Doğruer. Their wealth of ideas made a huge impact to this research.

I express my gratitude to all those who gave me the possibility to complete this thesis. I would like to use an opportunity to thank my previous advisor Professor Serkan Dağ. I am especially grateful to Professor Zafer Dursunkaya for his critical inputs. I also want to thank Professor İhsan Solak for his help in mathematical aspects.

I offer my sincerest gratitude to my manager Mr. Fatih Yaşar and my former manager Mr. Osman Başoğlu who have supported me throughout my research with their patience. I am also grateful to my wonderful and understanding group of colleagues. My special thanks are to Dr. Kenan Bozdaş, Kenan Bozkaya, Fahrettin Koç, and Rüştü Bilgin for giving me a helping hand whenever I needed.

TÜBİTAK SAGE is also greatly acknowledged for providing computational resources and making this work possible.

I cannot end without thanking my parents Adnan and Canan for their determination to give me a good education. My sister Elif and my brother Hasan were also always there for me. A final word of recognition goes to my wife İmran, my son Ahmet, and my daughter Ayşe, for being my happiness and source of confidence with their unconditional love, trust, and support.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xvii

LIST OF FIGURES xx

NOMENCLATURE xxv

CHAPTERS

1 INTRODUCTION 1

  1.1 Outline of the dissertation 4

2 REVIEW OF THE STATE OF THE ART 7

  2.1 Definition and types of optimization problems 7

    2.1.1 Multi-objective optimization problems 8

      2.1.1.1 Pareto optimality 8

      2.1.1.2 ε-dominance 9

    2.1.2 Constrained optimization problems 10

    2.1.3 Multimodal optimization problems 11

    2.1.4 Combinatorial optimization 12

2.2 Classical optimization techniques 12

2.2.1 Lagrange multipliers 13

2.2.2 Convex programming problem 13

2.2.3 Linear programming 14

2.2.4 Nonlinear programming 14

2.2.4.1 Nelder-Mead simplex method 15

2.2.4.2 Steepest descent method 16

2.2.4.3 Newton's method 17

2.3 Evolutionary algorithms 17

2.3.1 Global diversity evolutionary multi-objective optimizer - GDEMO 19

2.3.2 ε based multiobjective evolutionary algorithm - ε-MOEA . . . 19

2.3.3 Simulated annealing - SA 21

2.3.4 Genetic algorithms - GA 21

2.3.4.1 Binary genetic algorithm 22

2.3.4.2 Real-coded genetic algorithm 25

2.3.5 Non-dominated sorting genetic algorithm - NSGA2 25

2.3.6 Particle swarm optimization - PSO 26

2.3.6.1 A fundamental multi-objective PSO approach 29

2.3.6.2 Constraint handling in PSO 30

2.3.6.3 PSO for discrete- or mixed variable optimization problems 31

2.4 Design problems in the mechanical engineering domain 32

2.5 Research opportunity 34

2.6 Closure 36

3 FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION 37

3.1 General description of FEPSO 38

3.2 Initialization 40

3.3 Velocity update rules for infeasible particles 44

3.3.1 Selection of the AC and the global guide 44

3.3.1.1 The velocity update equation for infeasible particles . . 45

3.4 Velocity update rules for feasible particles 46

3.5 Particle flight 46

3.5.1 Enforcing decision variable limits 47

3.5.2 Virtual boundary search 49

3.6 Post-flight operations 51

3.7 Handling discrete decision variables 51

3.8 Preliminary tests on benchmark problems 52

3.8.1 Pressure vessel design problem 52

3.8.2 Tension/compression spring design problem 54

3.8.3 Welded beam design problem 54

3.8.4 Discussion of benchmark results 58

3.9 Closure 58

4 MULTI-OBJECTIVE FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION 63

4.1 General description of MOFEPSO 63

4.2 Sets of global- and personal non-dominated solutions: *gbest* and *pbest* 64

4.3 Initialization 66

4.4 Velocity update rules for feasible particles 68

4.5 Post-flight operations 69

4.6 Closure 69

5 FOUR-STAGE GEAR TRAIN PROBLEM 71

5.1 Single-objective four-stage gear train problem 71

5.1.1 Original four-stage gear train problem 72

5.1.2 Reduced constraints four-stage gear train problem 78

5.1.3 Mixed-valued four-stage gear train problem 80

5.1.4 Results and discussion 80

5.2 Multi-objective four-stage gear train problem 89

5.2.1 Problem definition for MOG, RC-MOG, and MV-MOG 89

5.2.2 Results and discussion 91

5.2.2.1 Statistical evaluation of simulation results 95

5.2.2.2 Exploration capability 95

5.3 Closure 101

6 HEAT PIPE DESIGN PROBLEM 105

6.1 Single-objective heat pipe design problem 107

6.1.1 Problem definition 107

6.1.2 Results and discussion 113

6.2 Multi-objective heat pipe design problem 117

6.2.1 Problem definition 119

6.2.2 Results and discussion 120

6.3 Closure 128

7 ALGORITHM CHARACTERISTICS 129

7.1 Qualitative and empirical analysis of complexity 129

7.2 Comparison of multi-objective and single-objective approaches . . . 133

7.2.1 Comparison method 134

7.2.2 Results and discussion 136

7.3 Particle flight characteristics 140

7.4 Closure 150

8 CONCLUSION 153

8.1 Future work 155

APPENDICES

A MATERIAL PROPERTIES USED FOR HEAT PIPE DESIGN PROBLEM . 179

A.1 Container and wick material properties 179

A.2 Working fluid properties 179

A.2.1 Methanol 180

A.2.2 Ethanol 180

A.2.3 Ammonia 181

B USAGE NOTES FOR MATLAB IMPLEMENTATION 183

B.1 The `mofepso` function 183

B.1.1 Syntax 183

B.1.2 Input argument 183

B.1.2.1 Constraint- and objective function definitions 188

B.1.3 Output argument 190

B.1.3.1 The particle structure 191

B.1.3.2 The position structure 192

B.1.4 Standard output 193

B.2 The `mofepsooptions` function 194

B.2.1 Syntax 195

B.2.2 Output argument 195

B.3 Example 195

CURRICULUM VITAE 203

LIST OF TABLES

TABLES

Table 2.1 Some popular mechanical design problems 33

Table 3.1 Summary of results for the pressure vessel design problem 53

Table 3.2 Summary of results for the tension/compression spring design problem 55

Table 3.3 Summary of results for the welded beam design problem 57

Table 5.1 Decision variables of GTP 74

Table 5.2 Constraint functions for GTP 75

Table 5.3 Constants for GTP 76

Table 5.4 Constraint functions for RC-GTP 79

Table 5.5 Decision variables of MV-GTP 80

Table 5.6 FEPSO algorithm parameters used in simulations 81

Table 5.7 FEPSO simulation results summary of all three four-stage gear train problems 82

Table 5.8 Best solutions found during FEPSO simulations for all three problems 83

Table 5.9 Comparison of results for GTP 85

Table 5.10 Comparison of results for RC-GTP 86

Table 5.11 Comparison of results for MV-GTP 86

Table 5.12 Algorithm parameters used for MOGA2 in benchmark simulations . 86

Table 5.13 Algorithm parameters used for NSGA2 in benchmark simulations . 87

Table 5.14 Similarities between decision variables- and constraint functions of single-objective and multi-objective variants of the GTP 90

Table 5.15 Algorithm parameters used for MOPSO 92

Table 5.16 Comparison of results obtained for MOG 93

Table 5.17 Comparison of results obtained for RC-MOG 94

Table 5.18 Comparison of results obtained for MV-MOG 94

Table 5.19 Some selected Pareto solutions 96

Table 5.20 Decision vectors and objective values for some selected Pareto solutions found by MOFEPSO 97

Table 5.21 Statistical results (p) for nonparametric comparisons of MOGA2, MOPSO, and NSGA2 Pareto solutions with MOFEPSO Pareto solutions using Wilcoxon method 99

Table 6.1 Decision variables 108

Table 6.2 Selected operating conditions 114

Table 6.3 All results obtained in simulations for the heat pipe problem 115

Table 6.4 A selected design obtained by FEPSO 117

Table 6.5 Statistical summary of results obtained in simulations for the multi-objective heat pipe problem 123

Table 7.1 Complexity comparison of MOFEPSO with generic multi-objective PSO with a constraint handling mechanism 130

Table 8.1 All problems and algorithms used in the dissertation 154

Table B.1 Fields of the options structure 184

Table B.2 Fields of the output structure 191

Table B.3 Fields of the particle structure 192

Table B.4 Fields of the position structure 192

LIST OF FIGURES

FIGURES

Figure 1.1 Disc spring drawing 3

Figure 2.1 Illustration of Pareto front in two dimensional objective space. . 10

Figure 2.2 Illustration of reflection in Nelder-Mead simplex algorithm in two dimensions ($N = 2$) 16

Figure 2.3 A generalized evolutionary algorithm flowchart for constrained multi-objective optimization problems 20

Figure 2.4 Basic principle of a genetic algorithm 23

Figure 2.5 Illustration of a crossover operation 24

Figure 2.6 Canonical PSO flowchart 27

Figure 3.1 Flowchart of the main FEPSO algorithm 39

Figure 3.2 Flowchart of the FEPSO initialization routine 41

Figure 3.3 Flowchart of the FEPSO particle flight routine 48

Figure 3.4 Illustration of limit violation rate in two dimensional decision space 49

Figure 3.5 Flowchart of the FEPSO virtual boundary search routine 60

Figure 3.6 Illustration of pressure vessel design problem 61

Figure 3.7 Illustration of tension/compression spring design problem 61

Figure 3.8 Illustration of welded beam design problem 61

Figure 4.1 Flowchart of main MOFEPSO algorithm 65

Figure 4.2 Flowchart of MOFEPSO initialization routine 67

Figure 5.1 Representative pitch circles associated with discrete shaft locations of the pinion and gear at stage s 73

Figure 5.2 Best GTP, RC-GTP, and MV-GTP solutions found by FEPSO . . 84

Figure 5.3 Mean number of constraint violations in the population during the RC-GTP runs. 88

Figure 5.4 Illustrations of some selected Pareto solutions found by MOFEPSO and their comparison to single-objective solutions found by FEPSO . . 98

Figure 5.5 Progression of Pareto solutions along iterations of selected MOFEPSO runs 99

Figure 5.6 Graphical comparison of Pareto solutions obtained by different algorithms for MOG, RC-MOG, and MV-MOG 100

Figure 5.7 Rates of congruence at first two stages for MOG Pareto solutions found by MOFEPSO 102

Figure 6.1 Heat pipe 107

Figure 6.2 Thermal resistance model 109

Figure 6.3 Graphical comparison of results obtained by NSGA2 and FEPSO for the heat pipe problem using different maximum iterations ($maxIter$) and population sizes ($popSize$) 116

Figure 6.4 Graphical comparison of results obtained by NSGA2 and FEPSO which have similar number of CFEs 118

Figure 6.5 Maximization of the heat pipe operating condition range 120

Figure 6.6 Flowchart of the numerical approach to evaluate Q_{max} of a heat pipe design 121

Figure 6.7 Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 100, number of iterations: 50) 124

Figure 6.8 Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 100, number of iterations: 100) 125

Figure 6.9 Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 200, number of iterations: 50) 126

Figure 6.10 Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 200, number of iterations: 100) 127

Figure 7.1 Mean number of CFEs versus number of decision variables (N) and swarm size (I) 132

Figure 7.2 Constraint- and objective functions of the SRN problem 135

Figure 7.3 Comparison of multi-objective results obtained for RC-MOG with results obtained by single-objective approaches using y^1, y^2, and y^3 criteria 137

Figure 7.4 Comparison of multi-objective results obtained for MV-MOG with results obtained by single-objective approaches using y^1, y^2, and y^3 criteria 138

Figure 7.5 All Pareto solutions obtained for the SRN problem in a single selected multi-objective optimization run (Pareto front - 1879 points) and solutions that are the best (have the lowest value) in terms of y^1, y^2, and y^3 criteria 139

Figure 7.6 Number of solutions obtained in the Pareto set with varying number of iterations- and swarm size 140

Figure 7.7 Constraint functions and the feasible regions of the hypercube problem in two dimensions 142

Figure 7.8 Progression of particles in the 2D design space (iterations 1 to 6) 143

Figure 7.9 Progression of particles in the 2D feasible region (iterations 2 to 7) 144

Figure 7.10 Progression of particles in the 2D feasible region (iterations 8 to 13) 145

Figure 7.11 Progression of particles in the 3d design space (iterations 1 to 6) 146

Figure 7.12 Progression of particles in the 3D feasible region (iterations 5 to 10) 147

Figure 7.13 Progression of particles in the 3D feasible region (iterations 11 to 16) 148

Figure 7.14 Comparison of boolean or real valued constraint functions for the RC-MOG problem 150

Figure 7.15 Variation of number of constraint violations as iterations progress in hypercube problem of different dimensions ($N = 20, 25, 30$) 151

Figure B.1 Directory structure of the MATLAB source code files for the example problem 196

NOMENCLATURE

Abbreviations

ε-MOEA	ε based multiobjective evolutionary algorithm
AC	Activated constraint
BFGS	Broyden-Fletcher-Goldfarb-Shanno algorithm
CFD	Computational fluid dynamics
CFE	Constraint function evaluation
CMOP	Constrained multi-objective optimization problem
CMOP	Constrained multi-objective optimization problem
DFP	Davidon-Fletcher-Powell algorithm
DOPSO	Dynamic-objective particle swarm optimization
EA	Evolutionary algorithm
FEM	Finite element method
FEPSO	Feasibility enhanced particle swarm optimization
GA	Genetic algorithms
GDEMO	Global diversity evolutionary multi-objective optimizer
GSEMO	Global simple evolutionary multi-objective optimizer
GTP	The original four-stage gear train problem
HPSO	Hybrid particle swarm optimization
MI-LXPM	Mixed integer Laplace crossover and power mutation

MNCV	Mean number of constraint violations
MOEA	Multi-objective evolutionary algorithm
MOFEPSO	Multi-objective feasibility enhanced particle swarm optimization
MOGA	Multi-objective genetic algorithm
MOP	Multi-objective problem
MV-GTP	Mixed-valued four-stage gear train problem
NPGA	Niched Pareto genetic algorithm
NSGA	Non-dominated sorting genetic algorithm
OCCURM	Occurrence matrix calculator
OFE	Objective function evaluation
PSO	Particle swarm optimization
RC-GTP	Reduced constraints four-stage gear train problem
SA	Simulated annealing
SQP	Sequential quadratic programming
VBS	Virtual boundary search

Symbols

$\mathbb{B}$	Boolean set
$\mathbb{N}$	Set of natural numbers
$\mathbb{N}_{>a}^{\leq b}$	Set of natural numbers greater than a and less than or equal to b
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{>a}^{\leq b}$	Set of real numbers greater than a and less than or equal to b
$\mathbf{B}$	Global set of non-dominated solutions (*gbest*)
$\mathbf{C}$	Matrix of current constraint function values of patticles in the swarm

$\mathbf{D}_i$	Personal set of non-dominated solutions of the ith particle (*pbest*)
$\mathbf{f}$	Vector objective function
$\mathbf{p}$	Constraint priority vector
$\mathbf{S}$	Constraint sensitivity matrix
$\mathbf{T}$	AC selection count matrix
$\mathbf{V}$	Matrix of current particle velocities in PSO
$\mathbf{X}$	Matrix of current particle positions in PSO
$\mathbf{x}$	Decision vector
$\mathbf{y}$	Objective vector
a_v	Velocity initialization factor
C_1	Social acceleration coefficien
C_2	Cognitive acceleration coefficient
f	Objective function
f_k	kth objective function
g_m	mth constraint function
I	Number of particles in the swarm
K	Number of objectives
M	Number of constraints
m	Index of constraint
N	Number of decision variables
n	Index of decision variable
q	Infeasibility rate
W	Inertia factor

w	Decision variable limits violation rate
x_n^L	Lower decision variable limit
x_n^U	Upper decision variable limit
y	Objective variable

CHAPTER 1

INTRODUCTION

With increasing complexity of engineering design problems, optimization has become a necessity instead of an option in the workflow of the designer. The domain of mechanical engineering is not an exception in this regard. It encompasses a wide variety of design problems of differing complexity- and characteristics. While some of these problems can easily be handled with widely used general purpose optimization algorithms, some require special techniques.

In fact, mechanical design itself is an optimization process that deals with many potentially conflicting objectives. Strength, fatigue, and weight are only several of the parameters commonly treated as design goals. Any sufficiently complicated mechanical system has numerous parts that need to be designed separately with different objectives. On the other hand, some features of these parts need to be specified through system level design work. By and large, these processes require application of optimization techniques for the final product to be as functional and as effective as possible. From this point of view, optimization is a decision making tool which allows the designer to reach this ultimate goal. Moreover, there is an increasing demand for optimal designs in every field of engineering due to the influence of more challenging economic and environmental constraints [1]. Advances in computational techniques and increasing computational power has led the transformation of computer aided design tools from mere analysis tools into design optimization tools.

Although a plethora of analytical- and numerical methods that calculate extreme values of a function have long been utilized in engineering design and analysis, classical methods are rarely useful in design of mechanical systems which involve many design variables as well as complex nonlinear outcomes. These classical methods generally yield local extrema whereas the designer is interested in the global optimum.

Unlike classical approaches, modern heuristic optimization techniques are generally stochastic in nature and much more versatile in terms of complexity and diversity of problems they can be applied to.

Vast majority of mechanical engineering design problems are constrained. In constrained problems, the values decision vectors can take are limited. Furthermore, there is a great chance that the optimum values of objectives lie on the constraint boundary in real-world engineering design optimization problems. This comes as no surprise since the best design would require usage of all available factors (resources, energy etc.) to their limits. Therefore, constraints carry utmost importance in all kinds of design optimization.

Some problems are highly-constrained, wherein the process of finding a feasible solution that satisfies all constraints becomes a major challenge of a similar scale for finding the optimum. Moreover, not all problems consist of only real valued continuous decision variables. There are many design variables that take integer- or predefined discrete values due to physical- or practical reasons.

Design of a disc spring shown in Figure 1.1 can be given as an example to constrained mechanical design optimization [2, 3]. The disc spring optimization problem involves minimization of the mass (or volume) by determining appropriate values for the design parameters while satisfying several constraints related with compressive stress, deflection, and geometric consistency. Design variables of the problem are the internal diameter (D_i), the external diameter(D_e), the thickness (t), and the height (h) of the spring. The problem can be defined as to minimize the volume approximated as

$$0.07076\pi(D_e^2 - D_i^2)t \tag{1.1}$$

subject to the following constraints:

$$\frac{4E\delta_{max}}{(1-\mu^2)\alpha D_e^2}\left[\beta\left(h - \frac{\delta_{max}}{2}\right) + \gamma t\right] \leq S \tag{1.2}$$

$$\frac{4E\delta_{max}}{(1-\mu^2)\alpha D_e^2}\left[\left(h - \frac{\delta_{max}}{2}\right)(h-\delta_{max})t + t^3\right] \geq 24\,\mathrm{kN} \tag{1.3}$$

$$h + t \leq 5\,\mathrm{cm} \tag{1.4}$$

$$D_e \geq D_i \tag{1.5}$$

where $S = 1380\,\mathrm{N/mm^2}$ is the allowable stress, $E = 207\,\mathrm{kN/mm^2}$ is the modulus of elasticity, and $\delta_{max} = 5\,\mathrm{mm}$ is the maximum deflection. Furthermore, α, β, and γ are some coefficients that are functions of D_e/D_i. The aim in this problem is to find the combination of design variables that make the volume minimum.

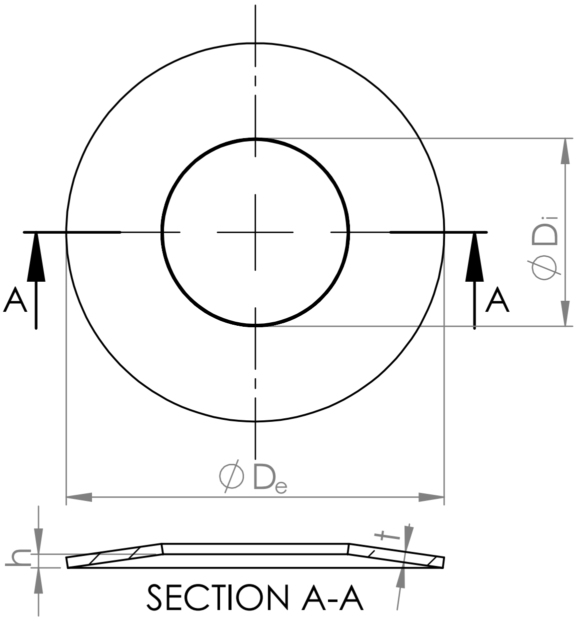


Figure 1.1: Disc spring drawing

The example given above is a simple problem with only a single objective. However, many engineering optimization problems involve several objective functions which are occasionally conflicting in nature. Such problems are called multi-objective problems and oftentimes do tend to also have a multitude of constraints. For example, in a beam design optimization, the designer might want to minimize the weight of the beam and maximize the load capacity. These two objectives would have conflicting impacts on the majority of design variables. Although a multi-objective problem can be converted to a single-objective one, treating it with a specially designed multi-objective method has certain advantages.

Another aspect of some design problems worth mentioning is the complexity of constraint- and objective functions that constitute them. In many cases, objective functions have a higher time complexity when compared with the constraint functions. Therefore, specialized techniques for handling easily-evaluated constraints or constraints that prohibit other evaluations in constrained problems (especially in highly-constrained ones) may alleviate the burden of the optimization algorithm in finding feasible points and the optimum.

A thorough investigation of the current literature on constraint handling methods for evolutionary algorithms indicates that there is a lack of an evolutionary programming framework (especially on swarm-intelligence) that is specialized for use in highly-constrained single- and multi-objective problems aiming to improve constraint satisfaction level of whole population. Such an algorithm would not only allow finding feasible design points in highly-constrained problems, but also constitute a preprocessing methodology for initializing a population that has a better level of constraint satisfaction for other optimization algorithms that could benefit from a population with less constraint violations. This would also provide an alternative workflow for cases where objective functions are too costly to evaluate in lieu of some constraint violations.

The motivation of this study is to develop a heuristic optimization technique that is capable of handling highly-constrained single-objective- and multi-objective mechanical design problems.

1.1 Outline of the dissertation

The dissertation is divided into 8 chapters including this introduction.

Chapter 2 includes a review of the basic theories and background of this study. Some popular concepts and approaches for both classical- and modern optimization techniques are reviewed. Additionally, some references on popular mechanical design problems in the literature are discussed before presentation of a synthesis for research opportunity.

Chapter 3 introduces a new method called the *feasibility enhanced particle swarm optimization* (FEPSO) which constitutes the core of the study. After a detailed description of the FEPSO algorithm, results of some preliminary tests performed on several widely used benchmark problems are given.

Chapter 4 introduces a modification of FEPSO for multi-objective optimization problems: multi-objective FEPSO (MOFEPSO). This chapter focuses only on enhancements made on top of FEPSO to allow handling multiple objectives.

Chapter 5 discusses the application of FEPSO to a highly-constrained problem (four-stage gear train problem) and introduces two new variants of the problem. Both single-objective- and multi-objective versions of the problem are discussed separately in two main sections of this chapter. Results obtained by FEPSO, MOFEPSO and other competing algorithms as well as data found in the literature are used to comparatively assess the performance of both FEPSO and MOFEPSO. The exploratory capability of the algorithm is also discussed with the aid of data obtained from solution of multi-objective version of the four-stage gear train problem.

Chapter 6 presents an application of FEPSO to design of a mesh wick type heat pipe. Once again, a comparative analysis is performed to investigate the algorithm behavior. Aim of this chapter is to show FEPSO's competence in a relatively less constrained problem and examine how certain parameters such as the swarm size effect results. In the second part of this chapter, a novel multi-objective version of the heat pipe design problem is introduced. Solutions obtained for this problem by MOFEPSO are presented and discussed.

Chapter 7 consists of three sections that describe different aspects of the algorithm. First section presents a qualitative and empirical analysis related with the time complexity of MOFEPSO. While comparing MOFEPSO and FEPSO in terms of practicality, performance, and efficiency; the second section of this chapter includes a general discussion- and comparison of single-objective- and multi-objective approaches. The third and final section in this chapter scrutinizes the unique characteristics of particle motion in FEPSO.

General conclusions are drawn in Chapter 8 and some direction for future studies are discussed.

Some concepts and results included in this dissertation have been published in *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* and *Engineering Optimization* with articles titled "Feasibility enhanced particle swarm optimization for constrained mechanical design problems [4]" and "Multi-objective feasibility enhanced particle swarm optimization [5]" respectively.

CHAPTER 2

REVIEW OF THE STATE OF THE ART

2.1 Definition and types of optimization problems

Optimization is routinely used in all fields of science and engineering. The Oxford Dictionary defines optimization as "*the action of making the best or most effective use of a situation or resource*". Theoretically, performing an optimization task in a problem means finding the most or best suitable solution of the problem [6]. Not unexpectedly, mathematical studies concentrate on properties of an ideal solution and methods to reach such a solution whereas practical optimization studies often deal with an approximate solution due to restrictions on computation power and time.

An optimization problem can either be stated as maximization or minimization of a function. These two problems can easily be converted to the other form such that to *minimize*

$$y = f(\mathbf{x}) \tag{2.1}$$

is equivalent to *maximize*

$$y = -f(\mathbf{x}) \tag{2.2}$$

where $f(\mathbf{x})$ is the objective function and $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T$ is the decision vector. If all of the problem's decision variables are real valued then $\mathbf{x} \in \mathbb{R}^{N\times 1}$. N is the number of decision variables which defines the dimension of the problem. Decision variables are usually bounded such that

$$x_n^L \leq x_n \leq x_n^U, \quad n \in \mathbb{N}_{>0}^{\leq N} \tag{2.3}$$

2.1.1 Multi-objective optimization problems

Many real-world optimization problems are interested in optimizing more than one measure simultaneously. In that case the optimization problem can be stated as *minimize*

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \equiv \left[\begin{array}{cccc} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \cdots & f_K(\mathbf{x}) \end{array} \right]^T \tag{2.4}$$

where K is the number of objective functions. It would be very improbable for these K functions to have minimums exactly at the same point. Therefore, in one way or another, minimizing these functions is contradicting. Unless all the functions being optimized in a multi-objective optimization problem have optimums at exactly the same point, we can not speak of an optimum. Hence, the scalar concept of optimality does not apply directly in the multi-objective setting [7].

Historically, multi-objective problems (MOPs) have been converted into single-objective ones to allow application of conventional optimization methods. One of the most common approaches is to utilize a fixed weight linear aggregation function as the single objective such that

$$y = f(\mathbf{x}) = \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{w_k f_k(x)} \tag{2.5}$$

where w_k are the constant weights that define the importance of each objective function. Note that this approach requires determination of the weights beforehand, hence it carries the risk of introducing bias or scaling flaws if no prior knowledge related with the problem exists. Instead, the concept of Pareto optimality can be employed to obtain Pareto optimal solutions that represent the best trade-off solutions. Therefore, the obtained solution set in a Pareto optimality based technique is truly multi-objective.

2.1.1.1 Pareto optimality

Before making a formal definition of Pareto optimality, the concept of *domination* must be explained. Domination is a relation defined between two solutions (or objective vectors of the solutions). Consider two decision vectors $\mathbf{x^a}$ and $\mathbf{x^b}$ of solutions a and

b. The corresponding objective vectors of these solutions would be

$$\mathbf{y^a} = \mathbf{f}(\mathbf{x^a}) = \begin{bmatrix} y_1^a & y_2^a & \cdots & y_K^a \end{bmatrix}^\mathbf{T} \tag{2.6}$$

$$\mathbf{y^b} = \mathbf{f}(\mathbf{x^b}) = \begin{bmatrix} y_1^b & y_2^b & \cdots & y_K^b \end{bmatrix}^\mathbf{T} \tag{2.7}$$

In a minimization problem, the vector $\mathbf{y^a}$ is said to dominate $\mathbf{y^b}$ iff (if and only if)

$$\left(\forall k \in \mathbb{N}_{>0}^{\leq K}\right) \left(y_k^a \leq y_k^b\right) \wedge \left(\mathbf{y^a} \neq \mathbf{y^b}\right) \tag{2.8}$$

Domination of $\mathbf{y^a}$ over $\mathbf{y^b}$ is denoted as

$$\mathbf{y^a} \prec \mathbf{y^b} \tag{2.9}$$

and implies that all components of vector $\mathbf{y^a}$ are as good as $\mathbf{y^b}$ and since $\mathbf{y^a} \neq \mathbf{y^b}$, at least one component is better for sure[1] [8].

Pareto optimality can only be defined in the existence of a solution set that contains multiple solutions to a problem. Therefore, let the set of decision vectors of all solutions found for a problem be

$$\mathbf{X} = \left\{ \mathbf{x}_\theta = \begin{bmatrix} x_{\theta,1} & x_{\theta,2} & \cdots & x_{\theta,N} \end{bmatrix}^\mathbf{T} : \theta \in \mathbb{N}_{>\mathbf{0}}^{\leq \boldsymbol{\Theta}} \right\} \tag{2.10}$$

where Θ is the number of solutions in the set. A solution $\mathbf{x}_* \in \mathbf{X}$ is Pareto optimal iff

$$(\nexists \theta \in \mathbb{N}_{>0}^{\leq \Theta}) \left(\mathbf{f}(\mathbf{x}_\theta) \prec \mathbf{f}(\mathbf{x}_*)\right) \tag{2.11}$$

In other words, if a solution is not dominated by any other solutions, it is a Pareto optimal solution. The concept of a solution being not dominated by any other solutions is also called non-domination. Hence, a Pareto optimal solution is *non-dominated*. The set of all Pareto optimal solutions is called the *Pareto set*. The image of the Pareto set in the objective space is the *Pareto front* [8]. Figure 2.1 illustrates Pareto front in a two dimensional objective space ($K = 2$) where Pareto front is likely to form a curve.

2.1.1.2 ε-dominance

Pareto dominance provides a binary comparison of an individual with other individuals it dominates. However, it doesn't provide a measure of how much it dominates another

---

[1] Note that $\mathbf{y^a} \prec \mathbf{y^b}$ is used to denote that $\mathbf{y^a}$ dominates $\mathbf{y^b}$ in a minimization problem. A maximization problem would require the same domination rule to be shown as $\mathbf{y^a} \succ \mathbf{y^b}$.

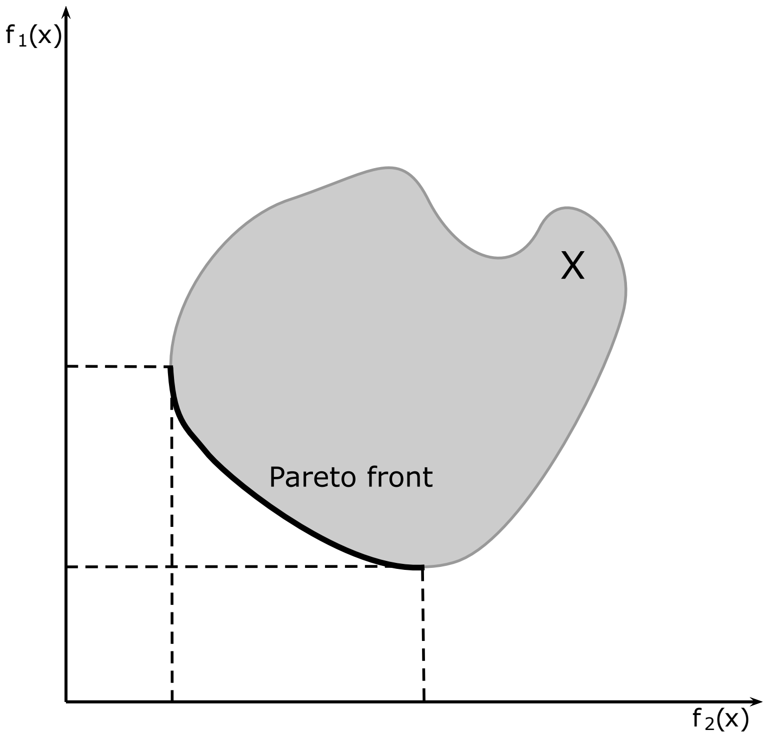


Figure 2.1: Illustration of Pareto front in two dimensional objective space.

individual or how close an individual is to dominate another individual. ε-dominance tries to put a measure on domination.

An objective function vector $\mathbf{f}(\mathbf{x}^*)$ is said to ε-dominate another objective function vector $\mathbf{f}(\mathbf{x})$ if $\mathbf{f}(\mathbf{x}^*) \prec \mathbf{f}(\mathbf{x}) + \mathbf{u}_k\varepsilon$, where $\mathbf{u}_k$ is a vector of ones of size K and $\varepsilon \geq 0$. ε-dominance is denoted as $\mathbf{f}(\mathbf{x}^*) {\prec_\varepsilon} \mathbf{f}(\mathbf{x})$. Note that if $\varepsilon = 0$, ε-domination is equivalent to Pareto domination. ε-domination is a weaker domination for $\varepsilon > 0$ such that:

$$[\mathbf{f}(\mathbf{x}^*) \prec \mathbf{f}(\mathbf{x})] \Rightarrow [(\mathbf{f}(\mathbf{x}^*) \prec_\varepsilon \mathbf{f}(\mathbf{x})] \qquad \text{for} \qquad \varepsilon > 0 \tag{2.12}$$

ε-domination can also be defined multiplicatively instead of additively. In that case,

$$\mathbf{f}(\mathbf{x}^*) {\prec_\varepsilon} \mathbf{f}(\mathbf{x}) \quad \text{if} \quad \mathbf{f}(\mathbf{x}^*) \prec (1+\varepsilon)\mathbf{f}(\mathbf{x}) \tag{2.13}$$

2.1.2 Constrained optimization problems

Some problems introduce limitations on the values decision vectors can take. In a broader sense, decision vectors are to be chosen from a restricted set such as $\mathbf{x} \in \mathbf{F}$. Here, $\mathbf{F}$ is the *feasible* portion of the decision hyper-space where allowable decision vectors lie. Although decision variable limits shown in Eq. 2.3 also impose restrictions on the decision vector, they are usually taken as the definition of the decision space ($\mathbf{U}$) such that

$$\mathbf{U} = \left\{ \mathbf{x} : \left(\mathbf{x_n^L} \leq \mathbf{x_n} \leq \mathbf{x_n^U} \wedge \mathbf{n} \in \mathbb{N}_{>\mathbf{0}}^{\leq \mathbf{N}} \right) \right\} \tag{2.14}$$

Note that, $\mathbf{U}$ is an N-dimensional hyper-space. Further constraints are generally in the form of inequalities and equalities in the following form:

$$\mathbf{F} = \mathbf{F^i} \cup \mathbf{F^e} \tag{2.15}$$

$$\mathbf{F^i} = \left\{ \mathbf{x} \in \mathbf{U} : \mathbf{g_m}(\mathbf{x}) \leq \mathbf{0} \wedge \mathbf{m} \in \mathbb{N}_{>\mathbf{0}}^{\leq \mathbf{M}} \right\} \tag{2.16}$$

$$\mathbf{F^e} = \left\{ \mathbf{x} \in \mathbf{U} : \mathbf{h_p}(\mathbf{x}) = \mathbf{0} \wedge \mathbf{p} \in \mathbb{N}_{>\mathbf{0}}^{\leq \mathbf{P}} \right\} \tag{2.17}$$

where g_m, M, h_p, and P are inequality constraint functions, number of inequality constraint functions, equality constraint functions, and number of inequality constraint functions respectively. For practical reasons, equality constraints are usually transformed into inequality constraints using $|h_p(\mathbf{x})| - \varepsilon \leq 0$ where ε is a non-negative real number chosen by the user. Therewith, $\mathbf{F} = \mathbf{F^i}$ and any solution $\mathbf{x} \notin \mathbf{F}$ is called *infeasible*.

As a result, a constrained multi-objective optimization problem (CMOP) can be defined in the general form as *minimize*

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \equiv \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \cdots & f_K(\mathbf{x}) \end{bmatrix}^T \tag{2.18}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq M} \tag{2.19}$$

The amount of constraint violation is often described using the *constraint vector* ($\mathbf{c}$) such that

$$\mathbf{c} = \mathbf{g}(\mathbf{x}) \equiv \begin{bmatrix} g_1(\mathbf{x}) & g_2(\mathbf{x}) & \cdots & g_M(\mathbf{x}) \end{bmatrix}^T \tag{2.20}$$

Note that, a solution is feasible iff

$$\left(\forall m \in \mathbb{N}_{>0}^{\leq M} \right) (c_m \leq 0) \tag{2.21}$$

2.1.3 Multimodal optimization problems

A multimodal optimization problem is a problem that has more than one local minimum. In a multimodal optimization problem it can be challenging to discover which minimum is the global minimum (i.e. if all possible minimums are found). Classical gradient methods easily fail in multimodal problems if necessary measures are not taken.

2.1.4 Combinatorial optimization

Not all optimization problems have continuous independent variables such that

$$\left(\exists n \in \mathbb{N}_{>0}^{\leq N}\right)\left(x_n \notin \mathbb{R}\right) \tag{2.22}$$

Many optimization problems deal with independent variables that are restricted to a set of discrete values. These types of problems are called “combinatorial optimization” problems.

The non-continuous nature of the independent variables render methods based on derivatives useless. In many cases trying each and every combination is out of question due to computational power or time limitations. Although it is impossible to be certain that one has the best solution without trying all, there are optimization techniques that provide a powerful way to find “good enough” solutions [9].

2.2 Classical optimization techniques

First optimization techniques depended on analytical methods provided by calculus. The simplest approach to finding the extrema of a function is to take its gradient and set it equal to zero such that

$$\nabla f(\mathbf{x}) = 0 \tag{2.23}$$

For example, let

$$f(x_1, x_2) = x_1 \sin(x_1) + x_2 \sin(2x_2) \tag{2.24}$$

where $x_1, x_2 \in \mathbb{R}_{\geq 0}^{\leq 10}$. Then,

$$\frac{\partial f}{\partial x_1} = \sin(x_1) + x_1 \cos(x_1) = 0 \tag{2.25}$$

and

$$\frac{\partial f}{\partial x_2} = \sin(2x_2) + 2x_2 \cos(2x_2) = 0 \tag{2.26}$$

A set of lines are obtained from the solution of these equations for x_1 and x_2. Intersections of these lines are the extrema of the function. Unfortunately, to be able to determine if these extrema are minima, one has to also check if $\nabla^2 > 0$. However, no information can be derived to indicate if any of these points is the global minima. Hence, the list of all extrema must be traversed anyway. This approach has several important limitations. Functions must be continuous and their analytical derivatives must exist. Therefore, considering the complexity of present design problems, it is highly impractical to employ such techniques.

2.2.1 Lagrange multipliers

The method introduced by Lagrange incorporates equality constraints to the analytical solution of the optimization problem. Therefore, the method deals with minimization of

$$y = f(\mathbf{x}) \tag{2.27}$$

subject to

$$h_p(\mathbf{x}) = 0 \tag{2.28}$$

where $p \in \mathbb{N}_{>0}^{\leq P}$. The method of Lagrange multipliers converts the problem into finding the extrema of a new function defined as follows [10, 11].

$$F(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{p=1}^{P} \lambda_p h_p(\mathbf{x}) \tag{2.29}$$

The new variables λ_p ensure satisfaction of equality constraints. Once again, Lagrange multipliers method is not widely used due to practical limitations. However, many modern methods are based on this approach. It is also possible to solve problems with inequality constraints ($g_m(\mathbf{x}) \leq 0$) through conversion of these constraint to equalities by adding nonnegative slack variables(δ^2) such that [12]

$$h_m = g_m(\mathbf{x}) + \delta^2 = 0 \tag{2.30}$$

2.2.2 Convex programming problem

Consider an optimization problem stated as *minimize*

$$y = f(\mathbf{x}) \tag{2.31}$$

subject to

$$g_m(\mathbf{x}) \leq 0 \tag{2.32}$$

This problem is called a *convex programming problem* if the objective function ($f(\mathbf{x})$) and the constraint functions ($g_m(\mathbf{x})$) are convex. As discussed in the previous section, the Lagrange function of this problem can be written as follows:

$$F(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\delta}) = f(\mathbf{x}) + \sum_{m=1}^{P} \lambda_m (g_m(\mathbf{x}) + \delta_m^2) \tag{2.33}$$

Note that if $\lambda_m \geq 0$, then $\lambda_m g_m(\mathbf{x})$ is also convex. Under these conditions, $F(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\delta})$ is also convex and its derivative becomes zero only at a single point which must be the absolute minimum.

2.2.3 Linear programming

Linear programming is an optimization method used in problems consisting of a linear objective function and linear constraint functions of many decision variables. The method is originally formulated by Wood and Dantzig [13] and Dantzig [14]. Although many algorithms emerged to solve LP problems, simplex algorithm of Dantzig remains the most popular. Simplex method is a powerful technique for solving LP problems. However, only a few engineering design problems involve a single linear objective function and linear constraint functions.

2.2.4 Nonlinear programming

As mentioned in previous sections, if objective function and constraint functions of the optimization problem are simple, some analytical techniques can be employed. However if functions that define the problem are complex and hard to manipulate these classical methods become obsolete.

Many numerical methods exist for dealing with optimization problems. Most of these methods share some basic principles and follow the steps outlined below:

1. Start with a trial point $\mathbf{x}^*$

2. Determine a direction $\mathbf{u}$ which is supposed to improve the solution
3. Determine the step size κ
4. Update the trial point by $\mathbf{x}^* := \mathbf{x}^* + \kappa\mathbf{u}$
5. Check if the optimum is reached. If not, continue to update trial point.

Approaches such as the golden section method, Fibonacci method, secant method, and the family of interpolation methods are some examples that deal with one-dimensional problems ($N = 1$).

Some nonlinear programming methods such as the random search method or the univariate method do not require derivatives. These approaches are commonly classified as *nongradient methods*. However, methods like the steepest descent or Newton's method depend on derivatives of the objective function, and therefore are called *gradient methods*.

The univariate method modifies the value of only one decision variable in each iteration. Although it performs well in certain types of problems, its convergence is not guaranteed and it might require many iterations.

2.2.4.1 Nelder-Mead simplex method

Most famous nongradient technique is probably the Nelder-Mead simplex method [15]. A simplex is the most fundamental geometric shape that can be constructed in N dimensions. When $N = 1$, a line is a simplex; when $N = 2$ a triangle is a simplex; wheres when $N = 3$ tetrahedron is a simplex. Therefore a simplex has $N + 1$ edges- and vertices. Nelder-Mead simplex method works in the following way:

1. Create an initial simplex using $N + 1$ vertices: $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{N+1}$
2. Reflect the worst vertex through the centroid of the hyperplane formed by the remaining vertices (Figure 2.2).
3. If the reflected point produces a new minimum, further expand the simplex in the same direction.

4. If the reflected point becomes the worst once again, the simplex is contracted using a predefined coefficient.

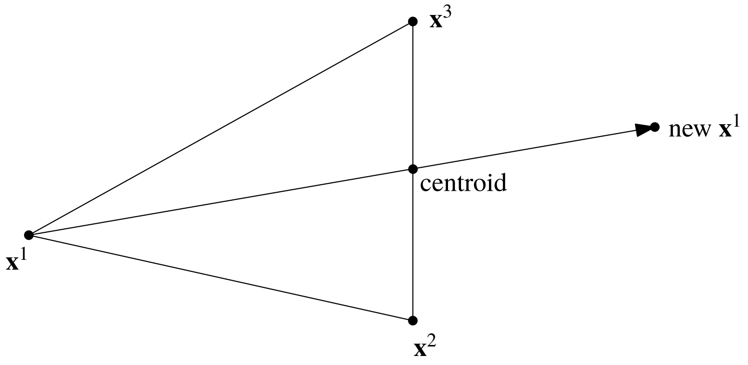


Figure 2.2: Illustration of reflection in Nelder-Mead simplex algorithm in two dimensions ($N = 2$)

Box [16] improved the Nelder-Mead simplex algorithm and allowed inclusion of inequality constraints. He called this modified approach the *complex method*.

2.2.4.2 Steepest descent method

Steepest descent method originated with Cauchy [17] and became a popular gradient based technique. It simply uses the negative of the objective function's gradient as the search direction in each iteration such that

$$\mathbf{u} = -\nabla f(\mathbf{x}) \tag{2.34}$$

Owing to the fact that gradient of the objective function is a local property, the steepest descent method always converges to a local minima if no special technique is employed. Steepest descent method is not widely used anymore due to emergence of more efficient algorithms. Many of these algorithms are modifications of Newton's Method.

2.2.4.3 Newton's method

Newton's method is based on the quadratic approximation of the objective function using its Taylor's series expansion

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^T \nabla f(\mathbf{x}^*) + \frac{(\mathbf{x} - \mathbf{x}^*)^T}{2!} \mathbf{H}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \tag{2.35}$$

where $\mathbf{x}^*$ is the point about which Taylor series is expanded and $\mathbf{H}$ is the Hessian matrix of second derivatives. By setting the gradient of Eq. 2.35 to zero for the minimum of $f(\mathbf{x})$ we obtain

$$\nabla f(\mathbf{x}^*) + \mathbf{H}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) = 0 \tag{2.36}$$

which can be solved for a better approximation as follows:

$$\mathbf{x}^* := \mathbf{x}^* - \mathbf{H}(\mathbf{x}^*)^{-1} \nabla f(\mathbf{x}^*) \tag{2.37}$$

Note that the Hessian of the objective function is rarely known. Therefore many algorithms exist for numerically approximating the Hessian matrix including the approaches used by the Davidon-Fletcher-Powell (DFP) algorithm [18] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [19–22].

Sequential quadratic programming (SQP) can considered to be a generalization of the Newton's method. It uses a quadratic approximation instead of the objective function [23]. The technique iteratively constructs an approximation to solve a local quadratic programming problem to form a starting point for the next iteration. There are many recent variants of the SQP.

2.3 Evolutionary algorithms

Virtually all of the algorithms mentioned in Section 2.2 are based on the same idea. They try to move an initial trial point downhill using different approaches in deciding which way to move and how much. However, these approaches mostly depend on local properties of the functions and are generally stuck at local extrema. Recent algorithms that are inspired in different ways from nature have proven to be efficient in many cases where limitations imposed by calculus render classical methods impractical. Evolutionary algorithms are the most commonly used nature inspired algorithms.

Evolutionary algorithms (EAs) can be defined as algorithms that evolve a problem solution over many iterations. These iterations are generally called *generations* and the way an EA evolves solutions is almost always inspired by nature. EAs that deal with MOPs are called *multi-objective evolutionary algorithms* (MOEAs) since their first use by Schaffer [24]. Amount of studies on EAs and MOEAs are tremendous. Along with the classics such as the genetic algorithm [25], non-dominated sorting genetic algorithm (NSGA) [26–28], simulated annealing [29], and differential evolution [30] there never is a lack of new EAs.

There are many published surveys of the state of the art of MOEAs in general [8, 9, 31] and in some specific topics such as constraint handling [32, 33] and applications to specific areas of interest like mechanical systems [34, 35] or aerospace engineering [1, 36, 37].

Evolutionary algorithms have been applied to many different kinds of optimization problems which were previously solved by relatively older types of optimization algorithms such as hill climbing or simplex methods. Problems in the mechanical engineering domain are no exceptions. As nature inspired algorithms, evolutionary optimization algorithms are commonly used in a wide range of problems with different characteristics. Although, most of these algorithms were initially developed for unconstrained problems, constraint handling techniques evolved to allow their use in constrained problems including CMOPs.

Many EAs share similar principles outlined in Figure 2.3. They use a set of individuals[2] (i.e. points in the decision space) called the population[3] to evolve new and assumedly better solutions. Therefore, the first step is the initialization of the population which generally involves randomly distributing individuals in the decision space. All other operations are generally performed in a loop until a set of termination criteria is met. In each iteration of the loop, constraints and objectives associated with each individual are calculated. There are several different approaches for these calculations. Generally, all objective- and constraint functions are simultaneously calculated for the whole population. However, there are algorithms where objectives are only calculated if the constraints are satisfied. This type of algorithms require specialized techniques since

---

[2] Individuals of the population are called *particles* in swarm intelligence algorithms.

[3] As the name implies, population is referred to as *swarm* in swarm intelligence algorithms.

no associated objective vector exists for some individuals. The basic principle for evolution of individuals is through selection of the best individuals in the population (or sometimes sorting them from best to worst) and using their features to combine with features of the other individuals. Real differences between particular EAs lie in the mechanisms they implement for the selection and combination of different individuals. These mechanisms almost always involve stochastic procedures. Some EAs implement a mechanism to randomly generate new features to bring additional exploratory potential to the algorithm. This type of behavior is commonly referred to as mutation. Although, simplest mutation approach would be to randomly alter some properties defining a randomly selected individual, many approaches with varying complexities exist. Another technique that many EAs employ is to maintain a separate archive of best individuals. Archiving is mainly used to have a record of the best solutions attained in the previous iterations which might be lost since the population is very dynamic. Individuals kept in the archives are also used in selection by some EAs.

Some major evolutionary algorithms are briefly described in the following sections. Special emphasis is given to genetic algorithms due to their importance. Particle swarm optimization is also depicted in more detail since the information presented is relevant for the following chapters.

2.3.1 Global diversity evolutionary multi-objective optimizer - GDEMO

A very basic form of an evolutionary algorithm for MOP is given in Algorithm 1 [9]. Global diversity evolutionary multiobjective optimizer (GDEMO) [38] is built upon this idea but uses ε-dominance instead of Pareto dominance. GDEMO includes an individual only if it is ε-nondominated. This actually makes it harder for an individual to be included.

2.3.2 ε based multiobjective evolutionary algorithm - ε-MOEA

ε based multiobjective evolutionary algorithm (ε-MOEA) maintains an archive besides the population. One individual from the population and one from the archive are crossed over to create a new individual. If the new individual dominates some

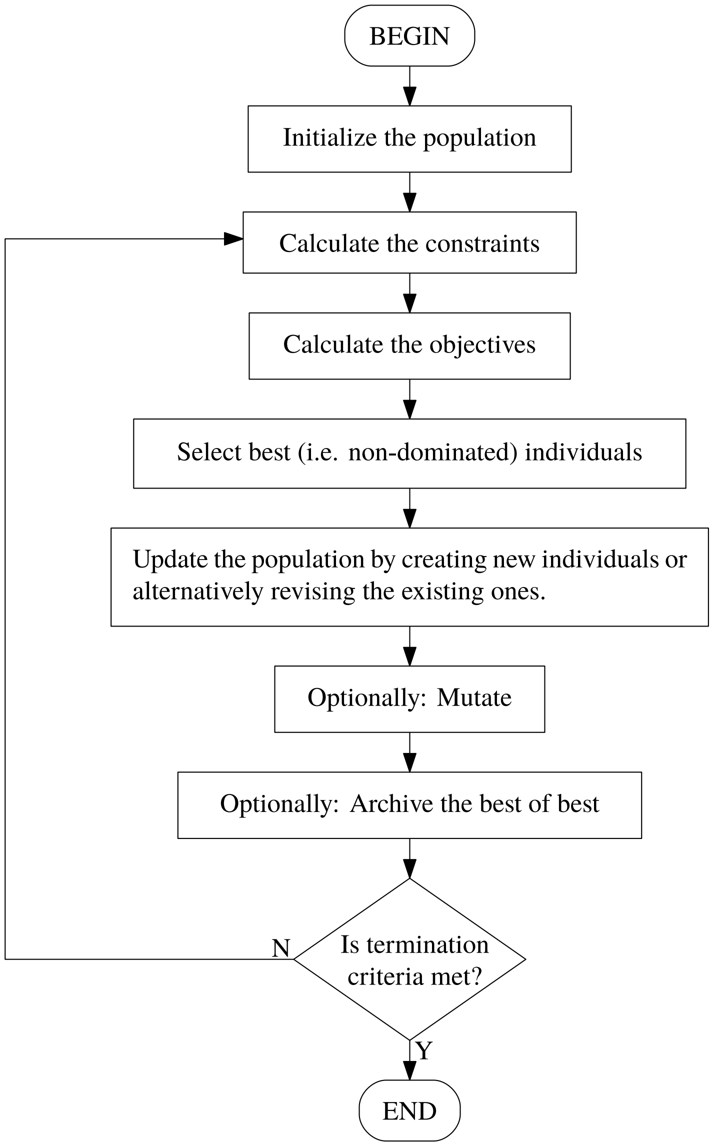

BEGIN
Initialize the population
Calculate the constraints
Calculate the objectives
Select best (i.e. non-dominated) individuals
Update the population by creating new individuals or alternatively revising the existing ones.
Optionally: Mutate
Optionally: Archive the best of best
Is termination criteria met?
N
Y
END

Figure 2.3: A generalized evolutionary algorithm flowchart for constrained multi-objective optimization problems

Algorithm 1 Global simple evolutionary multi-objective optimizer (GSEMO)

```
Initialize population with random individuals. $P := \{\mathbf{x}_{1,*}, \mathbf{x}_{2,*}, \ldots, \mathbf{x}_{I,*}\}$
Compute the objective function vectors (cost functions) $\mathbf{F}(\mathbf{x_i})$
repeat
    $y$ := randomly selected individual from $P$
    $z$ := randomly mutate $y$
    if $y$ is nondominated in $P$ then
        $P := \{P, y\}$
        Remove individuals $y$ dominates from $P$
    end if
until Termination condition
```

individuals in the population, it replaces one of them randomly. The new individual may be included in the archive and may remove some individuals from the archive based on Pareto dominance and ε-dominance [9, 39].

2.3.3 Simulated annealing - SA

SA is a metallurgy inspired global optimization technique [29]. It has also been utilized in MOPs with both Pareto based approaches [40] and non-Pareto based approaches [8, 41]. As seen in Algorithm 2, simulated annealing is a simple but very effective technique.

2.3.4 Genetic algorithms - GA

The term “genetic algorithm” has become to describe something very different from its initial depiction by Holland [25]. Holland’s original goal was to study the phenomenon of adaptation in nature and to develop ways to import mechanisms of natural adaptation into computer systems [42].

The easiest way to define genetic algorithms would be to call them “simulations of natural selection”. Although they weren’t originally developed for optimization, they serve well in this kind of problems. The working principle of genetic algorithms is

Algorithm 2 Simulated annealing

```
Set the initial temperature $T$
Define a cooling function $\alpha(T)$
Generate an initial solution $\mathbf{x}$
repeat
    Generate a candidate solution $\mathbf{x}^*$
    if $f(\mathbf{x}^*) < f(\mathbf{x})$ then
        $\mathbf{x} := \mathbf{x}^*$
    else
        Generate a random number $r \in \mathbb{R}_{\geq 0}^{\leq 1}$
        if $r < \exp\left[\left(f(\mathbf{x}) - f(\mathbf{x}^*)\right)/T\right]$ then
            $\mathbf{x} := \mathbf{x}^*$
        end if
    end if
    $T = \alpha(T)$
until Termination condition
```

best explained in its most basic form, namely the binary genetic algorithms.

2.3.4.1 Binary genetic algorithm

Binary genetic algorithms are the simplest form of genetic algorithms where properties of the system are coded in binary. Many alternative methods for each step of the binary genetic algorithm exists to support different applications. Fundamental steps of a typical genetic algorithm is shown with the flowchart in Figure 2.4.

The decision vector $\mathbf{x}$ of a genetic algorithm can be represented as a vector of real numbers, discrete numbers, a permutation of entities or a combination of these as suitable to the underlying problem [6]. As the name implies, binary genetic algorithms implement a string of binary digits. Sections of this string represent different traits of an individual. The complete string is concatenation of these n traits and looks like:

$$\underbrace{01101}_{x_1}\underbrace{01}_{x_2}\ldots\underbrace{110}_{x_n}.$$

For example, the section marked as x_2 in this string might represent the individual

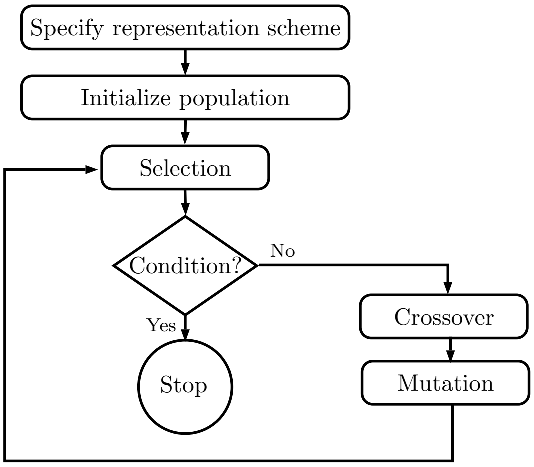


Figure 2.4: Basic principle of a genetic algorithm [6]

solution's type of material with the following coding:

$$\begin{aligned} 00 &= \text{aluminum} \\ 01 &= \text{steel} \\ 10 &= \text{titanium} \\ 11 &= \text{copper} \end{aligned}$$

Each bit is called an *allele*. A sequence of bits in an individual that contains information about some trait of that individual is called a gene. Specific genes are called genotypes, and the problem-specific parameter that a genotype represents is called a phenotype [9]. In our example two bits shown with x_2 is a gene. The individual shown in the example has the material genotype of 01, which corresponds to the material phenotype of "steel". The collection of all genes in an individual is called a chromosome.

Since genetic algorithms mimic biology, they need an initial population of solutions from which they select individuals to breed and reproduce new individuals (solutions). Usually, this initial population is created by a randomly generated set of solutions (size N). However, if previously known good solutions exist, they might be included in the initial population. Additional individuals can be created around a good solution by randomly perturbing it.

Selection is the operation that selects individual chromosomes in the current generation of the population for reproduction. An individual might be selected more than once. The fitter the chromosome, the more times it is likely to be selected to reproduce [42].

Therefore, the algorithm needs to evaluate the fitness of each and every individual of the generation. Needless to say this evaluation requires to first decipher the solution vector according to the chosen representation scheme and then calculate the fitness of each individual through a suitable fitness measure. In a single-objective minimization problem the value of the objective function multiplied by -1 can be used as the fitness.

The algorithm has to decide which individuals mate to produce children. Several different methods exist for this last step of the selection operation and one of the most popular is the *fitness-proportionate selection*. This selection scheme ensures that probability of selection of an individual is proportional to its fitness. In other words, each individual has a probability of being selected that is proportional to its fitness. For the first parent, the algorithm would select an individual as described above. For the second parent, the algorithm must ignore the previously selected parent and select from the remaining individuals in the same manner, i.e., with a probability proportional to their fitness.

Reproduction occurs through crossover. Crossover operation is for creation of new individuals for the next generation. When two parents are crossed over, two children are created. For this purpose a random crossover point is selected and alleles of parents after this point are swapped (see Figure 2.5).

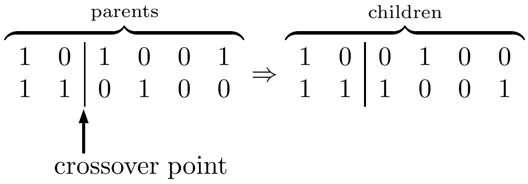


Figure 2.5: Illustration of a crossover operation

Mutation is basically randomly flipping bits of a chromosome string. Mutations occur rarely in living organisms and they should also occur rarely in genetic algorithms. Mathematically, mutation in genetic algorithm is implemented such that each allele of every individual chromosome string may flip with a predefined probability (such as 1%). Mutation aims to provide genetic variability by allowing variations that may not have been introduced with the initial population. Missing genetic information has a chance of being injected through mutation(s). This is more important for genetic algorithms where population size is in the order of hundreds against the millions in biology.

2.3.4.2 Real-coded genetic algorithm

Continuous decision variables can be discretized to be represented by binary genes. However, they can also be represented as a single real valued allele:

$$3.14 \quad 2.718 \quad 0.577 \quad 4.6692 \quad 1.618$$

Flowchart given in Figure 2.4 is principally true for real-coded genetic algorithms too. However, different crossover and mutation operators are applied due to real-coded variables instead of binary.

Many different crossover and mutation operators exist for use in real-coded problems. The simplest crossover operator to be used in a real-coded genetic algorithm would be the one that we also used in the binary genetic algorithm:

$$\overbrace{\begin{array}{ccc|cc} 3.14 & 2.718 & 0.577 & 4.669 & 1.618 \\ 4.21 & 0.739 & 0.11 & 5.23 & 0.123 \end{array}}^{\text{parents}} \Rightarrow \overbrace{\begin{array}{ccc|cc} 3.14 & 2.718 & 0.577 & 5.23 & 0.123 \\ 4.21 & 0.739 & 0.11 & 4.669 & 1.618 \end{array}}^{\text{children}}$$

However, this simple crossover operator simply exchanges values of decision variables between individuals and relies solely on mutation to create new values of decision variables that don't already exist in the initial population. In binary genetic algorithms however, since a variable is coded as several binary alleles, new values may occur based on the crossover operator or the crossover point.

2.3.5 Non-dominated sorting genetic algorithm - NSGA2

NSGA2 is one of the most popular and successful constrained multi-objective optimization algorithms. The successor of NSGA2 (non-dominated sorting genetic algorithm - II) was proposed by Srinivas and Deb [26]. The original algorithm, called NSGA, involved assigning cost of individuals based on how dominant they are. NSGA finds all non-dominated individuals, ranks them, and then removes them from fitness evaluation to find the next layer of non-dominated individuals. This procedure continues until all individuals are assigned a cost.

NSGA2 [27, 28] computes cost of individuals by evaluating not only the individuals that dominate them but also takes into account the individuals that they dominate. NSGA2 also uses the concept of *crowding distance* based on the distance to nearest individuals in the objective space to modify the fitness.

Many widely used algorithm frameworks including multi-objective genetic algorithm (MOGA) [43] and niched Pareto genetic algorithm (NPGA) [44] are based on or similar to NSGA2 [8].

2.3.6 Particle swarm optimization - PSO

Particle swarm optimization (PSO) is a swarm intelligence algorithm devised by Kennedy and Eberhart [45, 46]. PSO has attracted a great attention in the last two decades - especially for real-valued problems. The basic algorithm of canonical PSO as standardized by Bratton and Kennedy [47] has been effectively used in a wide range of problems both single- and multi-objective to produce satisfactory results with relatively low computational cost [48–52]. The readers may refer to the surveys presented by Banks, Vincent, and Anyakoha [53, 54] for history, classifications, and applications of PSO.

In PSO, the swarm consists of a certain number of particles which fly through the multidimensional search space. These particles change their position based on their personal experiences (the best position they remember to have been in) and success of other individuals in the swarm (i.e. their neighbours or all particles in the swarm). The behavioral characteristic of the particle that is driven by its previous experiences is called the cognitive influence. Each particle modifies its position at every iteration based on: current position, current velocity, distance between current position and its personal best (personal guide: *pbest*), and distance between current position and position of the leader (global guide: *gbest*). The influence of the best particle in the swarm or a group of particles within the swarm is called the social influence. Which particle or position (i.e. solution) influences the particle is determined by the neighborhood topology. Although, majority of PSO applications implement a topology called "*all topology*" where the social influence is driven by the best particle chosen from the entire swarm, other alternatives exist [55]. In some topologies, the social driver is chosen from a group of particles called the neighbors. All PSO topologies used in this study are the all topology. A flowchart of basic PSO algorithm is shown in Figure 2.6.

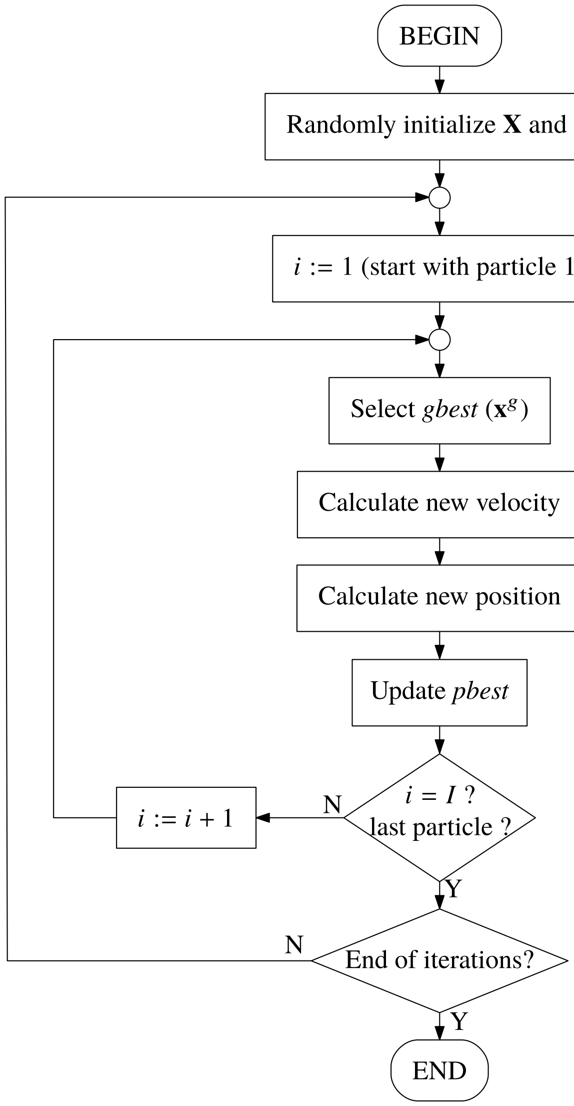


Figure 2.6: Canonical PSO flowchart

Let the current positions and velocities of particles in the swarm be

$$\mathbf{X} = [x_{i,n}] \in \mathbb{R}^{I \times N} \tag{2.38}$$

and

$$\mathbf{V} = [v_{i,n}] \in \mathbb{R}^{I \times N} \tag{2.39}$$

respectively[4]. Here, $i \in \mathbb{N}_{>0}^{\leq I}$ and $n \in \mathbb{N}_{>0}^{\leq N}$; I is the number of particles in the swarm; N is the number of decision variables. Therefore, each row of $\mathbf{X}$,

$$\mathbf{x}_{i,*} = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,N} \end{bmatrix} \tag{2.40}$$

represents the transposed position vector of the corresponding particle[5]. These matrices need to be randomly initialized at the beginning of the PSO routine. $\mathbf{X}$ is initialized to comply decision variable limits (Eq. 2.3) whereas the velocity matrix is generally initialized such that its elements are a fraction of their corresponding decision space dimension size:

$$v_{i,n} := a^v \left(2r - 1\right) \left(x_n^U - x_n^L\right) \tag{2.41}$$

where $r \in [0, 1] \subset \mathbb{R}$ is a uniformly-distributed random number and $a^v \in \mathbb{R}$ is a factor smaller than one. Since particles need to remember the best position they have been, a matrix of personal best positions must also be defined. The best positions at the beginning would be the initialized positions such that

$$\mathbf{X}^p = \mathbf{X} \tag{2.42}$$

As seen in Figure 2.6, PSO involves selection of *gbest* ($\mathbf{x}^{\mathbf{g}}$), velocity update, position update, and $pbest$ update for each particle in every iteration. *gbest* is selected as the position of the particle that has the best objective value. Formally:

$$\mathbf{x}^{\mathbf{g}} = \left(\mathbf{x}_{\varphi,*}\right)^{\mathbf{T}} \tag{2.43}$$

where φ is chosen such that

$$\left(\nexists i \in \mathbb{N}_{>0}^{\leq I}\right) \left(f\left(\left(\mathbf{x}_{i,*}\right)^T\right) < f\left(\left(\mathbf{x}_{\varphi,*}\right)^T\right)\right) \tag{2.44}$$

The velocity is calculated using

$$v_{i,n} := W v_{i,n} + C_1 r_{1,n} \left(x_n^G - x_{i,n}\right) + C_2 r_{2,n} \left(x_n^P - x_{i,n}\right) \tag{2.45}$$

---

[4] Note that $\mathbf{X}$ is redefined here as the matrix of particle positions in the decision space instead of an arbitrary solution set used in Eq. 2.10.

[5] Note that $\mathbf{x}_{i,*}$ represents the row vector that is the ith row of the $\mathbf{X}$ matrix. This convention is used throughout this dissertation.

where W is the inertia factor, C_1 is the social acceleration coefficient, C_2 is the cognitive acceleration coefficient while $r_{1,n}, r_{2,n} \in [0,1] \subset \mathbb{R}$ are uniformly-distributed random numbers. The choice of W, C_1, and C_2 may effect the performance of the algorithm. W is often reduced as the iterations commence (linearly reduced values of $W = 0.9$ at the first iteration towards $W = 0.4$ at the last are commonly used) to provide a more explorative behavior in the beginning. Although a fixed value of $C_1 = C_2 = 2$ is widely used in the literature, no major performance change is observed when C_1 and C_2 are set to values in the range $[1.5, 2.5]$. However, selection of these parameters may become important in different types of problems such as ones that are relatively less constrained when compared with problems employed in this study.

The updated velocity is used to calculate a new position such that

$$x^*_{i,n} := x_{i,n} + v_{i,n} \tag{2.46}$$

The new position is then used to check if the $pbest$ of the particle of interest requires updating. Therefore, if

$$f\left((\mathbf{x}^p_{i,*})^T\right) < f\left((\mathbf{x}_{i,*})^T\right) \tag{2.47}$$

then

$$\mathbf{x}^p_{i,n} := \mathbf{x}_{i,n} \tag{2.48}$$

2.3.6.1 A fundamental multi-objective PSO approach

Earliest published papers that use PSO in a multi-objective context are due to Coello and Lechuga [56] and Ray, Tai, and Seow [57]. Practically, the simplest scheme to convert PSO into a multi-objective algorithm only requires modification of approaches related with storage and selection of *gbest* and *pbest*:

1. In multi-objective PSO a single $gbest$ rarely exists. Instead, $gbest$ is randomly selected from non-dominated particles.
2. Similarly, $pbest$ must also be selected from the positions that are non-dominated among the positions that the particle has explored in the previous iterations.
3. An archive of non-dominated solutions must be created for each particle. The archive needs to be updated with each particle movement.

2.3.6.2 Constraint handling in PSO

PSO has also been applied to constrained optimization problems. Utilization of feasibility rules referred to as Deb's approach [58] has been the most popular choice in PSO based optimization techniques. Various other constraint handling concepts (such as the ε-constrained method [59], penalty functions [60, 61], the use of special operators [62–64], and conversion to multi-objective optimization [65]) are available in PSO literature. A very comprehensive survey on constraint handling methods used in PSO has been published by Jordehi [66].

Deb's approach[6] only requires modification of the domination principle explained in Section 2.1.1. Consider once again the two decision vectors $\mathbf{x^a}$ and $\mathbf{x^b}$ of solutions a and b. The corresponding objective vectors were given by Eqs. 2.6 and 2.7. Now, according to Eq. 2.20, constraint vectors of these two solutions can be defined such that

$$\mathbf{c^a} = \mathbf{g}(\mathbf{x^a}) = \begin{bmatrix} c_1^a & c_2^a & \cdots & c_M^a \end{bmatrix}^\mathbf{T} \tag{2.49}$$

$$\mathbf{c^b} = \mathbf{g}(\mathbf{x^b}) = \begin{bmatrix} c_1^b & c_2^b & \cdots & c_M^b \end{bmatrix}^\mathbf{T} \tag{2.50}$$

There may only be four cases:

1. If both solutions a and b are infeasible, the one with less constraint violation dominates the other.
2. If a is feasible and b is infeasible, a dominates b.
3. If b is feasible and a is infeasible, b dominates a.
4. If both solutions a and b are feasible, domination is decided according to Eq. 2.8

The only ambiguity in this definition is how "less constraint violation" is interpreted (when $M > 1$). While one can simply use

$$|\mathbf{c^a}| \overset{?}{<} |\mathbf{c^b}| \tag{2.51}$$

to determine if a dominates b, this may introduce scaling problems if elements of the $\mathbf{c}$ vectors have different orders of magnitude. There is no widely used convention for comparison of constraint violations.

---

[6] Also called *feasibility based domination* or *superiority of the feasible*

Due to its unique nature, some PSO approaches require particle level operations to handle constraints. Alvarez-Benitez, Everson, and Fieldsend [67] investigated ways to keep the particles within the feasible search space:

1. TRC: Truncate the location at the exceeded constraint boundary for this generation and reflect the velocity in the boundary so that the particle moves away [68].
2. Resample the stochastic terms in the velocity update formula until a feasible position is achieved [69].
3. Limit the velocity vector to stay in the feasible zone.
4. SHR: Like truncation but keeps the velocity direction the same to help the particle stay near the boundary.
5. EXP: Sample from a truncated exponential distribution oriented so that there is a high probability of samples close to the boundary and a lower probability of samples at the current position to allow particles that would have exceeded the boundaries to remain close to the boundaries.

2.3.6.3 PSO for discrete- or mixed variable optimization problems

PSO has also been utilized in discrete- or mixed variable optimization problems. Several different approaches exist for handling discrete variables in PSO:

- *Binary approaches:* Particle trajectories become probabilities for binary genotypes [70].
- *Special discrete operators:* A notion of movement in the search space is imposed by defining operations (like addition or subtraction) on positions and velocities [71–73].
- *Corresponding continuous space:* A continuous space, where the particles move in addition to the actual discrete space, is defined. Methods for transforming the positions in-between these spaces are also required [74].

- *Discrete velocity operators:* Velocity operators are defined according to positions that a particle is allowed to move [75, 76].

- *Truncation & transformation:* If the variable can only take pre-established real or integer values, it is truncated [77] or transformed [78] to the nearest possible value.

2.4 Design problems in the mechanical engineering domain

Almost all optimization problems involving mechanical system design are constrained. As highlighted in Table 2.1, many of these problems have more than several constraints and it is not uncommon for the feasible portion of the search space to be much smaller than the infeasible one. In highly-constrained problems, the process of finding a feasible solution becomes a major challenge of a similar scale for finding the optimum. Therefore, specialized approaches are required especially when the objective function is costly such as the problems involving mechanical system design. The cost of evaluating objective and constraint functions might significantly be different when constraint functions are generally easier to calculate. This is especially true when computer-aided design tools such as finite element method (FEM) are utilized. Furthermore, there may be cases where some constraints are very easy to evaluate whereas the rest could be as hard to evaluate as objective functions themselves. Note that there may be some extreme cases where it may be impossible to evaluate objective function(s) when some constraints are violated.

Although some simple benchmark problems may have functions of convex nature, sufficiently complicated mechanical design problems usually have non-convex objective- and constraint functions [79–82]. Besides, when computational techniques are used in solution of sophisticated problems such as large structural design problems, the designer usually does not have the necessary information to assess the characteristics (i.e. convexity) of the functions. Therefore, classical methods are generally not applicable to these problems.

Table 2.1: Some popular mechanical design problems

No.	Problem	Objectives	Constraints (other than variable bounds)	Design variables
1	Gear train [83–87]	1 nonlinear	6 nonlinear, 2 linear	5 continuous, 1 discrete
2	Radial ball bearing [88–90]	3 nonlinear	7 nonlinear, 2 linear	10 continuous
3	Multiple disc clutch brake [86, 91]	1 nonlinear	7 nonlinear, 1 linear	4 continuous(1), 1 discrete
4	Robot gripper [86, 92]	1 nonlinear	5 nonlinear, 1 linear	7 continuous
5	Four-stage gear train [86, 93–98]	1 nonlinear(2)	86 nonlinear	22 discrete(2)
6	Mixed-valued four-stage gear train (Originates from our study)	1 nonlinear	86 nonlinear	10 continuous, 12 discrete
7	Pressure vessel [2, 34, 60, 86, 99–107]	1 nonlinear	1 nonlinear, 3 linear	2 continuous, 2 discrete
8	Welded beam [2, 86, 100–104, 106, 108]	1 nonlinear	5 nonlinear, 2 linear	4 continuous
9	Tension/compression spring [2, 86, 100–104, 106, 109]	1 nonlinear	3 nonlinear, 1 linear	2 continuous, 1 discrete
10	Multispeed planetary transmission [86, 90, 110]	1 nonlinear	10 nonlinear, 5 linear, 1 condition, 1 equality	9 discrete
11	Leg mechanism [111, 112]	3 nonlinear	8 nonlinear	6 continuous
12	Speed reducer [113–115]	2 nonlinear	11 nonlinear	7 continuous
13	Disk brake [116, 117]	2 nonlinear	3 nonlinear, 2 linear	3 continuous, 1 discrete
14	Automobile mass optimization [118]	1 nonlinear	12 nonlinear	49
15	Ball bearing pivot link [34, 119]	1 nonlinear	3 nonlinear, 7 linear	2 continuous, 2 integer
16	Coupling with a bolted rim [34, 120]	1 nonlinear	3 nonlinear	2 continuous, 1 integer, 1 discrete
17	Structure/topology/truss(4) [35, 121–124]	1 to 4 nonlinear	0 to many(5) of all types	1 to many(5) of all types

(1) Although the problem is formulated with discrete variables they can actually be treated as continuous.

(2) Although the problem is originally treated as single objective, it can easily be treated as multi-objective as proposed by Dolen, Kaplan, and Seireg [98].

(3) 14 of these variables can actually be treated as continuous which is the how continuous version of this problem is proposed in this study.

(4) Many problems fall into this category.

(5) Many is used to imply hundreds (generally $>$ 300).

2.5 Research opportunity

Vast majority of mechanical engineering design problems are constrained. As a matter of fact, even widely used benchmark problems of mechanical engineering domain have at least several constraints (as seen in Table 2.1). As discussed above, many of these problems involve hard-to-satisfy constraints where only a small feasible region exists. One of the first studies discussing approaches for handling highly constrained problems in the context of mechanical design optimization is by Silva [125].

Highly constrained optimization problems can be defined as problems where a minimal change to a feasible solution is very likely to generate an unfeasible one [126]. Outside the optimal control literature first papers discussing methods for solving highly constrained problems are due to Berna, Locke, and Westerberg [127] and Mistree, Hughes, and Phuoc [128].

Moreover, unlike their counterparts, many design problems of the mechanical engineering domain have constraints that involve physical limits. These physical limits often render other functions such as the objective function incalculable. This condition is especially relevant in the case of complicated numerical approaches such as the FEM or CFD. For instance, violation of a geometric constraint may result in a model that is impossible to mesh and hence constitute a point in the decision space where the objective function cannot be calculated. Therefore, an optimization approach that doesn't require further calculations at such points is not only useful but also necessary in some cases. This is especially true for automated multi-disciplinary optimization scenarios where requiring further calculations for infeasible points would also mean waste of computational resources as well as errors arising from violation of physical limits. Most algorithms implemented in multi-disciplinary optimization tools [129] mark these points as erroneous and loose information that would otherwise be useful.

As design requirements and systems are evolving in complexity, design process is becoming more dependent on computer tools. One aspect of the computer aided design process is design optimization. While such processes are becoming more dependent on optimization tools, increasing complexity of designs, growing interdependencies, and advancements in technologies set more demanding requirements to optimization algorithms. Moreover, different types of algorithms may be required for different applications depending on the problem characterization. Even with the advancements in computer technology and parallelization, ever-increasing computation needs lead to long computational times for complicated analysis [130]. Consequently, suitable choice of algorithm(s) is crucial to avoid waste of computational resources. In some cases the algorithm choice may even require reformulation of the problem [131]. Availability of several techniques with different capabilities also allows hybridization or serialization of different optimization algorithms to deal with varying complications of advanced problems.

Many prominent evolutionary algorithms require calculation of all functions (including objective functions) at all positions. For instance algorithms that depend on penalty functions for constraint handling require calculation of objective functions at infeasible points. Likewise, many approaches that handle constraints as new objectives of a multi-objective technique behave the same way. There are methods such as Deb's approach [58] that can be utilized to overcome this obstacle. However, special techniques to support these methods are still required to handle highly constrained problems.

Both analytical and numerical methods have widely been applied to engineering design problems. Most optimization problems of engineering and scientific fields are nonlinear and constrained. Increasing complexity of design problems have also given a rise to studies that treat them as multi-objective optimization problems. Functions (objective- and constraint functions) constituting these problems are rarely smooth. In addition, these functions are many times discrete. Although classical optimization methods perform well in specific types of problems, they are generally ineffective when apllied to these intricate problems. There is a vast amout of research on meta-heuristic algorithms, especially evolutionary algorithms, to deal with such problems. An overwhelming majority of these algorithms are inspired from physical or natural phenomena. A brief review of some notable examples of such algorithms can be found in Section 2.3. Among these nature inspired algorithms, PSO has attracted wide interest due to its effectiveness in solving complicated optimization problems, versatility, and ease of implementation.

Adaptability and performance of PSO have led to development of many variants to satisfy different needs related with complicated scientific and engineering optimization problems. Many of these variants introduce modification of acceleration coefficients, addition of new operators, or hybridization to enhance diversity, prohibit early converance, and increase convergence performance [132]. Although some generalized approaches for highly constrained problems in the field of evolutionary algorithms exist [32, 133], there are not many studies related with specialized techniques for particle swarm optimization [60, 134–138].

A careful examination of the literature reveals that there is a lack of a general approach, especially in the field of swarm intelligence, that specializes in highly constrained problems where feasible region of the search space is very small. In fact, in highly constrained problems, the process of finding a feasible solution may become a challenge of a similar magnitude as finding the optimum solution itself. Especially, methods that require the initial swarm to be fully- or partially feasible are rendered useless when dealing with highly constrained problems, mainly because it is very hard to find a single feasible point, let alone initialize the whole population in the feasible region.

Continuous research is being conducted in the field of nature-inspired heuristic optimization algorithms and particle swarm optimization. As stated by Rao, Savsani, and Vakharia [139], “Behavior of nature is always optimum in its performance”.

2.6 Closure

Theoretical background of the study is laid out in this chapter. Definition of optimization and relevant types of problems are described. Classical- and modern optimization methods are described. Some popular techniques that fall into these categories are briefly explained. Evolutionary algorithms, especially the particle swarm optimization is explained with basic algorithm descriptions. Additionally, a brief summary of widely used problems of mechanical engineering domain is also given. Some characteristics of mechanical design problems are discussed.

Finally, a discussion of research opportunity is given with a summary of the field’s current state of the art, shortcomings, and needs.

CHAPTER 3

FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION

As mentioned previously, almost all optimization problems involving mechanical system design are constrained and it is not uncommon for the feasible portion of the search space to be much smaller. This type of problems are often categorized as highly constrained problems.

A thorough investigation of the current literature on constraint handling methods for evolutionary algorithms and PSO indicates that there is a lack of an evolutionary programming framework (especially on swarm-intelligence) that are specialized for use in highly-constrained problems aiming to improve constraint satisfaction level of whole population. Such an algorithm would not only allow finding feasible design points in highly-constrained problems, but also constitute a preprocessing methodology for initializing a population that has a better level of constraint satisfaction for other optimization algorithms that could benefit from a population with less constraint violations. This would also provide an alternative workflow for cases where objective functions are too costly to evaluate in lieu of some constraint violations.

The motivation of developing *feasibility enhanced particle swarm optimization (FEPSO)* technique was to develop a general methodology for highly-constrained optimization problems where all or some of the constraints are relatively easier to evaluate. Furthermore, another goal for the methodology was it to be able to improve constraint satisfaction level of the population as a whole, which actually can be considered as the progressive improvement of constraint satisfaction and requires utilization of constraint handling techniques in both individual (e.g. particle) and population levels. FEPSO was initially developed as a constrained single-objective optimization algorithm.

3.1 General description of FEPSO

FEPSO is an evolutionary optimization algorithm designed to handle highly constrained problems. FEPSO is based on the particle swarm optimization [45–47] but it employs special constraint handling techniques. Infeasible particles in FEPSO do not need to evaluate objective functions and fly only based on social attraction depending on a single violated constraint, called the activated constraint, which is selected at each iteration based on constraint priorities and flight occurs only along dimensions of the search space to which the activated constraint is sensitive. To ensure progressive improvement of constraint satisfaction, particles are not allowed to violate a satisfied constraint via the utilization of a special approach called the virtual boundary search.

FEPSO deals with constrained single-objective optimization problems that can be defined as to *minimize*

$$y = f(\mathbf{x}) \tag{3.1}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq M} \tag{3.2}$$

As seen in the schematic representation of the algorithm given in Figure 3.1, the behavior of the algorithm can be revealed by explaining five separate aspects:

1. Initialization
2. Velocity update rules for infeasible particles (infeasible particle behavior)
3. Velocity update rules for feasible particles (feasible particle behavior)
4. Particle flight
    (a) Enforcing decision variable limits
    (b) Virtual boundary search
5. Post-flight operations

Remaining sections of this chapter discuss each of these aspects separately.

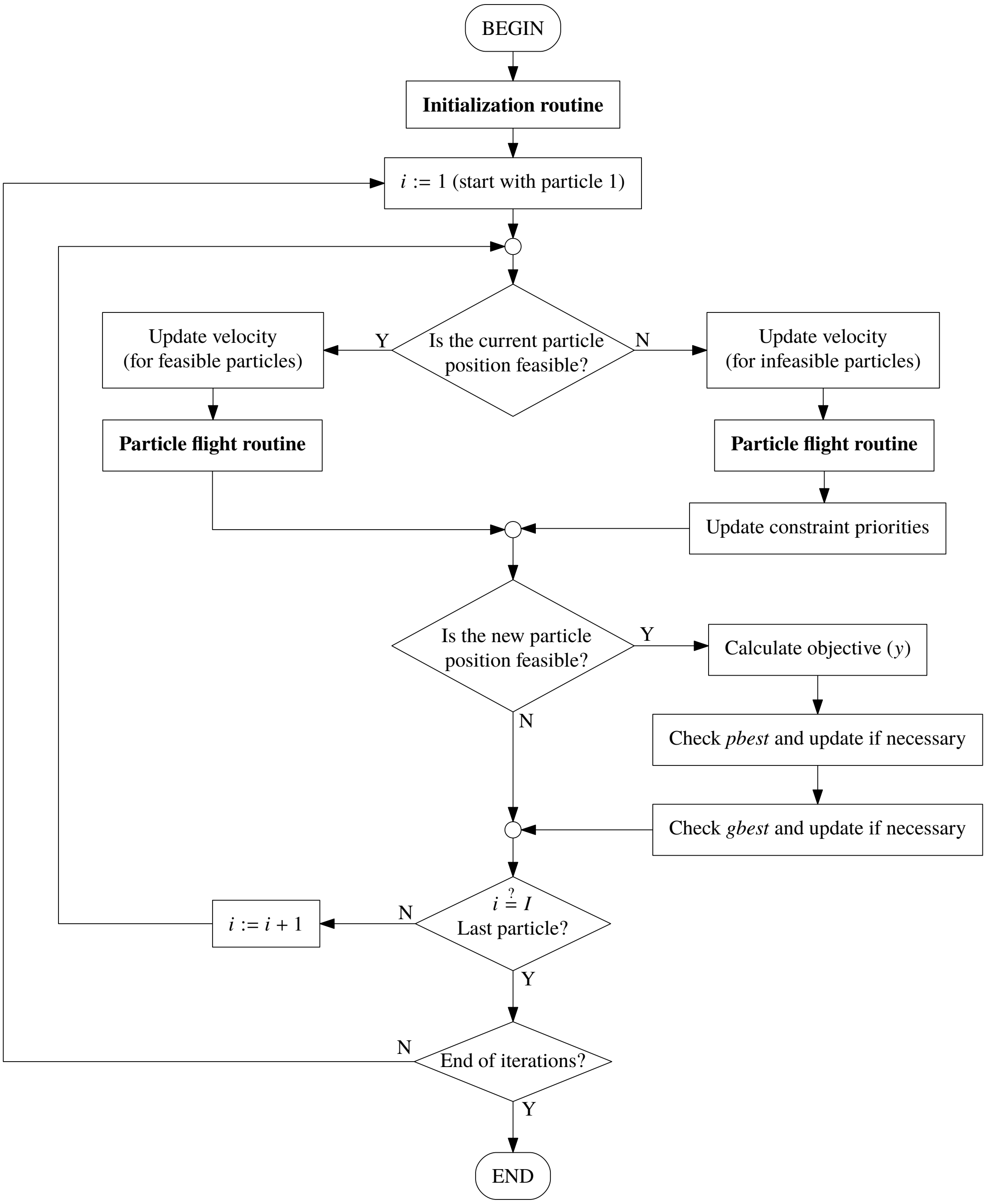

BEGIN
Initialization routine
i := 1 (start with particle 1)
Is the current particle position feasible?
Y
N
Update velocity (for feasible particles)
Update velocity (for infeasible particles)
Particle flight routine
Particle flight routine
Update constraint priorities
Is the new particle position feasible?
Y
N
Calculate objective (y)
Check pbest and update if necessary
Check gbest and update if necessary
i ≟ I
Last particle?
N
Y
i := i + 1
End of iterations?
N
Y
END

Figure 3.1: Flowchart of the main FEPSO algorithm

3.2 Initialization

The operations required to be performed before main MOFEPSO iterations start are labeled as the "Initialization routine" in Figure 3.1. Steps of the initialization routine are summarized in the flowchart shown in Figure 3.2.

Initialization of FEPSO starts with random initialization of the position matrix

$$\mathbf{X} = [x_{i,n}] \in \mathbb{R}^{I \times N} \tag{3.3}$$

where $i \in \mathbb{N}_{>0}^{\leq I}$ and $n \in \mathbb{N}_{>0}^{\leq N}$; I is the number of particles in the swarm; N is the number of decision variables. Several different approaches exist for random initialization of positions. The simplest would be to randomly position each particle by using a uniformly-distributed random number $r \in [0, 1] \subset \mathbb{R}$ such that

$$x_{i,n} = x_n^L + r\left(x_n^U - x_n^L\right) \tag{3.4}$$

However, in the implementation of FEPSO, *Latin Hypercube Sampling Method* is utilized to take advantage of its simultaneous stratification property on all input dimensions [140]. Therefore, the technique ensures that the initial population is well spread throughout all dimensions.

Other than the position matrix; the velocity matrix

$$\mathbf{V} = [v_{i,n}] \in \mathbb{R}^{I \times N} \tag{3.5}$$

and the constraint matrix

$$\mathbf{C} = [c_{i,m}] \in \mathbb{R}^{I \times M} \tag{3.6}$$

must also be initialized. Here, M is the number of constraints. As shown in Figure 3.2, initialization of particle's velocity, the constraints, *pbest*, and *gbest* are performed within a loop for each particle. The velocity of each particle is initialized with

$$v_{i,n} := a^v\left(2r - 1\right)\left(x_n^U - x_n^L\right) \tag{3.7}$$

where r is once again a uniformly-distributed random number and a^v is the velocity initialization factor[1]. Constraint values for a particle is calculated and recorded in the

---

[1] The velocity initialization factor ensures that the velocity vectors' components never exceed a fraction of the corresponding search space dimension's size. $a^v = 0.3$ is used in all implementations of FEPSO.

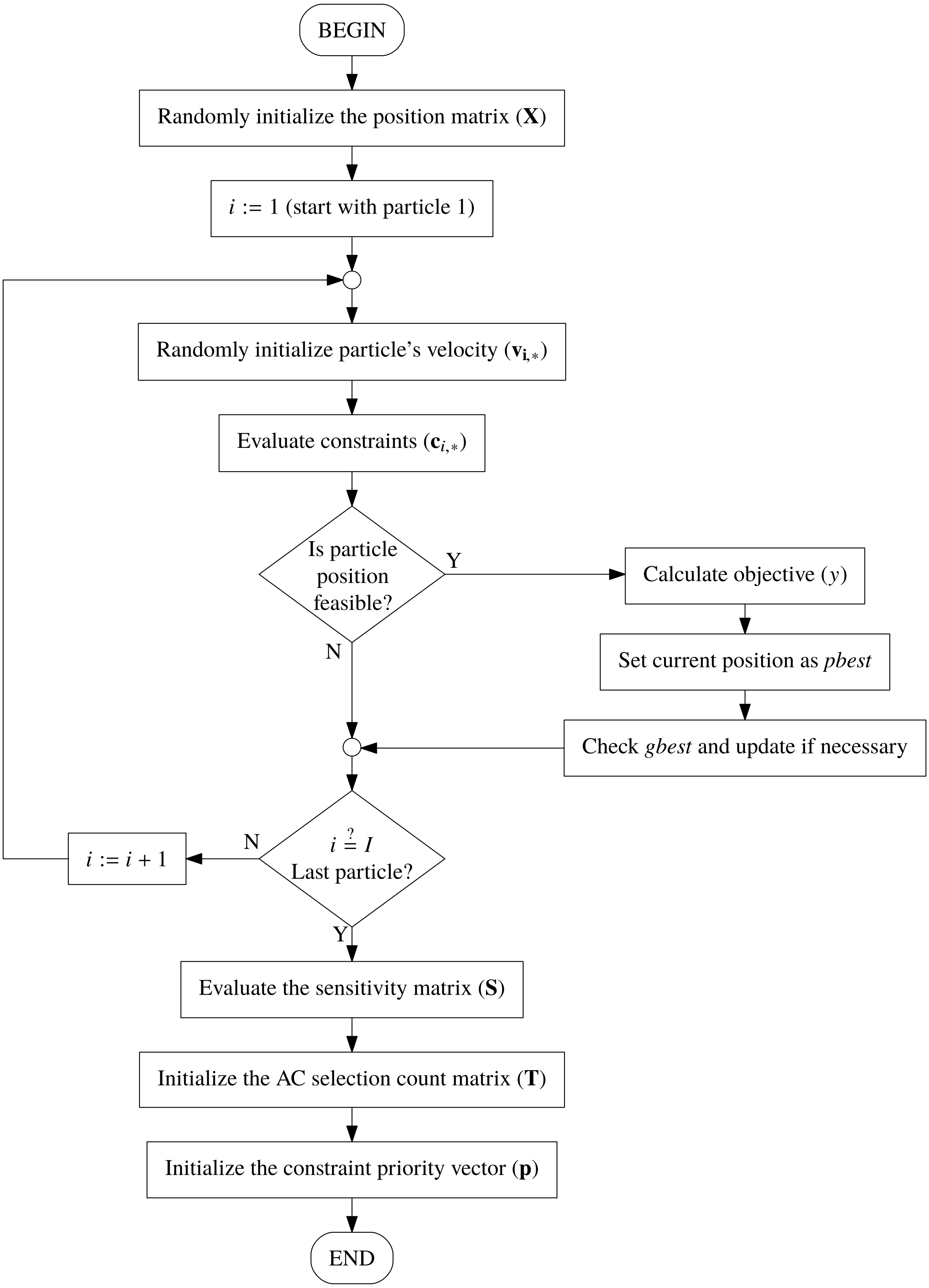

BEGIN
Randomly initialize the position matrix (X)
i := 1 (start with particle 1)
Randomly initialize particle's velocity (v_{i,*})
Evaluate constraints (c_{i,*})
Is particle position feasible?
Y
N
Calculate objective (y)
Set current position as pbest
Check gbest and update if necessary
i ?= I
Last particle?
N
Y
i := i + 1
Evaluate the sensitivity matrix (S)
Initialize the AC selection count matrix (T)
Initialize the constraint priority vector (p)
END

Figure 3.2: Flowchart of the FEPSO initialization routine

corresponding row of the constraint matrix $\mathbf{C}$ such that[2]

$$\mathbf{c}_{i,*} := \begin{bmatrix} g_1(\mathbf{x}_{i,*}^T) & g_2(\mathbf{x}_{i,*}^T) & \cdots & g_M(\mathbf{x}_{i,*}^T) \end{bmatrix} \tag{3.8}$$

FEPSO keeps track of the *gbest* as a tuple of the best position and its corresponding objective value:

$$\mathbf{b} = (\mathbf{b}^x, b^y) \tag{3.9}$$

Initially, it is assumed that $b^y = \infty$ which is the worst possible objective value. Similarly, each particle has its own tuple of *pbest*:

$$\mathbf{d_i} = (\mathbf{d}_i^x, d_i^y) \tag{3.10}$$

During the initialization of a particle[3], after the constraints are calculated, if the particle position is found to be feasible, the objective function is evaluated and *pbest* is set to the current position. That is if

$$\left(\forall m \in \mathbb{N}_{>0}^{\leq M}\right)(c_{i,m} \leq 0) \tag{3.11}$$

then

$$\mathbf{d}_i^x = \mathbf{x}_{i,*}^T \tag{3.12}$$

$$d_i^y = f(\mathbf{x}_{i,*}^T) \tag{3.13}$$

Furthermore, if the recently calculated objective is better than *gbest*; that is if

$$d_i^y < b^y \tag{3.14}$$

then *gbest* is replaced with the new solution such that

$$\mathbf{b}^x = \mathbf{d}_i^x \tag{3.15}$$

$$b^y = d_i^y \tag{3.16}$$

Note that objective function is only evaluated if the particle position is feasible. FEPSO does not calculate the objective functions of infeasible positions by design. PSO implementations [65, 141] and other evolutionary algorithms utilizing feasibility-based rules

---

[2] Note that $\mathbf{x}_{i,*}$ is the row vector representing the position of the particle and $\mathbf{x}_{i,*}^T$ is its transpose. Therefore $g_1(\mathbf{x}_{i,*}^T)$ is the value of first constraint function calculated at the ith particle's position.

[3] The particle of interest is assumed to be the ith particle.

such as Deb's approach [58] commonly refrain from calculating objective functions in infeasible regions of the design space. However, FEPSO behaves somewhat differently. Although the details of how particle flight occurs in FEPSO will be explained in Section 3.5, it is worth mentioning here that FEPSO does not allow particles to fly into infeasible regions if they are already feasible. It also has a multidimensional feasibility approach, such that even if the particle was in an infeasible region it is not allowed to fly into a region where its already-satisfied constraints become violated. Infeasible particles in FEPSO do not try to optimize their objectives. Instead, they try to become feasible. This approach, in turn, leads to a minimum number of objective function evaluations. However, if objective evaluations are not too costly, this behaviour can be modified accordingly.

FEPSO does not expect the initial population to contain feasible particles but requires the population to have at least one non-violating particle for each constraint.

Remaining steps of initialization are not particle level operations. Therefore, they are not performed within the particle loop (see Figure 3.2). These steps involve assignment of initial values to some swarm level variables required by the FEPSO algorithm.

Flight of infeasible particles in FEPSO are influenced by an *activated constraint* (AC) selected at each iteration for every particle. Selection of the AC is discussed in section 3.3.1. Infeasible particles in FEPSO only fly along dimensions to which the AC is sensitive. Therefore, the matrix

$$\mathbf{S} = [s_{m,n}] \in \mathbb{B}^{M \times N} \tag{3.17}$$

is initialized such that

$$s_{m,n} = \begin{cases} 1, & \text{constraint m is sensitive to variable n} \\ 0, & otherwise \end{cases} \tag{3.18}$$

Note that $\mathbb{B}$ is the Boolean set.

Constraint sensitivity calculation is similar to OCCURM (occurrence matrix calculator) in constraint decomposition technique of Khorshid and Seireg [96]. However, instead of performing only one evaluation for each- and every direction in the search space, FEPSO temporarily moves the initialized particles in one principal direction at a time for a predefined amount (i.e. some percentage of the design variable range $x_n^U - x_n^L$)

so as to be able to catch sensitivities that only exist in certain regions of the search domain. Once the sensitivities are acquired in the initialization phase, the $\mathbf{S}$ matrix is fixed.

Selection of AC depends on the constraint priorities and the number of selection recurrences for each constraint. Although these values are updated after each particle flight, they have to be initialized properly. Therefore, the AC selection count matrix

$$\mathbf{T} = [t_{i,m}] \in \mathbb{N}^{I \times M} \tag{3.19}$$

is initialized as a zero matrix (since no constraint has been yet selected as the AC for any particle) and the constraint priority vector

$$\mathbf{p} = [p_1, p_2, \ldots, p_M]^T \in \mathbb{N}^{M \times 1} \tag{3.20}$$

is calculated simply by counting the number of particles that violate each constraint. The constraint violated by the maximum number of particles gets the highest priority.

$$p_m = \sum_{i=1}^{I} z_{i,m}, \qquad z_{i,m} = \begin{cases} 1, & c_{i,m} > 0 \\ 0, & otherwise \end{cases} \tag{3.21}$$

where p_m is the priority of the mth constraint. Note that $c_{i,m} > 0$ implies that particular constraint has been violated.

3.3 Velocity update rules for infeasible particles

Feasible- and infeasible particles behave differently in FEPSO. This difference in behaviors is due to dissimilar velocity update rules. This section describes operations performed for infeasible particle velocity updating and the next section (Section 3.4) focuses on feasible particles.

3.3.1 Selection of the AC and the global guide

The flight of infeasible particles is based on the AC selected for each particle at the beginning of each iteration. The AC chosen depends on the current constraint priorities and the number of selection recurrences of the constraints as AC for each- and every particle. The particle chooses the highest priority constraint among the constraints that have been least chosen as the AC. Therefore, AC (as denoted as a) is selected for the ith particle such that

$$a \in \mathbf{R} \quad \wedge \quad p_a = \max_{m \in \mathbf{R}} (p_m) \tag{3.22}$$

where

$$\mathbf{R} = \left\{ m \in \mathbb{N} : t_{i,m} = \min_{r \in \mathbf{R}^*} (t_{i,r}) \right\} \tag{3.23}$$

$$\mathbf{R}^* = \left\{ r \in \mathbb{N}_{>0}^{\leq M} : c_{i,r} > 0 \right\} \tag{3.24}$$

Here, $\mathbf{R}^*$ is the set of constraints that the ith particle violates; $\mathbf{R}$ denotes the set of constraints that are visited the least among $\mathbf{R}^*$. After the selection of the AC, $\mathbf{T}$ is updated as

$$t_{i,a} := t_{i,a} + 1 \tag{3.25}$$

This selection scheme ensures that no high priority constraint is overemphasized and that the diversity of the swarm is preserved.

After selection of the AC, a global guide can be chosen for the infeasible particle to lead its flight. Notice that infeasible particles in FEPSO only serve for social stimulation (i.e. they have no cognitive impulse). The guide is randomly selected from the set of particles that do not violate the AC as

$$\mathbf{J} = \{ j \in \mathbb{N}_{>0} : c_{j,a} \leq 0 \} \tag{3.26}$$

Therefore, the guide position $\mathbf{x}^G$ can be selected randomly from the set of current positions of particles in $\mathbf{J}$:

$$\mathbf{x}^G = \mathbf{x}_{j,*} \tag{3.27}$$

where j is an arbitrarily selected element in $\mathbf{J}$.

3.3.1.1 The velocity update equation for infeasible particles

Presuming that the ith particle offers an infeasible solution, its velocity can be updated by

$$v_{i,n} := s_{a,n} \left[W v_{i,n} + C_1 r_{1,n} \left(x_n^G - x_{i,n} \right) \right] \tag{3.28}$$

where W is the inertia factor; C_1 is the social acceleration coefficient; $r_{1,n} \in [0,1] \subset \mathbb{R}$ are uniformly-distributed random numbers. The Boolean sensitivity term $s_{a,n}$ sets the velocity term $v_{i,n}$ to zero if the AC (a) is not sensitive to the decision variable n, hence blocking motion along dimensions in which the AC is insensitive. After the velocity is updated, particle flight occurs as elaborated in Section 3.5. Particle flight routine is common for feasible- and infeasible particles. After infeasible particle flight, constraint priorities are updated using Eqn. 3.21.

3.4 Velocity update rules for feasible particles

Feasible particles in FEPSO are socially attracted to *gbest* $(\mathbf{b})$ and cognitively attracted to their own *pbest* $(\mathbf{d}_i)$. To be specific, a feasible particle's velocity is updated by

$$v_{i,n} := W v_{i,n} + C_1 r_{1,n} \left(b_n^x - x_{i,n}\right) + C_2 r_{2,n} \left(d_{i,n}^x - x_{i,n}\right) \tag{3.29}$$

where C_2 is the cognitive acceleration coefficient while $r_{1,n}, r_{2,n} \in [0,1] \subset \mathbb{R}$ are uniformly-distributed random numbers.

3.5 Particle flight

Particle flight in FEPSO occurs in such a way that if a particle does not violate a constraint at the beginning of its flight, it cannot violate it at the end of its flight. The flight of particles are truncated at the virtual boundary defined in search space where any one of the constraints previously satisfied become violated. This approach is similar to TRC [67] except for the final velocities of the particles colliding with a virtual boundary. FEPSO utilizes an approach called "stick", where velocities of particles vanish at the virtual boundary. A similar approach called the "absorbing walls" exists for particles violating the solution space boundaries [142]. However, TRC, "stick", and "absorbing walls" are defined for decision variable limits (i.e. linear constraints that depend on a single decision variable) whereas FEPSO can apply these principles to any kind of constraint with differing complexities through *virtual boundary search* explained in Section 3.5.2.

Particle flight shown in Figure 3.3 starts with creation of a candidate position $\mathbf{x}^* \in \mathbb{R}^{N\times 1}$. The elements of $\mathbf{x}^*$ are calculated as follows:

$$x_n^* := x_{i,n} + v_{i,n} \tag{3.30}$$

Note that $\mathbf{x}^*$ is only a candidate position since it has to satisfy two criteria before it can become the new particle position of interest:

1. $\mathbf{x}^*$ should not violate any decision variable limits:

   $$(\nexists n)\left(x_n^* < x_n^L \quad \vee \quad x_n^* > x_n^U\right) \tag{3.31}$$

2. $\mathbf{x}^*$ should not violate constraints that the current position of the particle of interest[4] satisfied previously:

   $$(\nexists m)\ \left(c_{i,m} \leq 0 \wedge c_m^* > 0\right) \tag{3.32}$$

   where $\mathbf{c}^*$ is the constraint vector corresponding to $\mathbf{x}^*$ such that

   $$\mathbf{c}^* := \begin{bmatrix} g_1(\mathbf{x}^*) & g_2(\mathbf{x}^*) & \cdots & g_M(\mathbf{x}^*) \end{bmatrix}^T \tag{3.33}$$

FEPSO checks the first criterion above. If satisfied, Eqn. 3.33 is evaluated to check the second criterion. However, in case the first criterion is not satisfied, the decision variable limits are enforced (as explained in Section 3.5.1) before Eqn. 3.33 is evaluated. Similarly, if the second criterion is not satisfied, the candidate position is modified until it satisfies the conditions described in Section 3.5.2. Note that the second criterion ensures that $\mathbf{x}^*$ does not violate any previously satisfied constraints. Therefore, if a particle is in a feasible position, it cannot move to an infeasible position during its flight. After both criteria are satisfied by $\mathbf{x}^*$, it is assigned as the new position of the particle and post-flight operations (explained in Section 3.6) are performed.

3.5.1 Enforcing decision variable limits

If the candidate position $\mathbf{x}^*$ violates any variable limits, a limit violation rate (w) is computed as shown in Figure 3.4 to constrain the particle within the decision space

---

[4] Note that particle of interest is denoted as ith particle

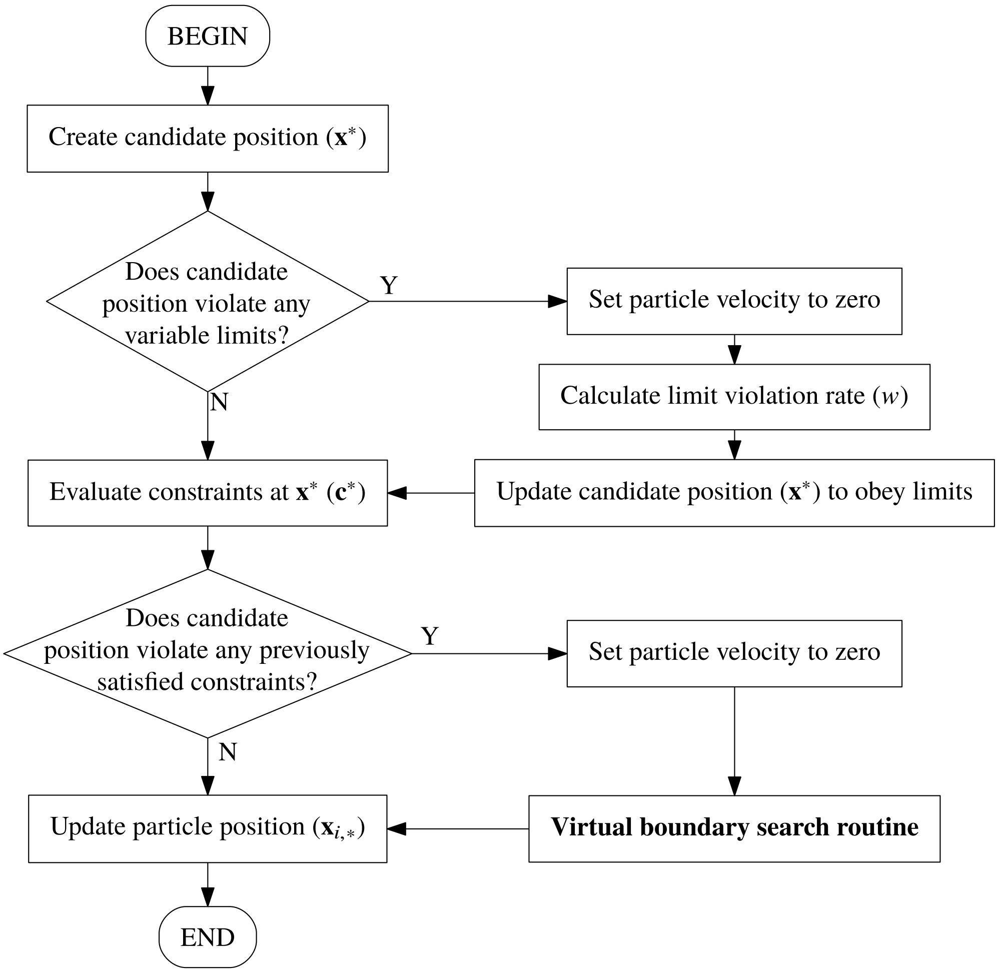

BEGIN
Create candidate position ($\mathbf{x}^*$)
Does candidate position violate any variable limits?
Y
N
Set particle velocity to zero
Calculate limit violation rate (w)
Update candidate position ($\mathbf{x}^*$) to obey limits
Evaluate constraints at $\mathbf{x}^*$ ($\mathbf{c}^*$)
Does candidate position violate any previously satisfied constraints?
Y
N
Set particle velocity to zero
Virtual boundary search routine
Update particle position ($\mathbf{x}_{i,*}$)
END

Figure 3.3: Flowchart of the FEPSO particle flight routine

bounds.

$$w^L = \max_n \left(\frac{x_n^L - x_n^*}{x_{i,n} - x_n^L} \right) \tag{3.34}$$

$$w^U = \max_n \left(\frac{x_n^* - x_n^U}{x_n^U - x_{i,n}} \right) \tag{3.35}$$

$$w = \max \left(w^L, w^U \right) \tag{3.36}$$

The candidate position is updated using the limit violation rate such that

$$x_n^* := x_{i,n} + \frac{x_n^* - x_{i,n}}{1 + w} \tag{3.37}$$

Likewise, the velocity of the particle is set to zero: $v_{i,n} := 0$.

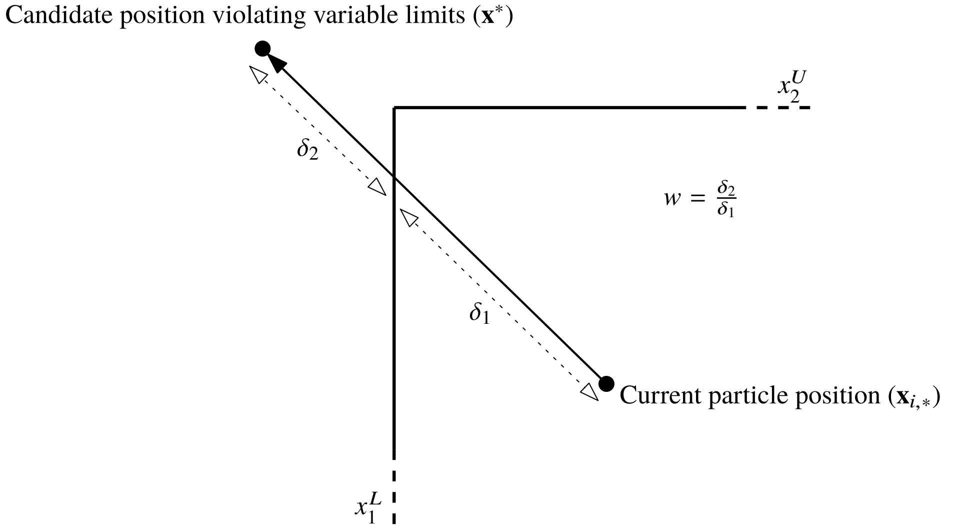

Candidate position violating variable limits ($\mathbf{x}^*$)
x_2^U
δ_2
$w = \frac{\delta_2}{\delta_1}$
δ_1
Current particle position ($\mathbf{x}_{i,*}$)
x_1^L

Figure 3.4: Illustration of limit violation rate in two dimensional decision space

3.5.2 Virtual boundary search

Virtual boundary search (VBS), as described in Figure 3.5, seeks the virtual boundary where any one of the previously satisfied constraints become violated. VBS is performed for every particle flight that occurs towards a region where any of the previously satisfied constraints become violated. VBS pulls the particle back to the boundary where such a violation occurs.

First step of the VBS is the assignment of the virtually feasible- ($\mathbf{x}^+$) and the virtually infeasible ($\mathbf{x}^-$) edges.

$$\mathbf{x}^+ := \mathbf{x}_i \qquad\qquad \mathbf{c}^+ := \mathbf{c}_i \tag{3.38}$$

$$\mathbf{x}^- := \mathbf{x}^* \qquad\qquad \mathbf{c}^- := \mathbf{c}^* \tag{3.39}$$

Then, the infeasibility rate is calculated as follows:

$$q = \max_{m \in \mathbf{Y}} \left(\frac{c_m^-}{-c_m^+} \right) \tag{3.40}$$

where $\mathbf{Y}$ is the set of constraints that are satisfied by the current position of the particle as

$$\mathbf{Y} = \left\{ m \in \mathbb{N}_{>0}^{\leq M} : c_{i,m} \leq 0 \right\} \tag{3.41}$$

Infeasibility rate is used to create a new trial position:

$$\mathbf{x}^\tau := \mathbf{x}^+ + \frac{\mathbf{x}^- - \mathbf{x}^+}{1+q} \tag{3.42}$$

The corresponding constraint vector $\mathbf{c}^\tau$ is also calculated with

$$\mathbf{c}^\tau := \begin{bmatrix} g_1(\mathbf{x}^\tau) & g_2(\mathbf{x}^\tau) & \cdots & g_M(\mathbf{x}^\tau) \end{bmatrix}^T \tag{3.43}$$

If the trial position does not violate the constraints satisfied by the current position of the particle, it said to be virtually feasible as defined in Eqn. 3.32. The virtual feasibility proposition can be rewritten for $\mathbf{x}^\tau$ as follows:

$$(\nexists m) \;\; (c_{i,m} \leq 0 \wedge c_m^\tau > 0) \tag{3.44}$$

Therefore, if the proposition in Eqn. 3.44 is true, $\mathbf{x}^\tau$ becomes virtually feasible and it replaces $\mathbf{x}^+$:

$$\mathbf{x}^+ := \mathbf{x}^\tau \qquad\qquad \mathbf{c}^+ := \mathbf{c}^\tau \tag{3.45}$$

Otherwise, it replaces $\mathbf{x}^-$:

$$\mathbf{x}^- := \mathbf{x}^\tau \qquad\qquad \mathbf{c}^- := \mathbf{c}^\tau \tag{3.46}$$

Iterations continue by recalculating the infeasibility rate in Eqn. 3.40 until the following VBS termination criterion is attained:

$$(\forall n) \left(\frac{|x_n^\tau - x_n^e|}{x_n^U - x_n^L} \leq x^{tol} \right) \tag{3.47}$$

where x^{tol} is the tolerance; $\mathbf{x}^e$ denotes the edge closer to $\mathbf{x}^\tau$ such that

$$\mathbf{x}^e := \begin{cases} \mathbf{x}^+, & q \geq 1 \\ \mathbf{x}^-, & q < 1 \end{cases} \tag{3.48}$$

Finally, the candidate position is updated as the virtually feasible edge of the VBS such that

$$\mathbf{x}^* := \mathbf{x}^+ \qquad\qquad \mathbf{c}^* := \mathbf{c}^+ \tag{3.49}$$

3.6 Post-flight operations

The updated candidate position that satisfies both criteria given by Eqns. 3.31 and 3.32 is assigned as the new position of the particle such that

$$\mathbf{x}_i := \mathbf{x}^* \qquad\qquad \mathbf{c}_i := \mathbf{x}^* \tag{3.50}$$

Some post-flight operations are conducted depending on pre-flight- and post-flight feasibilities of the particle. If the particle's previous position was infeasible, constraint priorities are updated using Eqn. 3.21. Furthermore, if the new position is found to be feasible, the objective ($y^* := f(\mathbf{x}^*)$) is calculated. If $y^* < d_i^y$ then *pbest* is updated such that

$$\mathbf{d}_i^x := \mathbf{x}^* \qquad\qquad d_i^y := y^* \tag{3.51}$$

Similarly, if $y^* < b^y$ then

$$\mathbf{b}^x := \mathbf{x}^* \qquad\qquad b^y := y^* \tag{3.52}$$

3.7 Handling discrete decision variables

FEPSO treats all decision variables as real variables. However, if any of these variables can only take some discrete values then they can be transformed into the nearest allowable discrete value (see the transformation approach explained in Section 2.3.6.3) before they are used in constraint- and objective functions. Hence, for this type of decision variables, FEPSO only requires a valid decision variable range for the real valued image of the discrete variable and a one-way transformation function to convert the real value to the corresponding discrete value.

3.8 Preliminary tests on benchmark problems

FEPSO was applied to several benchmark problems involving mechanical design optimization. Although problem definitions are presented by Rao and Savsani [86], they are repeated here for the sake of completeness. A compilation of solutions to these problems found by different methods are given by Sadollah et al. [143]. These results are also shown in comparison with results found with FEPSO.

In all FEPSO runs performed, swarm size was set to 50. The inertia factor W was linearly varied from 0.9 to 0.4. The social- and cognitive acceleration factors were taken as 2 ($C_1 = C_2 = 2$). Simulations were repeated 25 times.

3.8.1 Pressure vessel design problem

The pressure vessel design problem's goal is to minimize the cost of a cylindrical pressure vessel with hemispherical heads illustrated in Figure 3.6. The decision vector is expressed as

$$\mathbf{x} = [T_s \quad T_h \quad R \quad L]^T \tag{3.53}$$

where T_s, T_h, R, and L are shell thickness, head thickness, inner radius, and cylindrical section length of the pressure vessel respectively. Although the original definition of the problem limits values of T_s and T_h to integer multiples of 0.0625, many studies dealing with the pressure vessel design problem ignore this limitation [143, 144]. Therefore, to be able to produce results comparable with the existing results in the literature, both cases are included. For the unlimited case it is assumed that $T_s, T_h \in \mathbb{R}_{\geq 0.1}^{\leq 99}$ whereas for the limited case $T_s, T_h \in \left\{T = 0.0625t : t \in \mathbb{N}_{\geq 2}^{\leq 1584}\right\}$. For both cases $R, L \in \mathbb{R}_{\geq 10}^{\leq 100}$. The problem can be defined as to *minimize*

$$f(\mathbf{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{3.54}$$

subject to

$$g_1(\mathbf{x}) = -x_1 + 0.0193x_3 \leq 0 \tag{3.55}$$

$$g_2(\mathbf{x}) = -x_2 + 0.00954x_3 \leq 0 \tag{3.56}$$

$$g_3(\mathbf{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1\,296\,000 \leq 0 \tag{3.57}$$

$$g_4(\mathbf{x}) = x_4 - 240 \leq 0 \tag{3.58}$$

Table 3.1: Summary of results for the pressure vessel design problem

Algorithm	CFE	OFE	Best	Mean	Worst
GA3	900 000	900 000	6288.7445	6293.843	6308.497
GA4	80 000	80 000	6059.9463	6177.253	6469.322
CPSO	240 000	240 000	6061.0777	6147.133	6363.8041
HPSO	81 000	81 000	6059.7143	6099.932	6288.677
NM-PSO	80 000	80 000	5930.3137	5946.790	5960.0557
G-QPSO	8000	8000	6059.7208	6440.379	7544.4925
QPSO	8000	8000	6059.7209	6440.379	8017.2816
PSO	8000	8000	6693.7212	8756.680	14 076.324
CDE	204 800	204 800	6059.734	6085.230	6371.0455
UPSO	100 000	100 000	6544.270	9032.550	-
PSO-DE	42 100	42 100	6059.714	6059.714	-
ABC	30 000	30 000	6059.714	6245.308	-
$(\mu + \lambda)$-ES	30 000	30 000	6059.7016	6379.938	-
TLBO	10 000	10 000	6059.7143	6059.714	-
MBA	70 650	70 650	5889.3216	6200.648	6392.5062
MOGA2	25 000	25 000	6059.723	6127.401	6502.751
NSGA2	25 000	25 000	6059.734	6201.523	6718.511
FEPSO (limited)	9834	5171	6059.714	6457.607	7544.493
FEPSO	29 652	25 036	5885.3328	6242.807	6719.8258

3.8.2 Tension/compression spring design problem

The tension/compression spring design problem aims to minimize the weight of a spring while satisfying constraints imposed on minimum deflection, shear stress, and outside diameter (see Figure 3.7). The problem can be defined as to *minimize*

$$f(\mathbf{x}) = (N+2)Dd^2 \tag{3.59}$$

subject to

$$g_1(\mathbf{x}) = 1 - \frac{D^3 N}{71\,785 d^4} \leq 0 \tag{3.60}$$

$$g_2(\mathbf{x}) = \frac{4D^2 - dD}{12\,566(Dd^3 - d^4)} + \frac{1}{5108 d^2} - 1 \leq 0 \tag{3.61}$$

$$g_3(\mathbf{x}) = 1 - \frac{140.45 d}{D^2 N} \leq 0 \tag{3.62}$$

$$g_4(\mathbf{x}) = \frac{D+d}{1.5} - 1 \leq 0 \tag{3.63}$$

where

$$\mathbf{x} = [d \quad D \quad N]^T \tag{3.64}$$

while $d \in \mathbb{R}_{\geq 0.05}^{\leq 2}$, $D \in \mathbb{R}_{\geq 0.25}^{\leq 1.3}$, and $N \in \mathbb{N}_{\geq 2}^{\leq 15}$ are the wire diameter, the mean coil diameter, and the number of active coils respectively.

3.8.3 Welded beam design problem

Objective of the welded beam design optimization problem (see Figure 3.8) is to minimize the cost. Height of weld (h), length of weld (L), height of beam (t), and width of beam (b) are the design variables; hence the decision vector is written as follows:

$$\mathbf{x} = [h \quad L \quad t \quad b]^T \tag{3.65}$$

where $h, b \in \mathbb{R}_{\geq 0.1}^{\leq 2}$ and $L, t \in \mathbb{R}_{\geq 0.1}^{\leq 10}$. Problem constraints involve shear stress (τ), bending stress (σ), buckling load (P_c), and deflection of the beam (δ). Formally, the problem is stated as to *minimize*

$$f(\mathbf{x}) = 1.104\,71 x_1^2 x_2 + 0.048\,11 x_3 x_4 (14 + x_2) \tag{3.66}$$

subject to

$$g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{max} \leq 0 \tag{3.67}$$

Table 3.2: Summary of results for the tension/compression spring design problem

Algorithm	CFE	OFE	Best	Mean	Worst
QPSO	2000	2000	0.012 669	0.013 854	0.018 127
PSO	2000	2000	0.012 857	0.019 555	0.071 802
DE	204 800	204 800	0.012 670	0.012 703	0.012 790
DELC	20 000	20 000	0.012 665	0.012 665	0.012 666
DEDS	24 000	24 000	0.012 665	0.012 669	0.012 738
HEAA	24 000	24 000	0.012 665	0.012 665	0.012 665
PSO-DE	24 950	24 950	0.012 665	0.012 665	0.012 665
SC	25 167	25 167	0.012 669	0.012 923	0.016 717
UPSO	100 000	100 000	0.013 120	0.022 940	-
CDE	240 000	240 000	0.012 670	0.012 703	-
$(\mu + \lambda)$-ES	30 000	30 000	0.012 689	0.013 165	-
ABC	30 000	30 000	0.012 665	0.012 709	-
TLBO	10 000	10 000	0.012 665	0.012 666	-
MBA	7650	7650	0.012 665	0.012 713	0.012 900
MOGA2	25 000	25 000	0.012 665	0.012 799	0.128 54
NSGA2	25 000	25 000	0.012 666	0.012 794	0.128 06
FEPSO	41 381	24 942	0.012 666	0.013 107	0.015 327

$$g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{max} \leq 0 \tag{3.68}$$

$$g_3(\mathbf{x}) = x_1 - x_4 \leq 0 \tag{3.69}$$

$$g_4(\mathbf{x}) = 0.104\,71 x_1^2 + 0.048\,11 x_3 x_4 (14 + x_2) - 5 \leq 0 \tag{3.70}$$

$$g_5(\mathbf{x}) = 0.125 - x_1 \leq 0 \tag{3.71}$$

$$g_6(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{max} \leq 0 \tag{3.72}$$

$$g_7(\mathbf{x}) = P - P_c(\mathbf{x}) \leq 0 \tag{3.73}$$

where

$$\tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{3.74}$$

$$\tau' = \frac{P}{\sqrt{2}x_1 x_2} \tag{3.75}$$

$$\tau'' = \frac{MR}{J} \tag{3.76}$$

$$M = P\left(L + \frac{x_2}{2}\right) \tag{3.77}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{3.78}$$

$$J = 2\left[\sqrt{2}x_1 x_2 \left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)\right] \tag{3.79}$$

$$\sigma(\mathbf{x}) = \frac{6PL}{x_4 x_3^2} \tag{3.80}$$

$$\delta(\mathbf{x}) = \frac{4PL^3}{E x_3^3 x_4} \tag{3.81}$$

$$\delta(\mathbf{x}) = \frac{4PL^3}{E x_3^3 x_4} \tag{3.82}$$

$$P_c(\mathbf{x}) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{3.83}$$

The constant values used in these calculations are as follows: $P = 6000\,\mathrm{lb}$, $L = 14\,\mathrm{in}$, $E = 30 \times 10^6\,\mathrm{psi}$, $G = 12 \times 10^6\,\mathrm{psi}$, $\tau_{max} = 13\,600\,\mathrm{psi}$, $\sigma_{max} = 30\,000\,\mathrm{psi}$, $\delta_{max} = 0.25\,\mathrm{in}$.

Table 3.3: Summary of results for the welded beam design problem

Algorithm	CFE	OFE	Best	Mean	Worst
GA3	900 000	900 000	1.748 309	1.771 973	1.785 835
GA4	80 000	80 000	1.728 226	1.792 654	1.993 408
CAEP	50 020	50 020	1.724 852	1.971 809	3.179 709
CPSO	240 000	240 000	1.728 024	1.748 831	1.782 143
HPSO	81 000	81 000	1.724 852	1.749 040	1.814 295
PSO-DE	66 600	66 600	1.724 852	1.724 852	1.724 852
NM-PSO	80 000	80 000	1.724 717	1.726 373	1.733 393
MGA	-	-	1.824 500	1.919 000	1.995 000
SC	33 095	33 095	2.385 435	3.002 588	6.399 679
DE	204 800	204 800	1.733 461	1.768 158	1.824 105
UPSO	100 000	100 000	1.921 990	2.837 210	-
CDE	240 000	240 000	1.733 460	1.768 150	-
$(\mu + \lambda)$-ES	30 000	30 000	1.724 852	1.777 692	-
ABC	30 000	30 000	1.724 852	1.741 913	-
TLBO	10 000	10 000	1.724 852	1.728 447	-
MBA	47 340	47 340	1.724 853	1.724 853	1.724 853
MOGA2	25 000	25 000	1.724 852	1.728 460	1.882 873
NSGA2	25 000	25 000	1.724 852	1.756 723	1.918 285
FEPSO	30 459	22 549	1.724 852	1.724 852	1.724 852

3.8.4 Discussion of benchmark results

Results of FEPSO runs for each problem are separately given in Tables 3.1, 3.2, and 3.3. Solutions found in the literature are also included in these tables. FEPSO was able to find solutions that are either equivalent to or better than the best solutions found in previous studies. FEPSO achieved to obtain results within reasonable number of constraint function evaluations when compared with other methods. In some cases such as the unlimited version of the tension/compression spring design problem, FEPSO attained the best result found in the surveyed literature with remarkably low number of constraint- and objective function evaluations. These benchmark results show that FEPSO is a competent algorithm for constrained design optimization. Usage of FEPSO in some large scale problems are addressed in the following chapters.

Note that MOGA2 [145] and NSGA2 [58] are actually multi-objective algorithms. However, they can also handle constrained single-objective problems. It can be observed in Tables 3.1, 3.2, and 3.3 that both MOGA2 and NSGA2 are very competitive algorithms. Their good performance and ability to handle both single-objective and multi-objective problems make them good candidates for benchmarking. Both NSGA2 [146–149] and MOGA2 [150–153] are widely used in benchmark comparisons by many studies including ones dealing with single-objective approaches [146, 154].

3.9 Closure

This chapter introduces FEPSO method to solve constrained optimization problems in engineering design. This new approach does not require objective function evaluation at infeasible points. Moreover, it does not require the swarm to have feasible particles when initialized. It is designed to perform satisfactorily when the problem is highly-constrained. FEPSO progressively improves the level of constraint satisfaction in the swarm at each iteration. Apart from classical PSO parameters (W, C_1, C_2), it only requires one additional tolerance parameter (x^{tol})[5]. Several differences of FEPSO when compared to other techniques are listed below:

---

[5] Since the velocity initialization factor (a^v) is just used to limit values of the velocity matrix elements during initialization, it is not taken as a parameter and rarely needs to be changed.

- During the flight of a particle, if a previously satisfied constraint becomes violated, FEPSO seeks the point where the violation occurs (virtual boundary) along the path and stops the particle at that point. Therefore, FEPSO never allows a satisfied constraint to become violated. This provides a progressive improvement in terms of constraint satisfaction level of the swarm.

- Particles that are feasible fly similar to how particles fly in the canonical PSO except for how constraints are handled (as described in the previous item). However, infeasible particles fly in a novel way:

    - Infeasible particles activate a constraint that will drive the social attraction. The AC will be selected based on constraint priorities (purely based on number of particles that violate the constraint). However, activation of constraints are done in a cyclic fashion where in the following iteration the particle selects next highest priority constraint. This prohibits over-emphasizing certain constraints and helps maintaining diversity. Although prioritizing constraints based on number of violations is not a new concept, the rest of the procedure is novel.
    - Infeasible particles fly based on only social attraction driven by the AC by simply being attracted to a particle which does not violate the AC.
    - Flight of infeasible particles do not occur in all dimensions in FEPSO. FEPSO identifies which constraints are sensitive to which decision variables and only allows flight through dimensions to which the AC is sensitive. This feature is designed to achieve a faster convergence to feasible regions while keeping already satisfied constraints intact.

- FEPSO does not require objective function evaluation of infeasible points. Therefore, it can be used in problems where violation of some constraints prohibit calculation of objective functions.

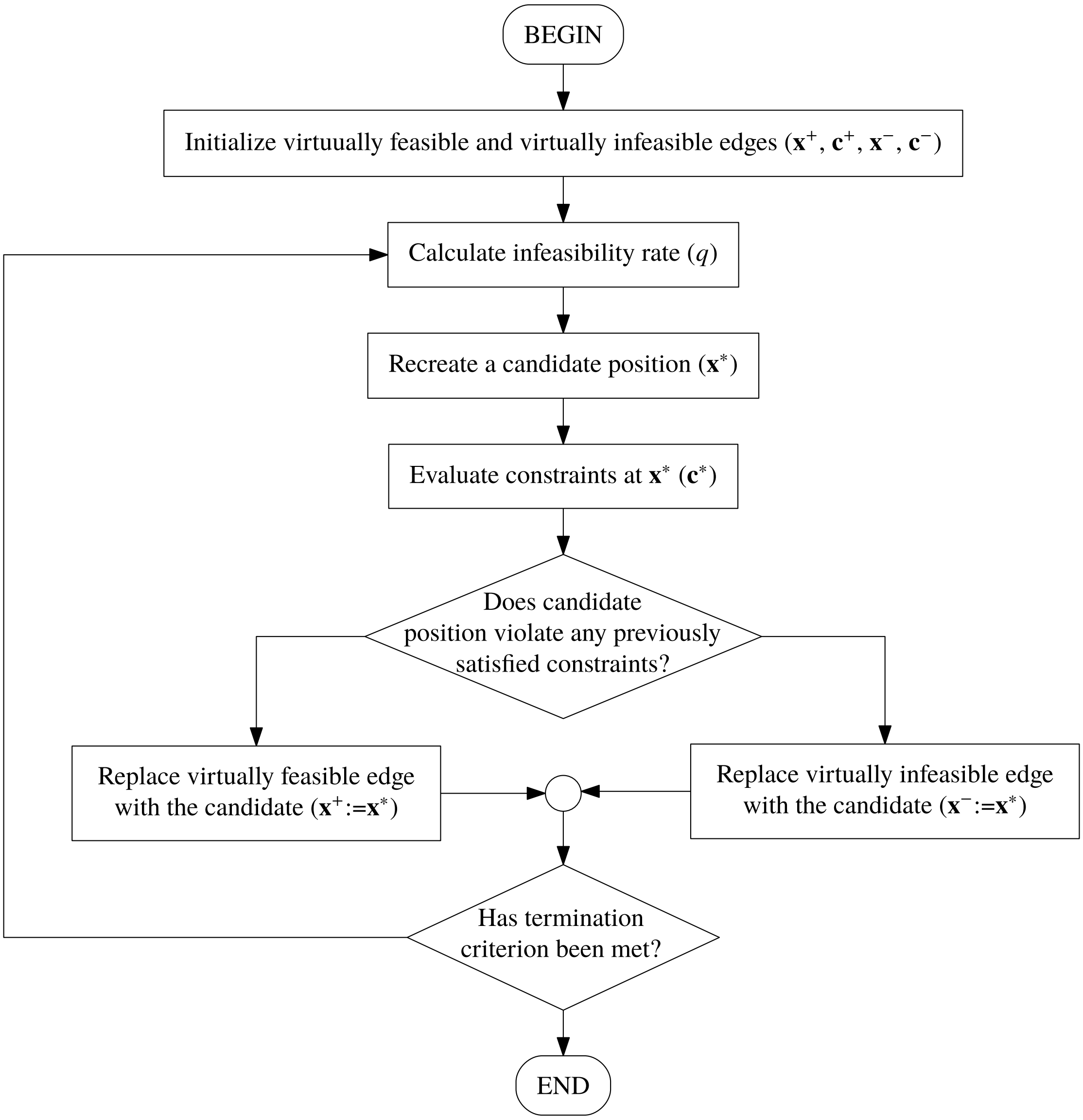

BEGIN
Initialize virtuually feasible and virtually infeasible edges ($\mathbf{x}^+$, $\mathbf{c}^+$, $\mathbf{x}^-$, $\mathbf{c}^-$)
Calculate infeasibility rate (q)
Recreate a candidate position ($\mathbf{x}^*$)
Evaluate constraints at $\mathbf{x}^*$ ($\mathbf{c}^*$)
Does candidate position violate any previously satisfied constraints?
Replace virtually feasible edge with the candidate ($\mathbf{x}^+$:=$\mathbf{x}^*$)
Replace virtually infeasible edge with the candidate ($\mathbf{x}^-$:=$\mathbf{x}^*$)
Has termination criterion been met?
END

Figure 3.5: Flowchart of the FEPSO virtual boundary search routine

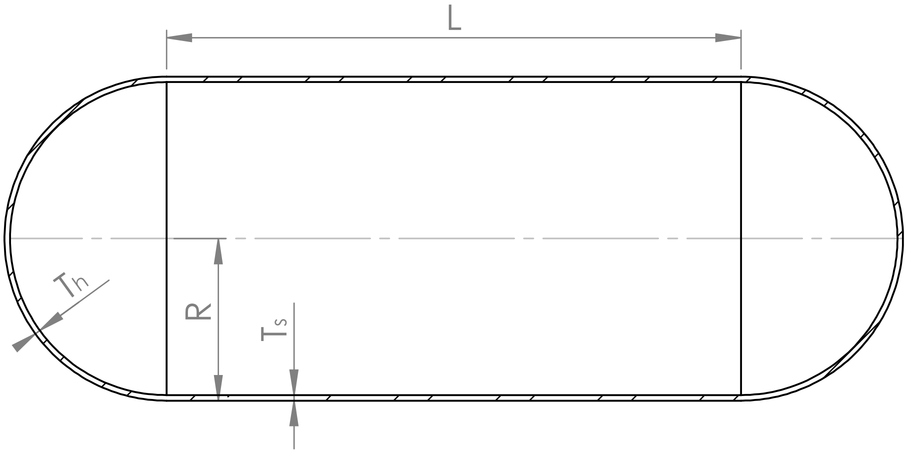


Figure 3.6: Illustration of pressure vessel design problem

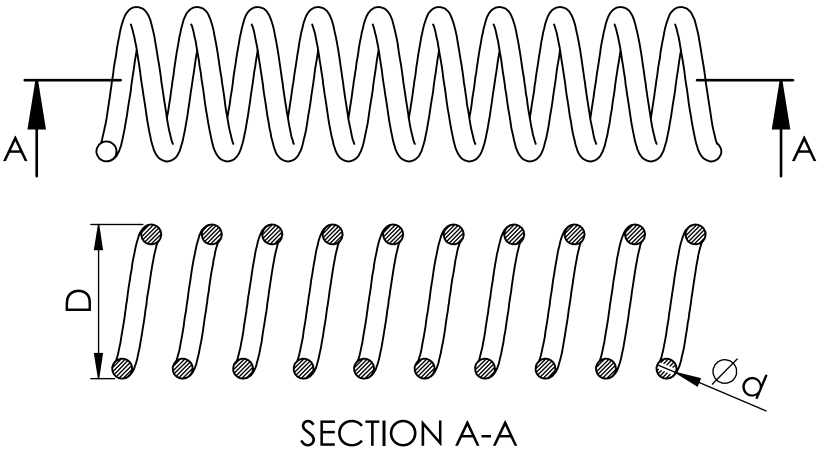


Figure 3.7: Illustration of tension/compression spring design problem

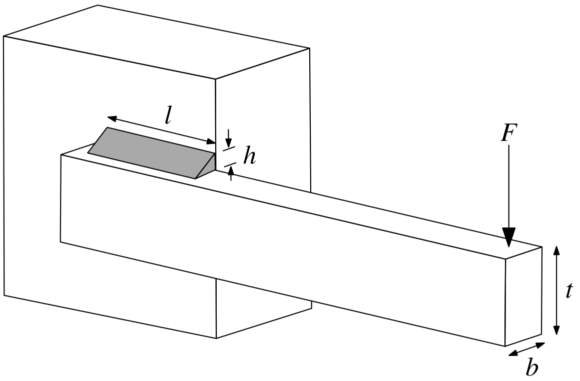


Figure 3.8: Illustration of welded beam design problem

CHAPTER 4

MULTI-OBJECTIVE FEASIBILITY ENHANCED PARTICLE SWARM OPTIMIZATION

Most engineering optimization problems involve several objective functions which are occasionally conflicting in nature. Hence, the goal of this chapter is to introduce a new method for dealing with highly constrained multi-objective problems. The presented method is based on FEPSO (see Chapter 3) that exclusively handles single objective optimization problems. *Multi-objective FEPSO* (MOFEPSO) is an enhanced approach utilizing a Pareto dominance technique to deal with multi-objective problems.

Mathematically, the main difference of type of problems MOFEPSO deals with is that there are multitude of objective functions and therefore for every position (i.e. point) in the decision space instead of a single objective value there is an objective vector. Therefore, as discussed in Section 2.1.1, a single solution does not exist.

4.1 General description of MOFEPSO

MOFEPSO is designed to handle highly-constrained multi-objective optimization problems. MOFEPSO employs repositories of non-dominated and feasible positions (or solutions) to guide feasible particle flight. Other than the mechanisms required for handling multiple objectives, MOFEPSO approach is identical to FEPSO. Hence, MOFEPSO shares the following five aspects with several modifications:

1. Initialization
2. Velocity update rules for infeasible particles (infeasible particle behavior)
3. Velocity update rules for feasible particles (feasible particle behavior)

4. Particle flight
    (a) Enforcing decision variable limits
    (b) Virtual boundary search
5. Post-flight operations

Differences come into play in initialization, velocity update rules for feasible particles, and post-flight operations. Remaining sections of this chapter focus on these differences and identical features of the algorithm will not be repeated. These identical features, namely the velocity update rules for infeasible particles and particle flight are explained in detail in sections 3.3 and 3.5 respectively.

MOFEPSO deals with constrained multi-objective optimization problems that can be defined as to *minimize*[1]

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \equiv \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \cdots & f_K(\mathbf{x}) \end{bmatrix}^T \tag{4.1}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq M} \tag{4.2}$$

A schematic representation of the algorithm is given in Figure 4.1. Actions shown in blue in this figure are unique to MOFEPSO and therefore will be detailed in the following sections.

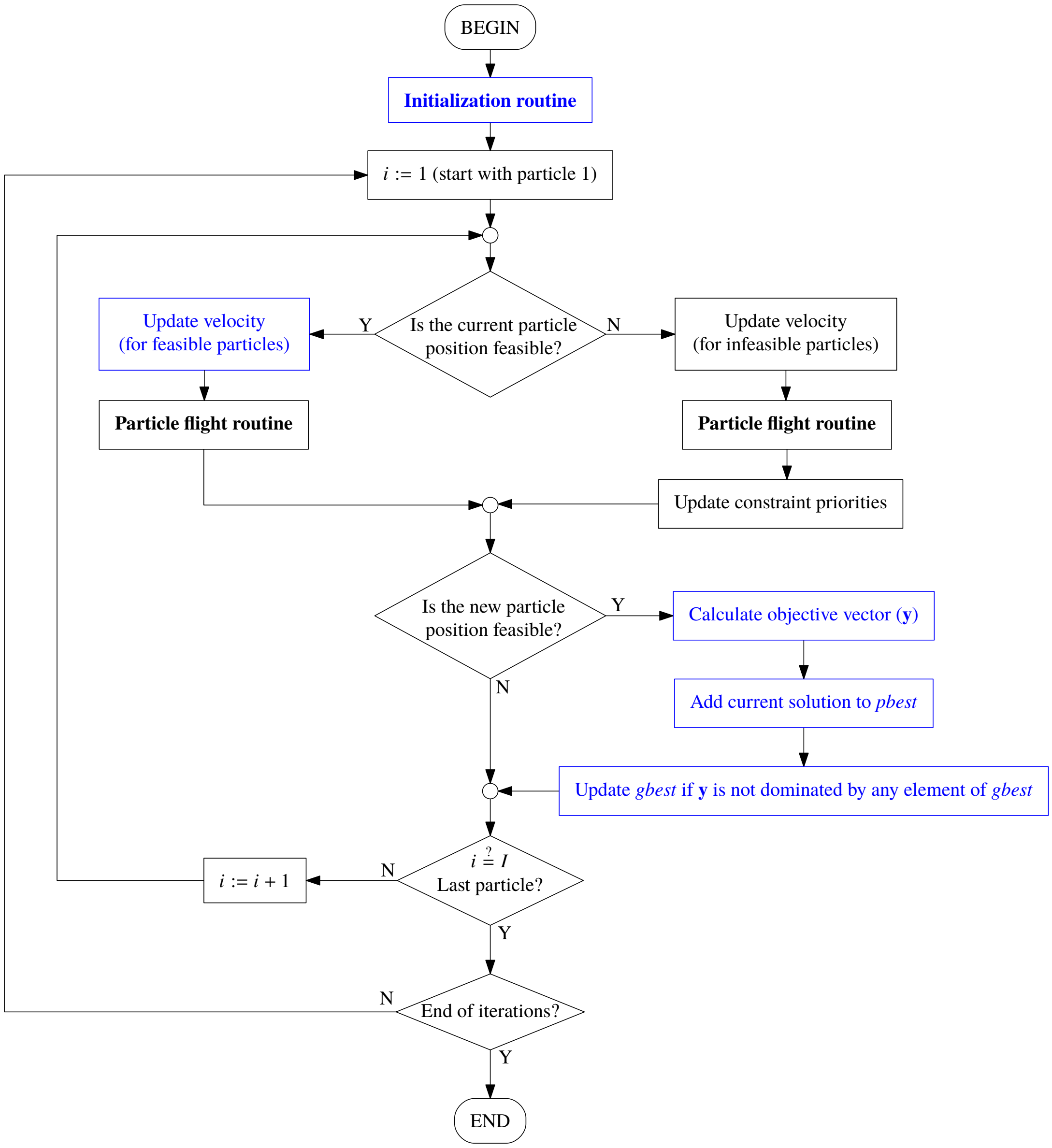
4.2 Sets of global- and personal non-dominated solutions: *gbest* and *pbest*

Before going into details of initialization and feasible particle velocity update rules, the concept of global- and personal non-dominated sets must be explained since it is used both when initializing the particles and after a particle moves into a feasible position. Due to the fact that MOFEPSO deals with multi-objective problems it requires a mechanism for keeping a repository of best solutions at swarm level and separate repositories of best solutions for every particle.

MOFEPSO employs a Pareto based approach where *gbest* becomes a set of non-dominated solutions among all feasible solutions found by all particles of the swarm.

---

[1] Note that the function $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^K$ is a *vector function* and therefore it is written in boldface.


BEGIN
Initialization routine
$i := 1$ (start with particle 1)
Is the current particle position feasible?
Y
N
Update velocity (for feasible particles)
Update velocity (for infeasible particles)
Particle flight routine
Particle flight routine
Update constraint priorities
Is the new particle position feasible?
Y
N
Calculate objective vector (y)
Add current solution to pbest
Update gbest if y is not dominated by any element of gbest
$i \overset{?}{=} I$
Last particle?
N
Y
$i := i + 1$
End of iterations?
N
Y
END

* Actions shown in blue indicate variations from FEPSO.

Figure 4.1: Flowchart of main MOFEPSO algorithm

On the other hand, the *pbest* set associated with each particle behaves in a similar way except that it keeps track of feasible positions attained by the particles themselves. The members of these sets are actually the ordered pairs of decision- and objective function vectors for non-dominated positions. The non-dominated set *gbest* can be written as

$$\mathbf{B} = \left\{ \mathbf{b}_\phi = \left(\mathbf{b}_\phi^x, \mathbf{b}_\phi^y \right) : \phi \in \mathbb{N}_{>0}^{\leq \Phi} \right\} \tag{4.3}$$

where Φ denotes the number of non-dominated positions. Similarly, each particle has its own non-dominated set (*pbest*) as

$$\mathbf{D}_i = \left\{ \mathbf{d}_{i,\omega} = \left(\mathbf{d}_{i,\omega}^x, \mathbf{d}_{i,\omega}^y \right) : \omega \in \mathbb{N}_{>0}^{\leq \Omega_i} \right\} \tag{4.4}$$

where Ω_i is the number of non-dominated positions associated with the ith particle.

If the ith particle is not feasible, $\mathbf{D}_i = \emptyset$ while $\mathbf{B} = \emptyset$ if there are no feasible particles in the swarm. In MOFEPSO, positions can only be in the *pbest* or *gbest* sets if they are feasible. Note that MOFEPSO does not calculate objective functions for infeasible positions. As will be discussed in more detail in the following sections, for every new particle position that is found to be feasible, the objective function is evaluated and the solution is checked against both the particle's *pbest* and *gbest*. For each set, if the new solution is not dominated by any member of the set, it is added to the set. Consequently, any existing member that is dominated by the new solution is removed.

4.3 Initialization

When compared with FEPSO, only difference in the initialization routine of MOFEPSO is related with how *gbest* set and individual *pbest* sets are initialized (see Figure 4.2). All other variables including the position matrix ($\mathbf{X}$), the velocity matrix ($\mathbf{V}$), the sensitivity matrix ($\mathbf{S}$), the AC selection count matrix ($\mathbf{T}$), and the constraint priority vector ($\mathbf{p}$) are initialized identically.

Before the particle initialization loop begins, *gbest* must be initialized such that

$$\mathbf{B} := \{\} \tag{4.5}$$

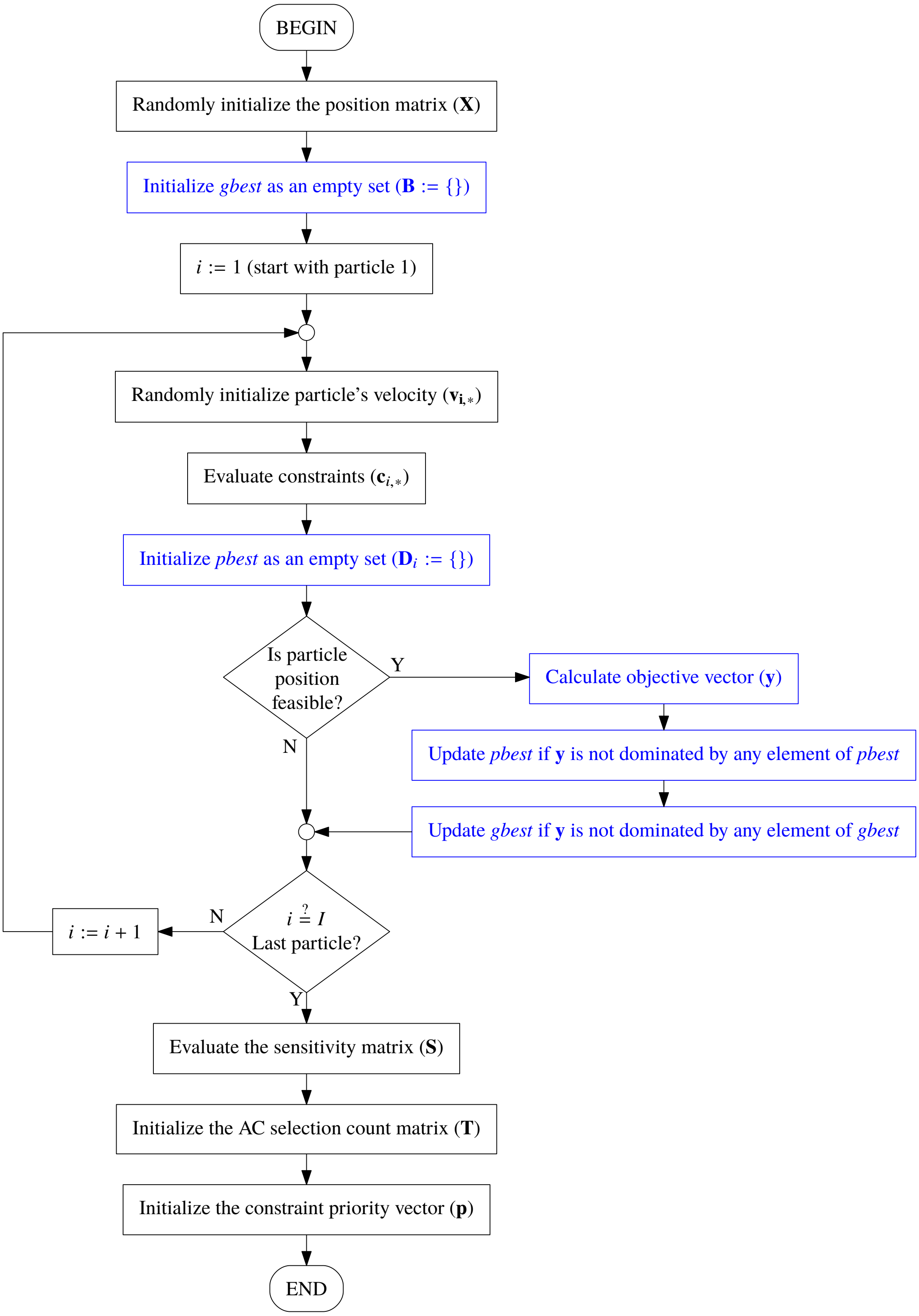

BEGIN
Randomly initialize the position matrix (**X**)
Initialize *gbest* as an empty set (**B** := {})
i := 1 (start with particle 1)
Randomly initialize particle's velocity ($\mathbf{v}_{\mathbf{i},*}$)
Evaluate constraints ($\mathbf{c}_{i,*}$)
Initialize *pbest* as an empty set ($\mathbf{D}_i$:= {})
Is particle position feasible?
Y
N
Calculate objective vector (**y**)
Update *pbest* if **y** is not dominated by any element of *pbest*
Update *gbest* if **y** is not dominated by any element of *gbest*
$i \overset{?}{=} I$
Last particle?
N
Y
$i := i + 1$
Evaluate the sensitivity matrix (**S**)
Initialize the AC selection count matrix (**T**)
Initialize the constraint priority vector (**p**)
END

Figure 4.2: Flowchart of MOFEPSO initialization routine

Within the particle loop[2], particle's *pbest* is also initialized as

$$\mathbf{D}_i := \{\} \tag{4.6}$$

After the constraints are calculated, if the particle position is found to be feasible, the objective function is evaluated and added to *pbest*. That is if

$$\left(\forall m \in \mathbb{N}_{>0}^{\leq M}\right)\left(c_{i,m} \leq 0\right) \tag{4.7}$$

then

$$\mathbf{y}^* := \mathbf{f}\left(\mathbf{x}_{i,*}^T\right) \tag{4.8}$$

$$\mathbf{D}_i := \mathbf{D}_i \cup \left\{\left(\mathbf{x}_{i,*}^T, \mathbf{y}^*\right)\right\} \tag{4.9}$$

Furthermore, if the recently calculated objective is non-dominated among *gbest*; that is if

$$\left(\nexists \phi \in \mathbb{N}_{>0}^{\leq \Phi}\right)\left(\mathbf{b}_\phi^y \prec \mathbf{y}^*\right) \tag{4.10}$$

then all elements of *gbest* that are dominated by $\mathbf{y}^*$ are removed such that

$$\mathbf{B}^- = \left\{\mathbf{b}_\phi : \mathbf{y}^* \prec \mathbf{b}_\phi^y \wedge \phi \in \mathbb{N}_{>0}^{\leq \Phi}\right\} \tag{4.11}$$

$$\mathbf{B} := \mathbf{B} \setminus \mathbf{B}^- \tag{4.12}$$

and the solution is added to *gbest*

$$\mathbf{B} := \mathbf{B} \cup \left\{\left(\mathbf{x}_{i,*}^T, \mathbf{y}^*\right)\right\} \tag{4.13}$$

4.4 Velocity update rules for feasible particles

Feasible particles in MOFEPSO are socially attracted to positions in the *gbest* set ($\mathbf{B}$) and cognitively attracted to particles in their own *pbest* set ($\mathbf{D}_i$). Therefore, a couple of positions randomly selected from *gbest* and *pbest* are used as global- ($\mathbf{x}^G$) and personal ($\mathbf{x}^P$) guide positions. To be specific, let ith particle be a feasible particle. After $\mathbf{x}^G$ and $\mathbf{x}^P$ are randomly selected from $\mathbf{B}$ and $\mathbf{D}_i$ respectively, its velocity is updated by

$$v_{i,n} := W v_{i,n} + C_1 r_{1,n}\left(x_n^G - x_{i,n}\right) + C_2 r_{2,n}\left(x_n^P - x_{i,n}\right) \tag{4.14}$$

where C_1 and C_2 are the social and cognitive acceleration coefficients respectively while $r_{1,n}, r_{2,n} \in [0,1] \subset \mathbb{R}$ are uniformly-distributed random numbers.

---

[2] The particle of interest in the loop is taken as the ith particle. All following equations are written accordingly.

4.5 Post-flight operations

As in FEPSO, the updated candidate position that satisfies both the decision variable limits and virtual feasibility rule, given by Eqns. 3.31 and 3.32 respectively, is assigned as the new position of the particle such that

$$\mathbf{x}_i := \mathbf{x}^* \qquad\qquad \mathbf{c}_i := \mathbf{x}^* \tag{4.15}$$

Some post-flight operations are conducted depending on pre-flight- and post-flight feasibilities of the particle. If the particle's previous position was infeasible, constraint priorities are updated using Eqn. 3.21.

If the new position is found to be feasible[3], the objective vector ($\mathbf{y}^* := f(\mathbf{x}^*)$) is calculated. If

$$\left(\nexists \omega \in \mathbb{N}_{>0}^{\leq \Omega_i}\right)\left(\mathbf{d}_{i,\omega}^{y} \prec \mathbf{y}^*\right) \tag{4.16}$$

then *pbest* is updated such that

$$\mathbf{D}^- = \left\{\mathbf{d}_{i,\omega} : \mathbf{y}^* \prec \mathbf{d}_{i,\omega}^{y} \wedge \omega \in \mathbb{N}_{>0}^{\leq \Omega_i}\right\} \tag{4.17}$$

$$\mathbf{D}_i := \mathbf{D}_i \setminus \mathbf{D}^- \tag{4.18}$$

and the solution is added to *pbest*

$$\mathbf{D}_i := \mathbf{D}_i \cup \left\{\left(\mathbf{x}_{i,*}^{T}, \mathbf{y}^*\right)\right\} \tag{4.19}$$

Similarly, $\mathbf{y}^*$ is checked against *gbest* using Eqn. 4.10 and if it is found to be non-dominated, *gbest* is updated using Eqns. 4.11, 4.12, and 4.13.

4.6 Closure

This chapter introduces the multi-objective version of the FEPSO. As a consequence of differing characteristics of type of problems MOFEPSO deals with, additional techniques particularly with regard to handling of multiple objectives are necessary. Pareto based approaches for handling such objectives are implemented in MOFEPSO. Hence, non-dominated repositories are maintained at both swarm- and particle level.

---

[3] $\left(\forall m \in \mathbb{N}_{>0}^{\leq M}\right)(c_{i,m} \leq 0)$

Social- and cognitive guides are randomly selected from these repositories to govern particle flight. Other aspects of MOFEPSO such as the infeasible particle behavior, particle flight routine, and virtual boundary search are identical with FEPSO.

Since MOFEPSO shares the same roots with FEPSO, it retains main characteristics and advantages. Satisfied constraint are never allowed to become violated in MOFEPSO too. This provides a progressive improvement in terms of constraint satisfaction level of the swarm. Besides, objective function is not calculated at infeasible points explored in the decision space. This feature saves computational resources especially when complicated objective functions are involved.

CHAPTER 5

FOUR-STAGE GEAR TRAIN PROBLEM

5.1 Single-objective four-stage gear train problem

The original four-stage *gear train problem* (GTP) proposed by Pomrehn and Papalambros [93] is a highly-constrained single objective problem that many algorithms have difficulty in finding feasible solutions without exercising special techniques. In fact, many research efforts employ GTP as a benchmark test. The optimization approaches to this problem involve reformulations, modifications, reductions, heuristic injections, or designer interaction due to its highly-constrained nature in its original form. Pomrehn and Papalambros [94] apply special solution space reduction techniques; Khorshid and Seireg [96] utilize designer interaction; Dolen, Kaplan, and Seireg [98] develop a genetic algorithm incorporating a heuristic search on the most-violated constraints (which in turn leads to the reduction of the search domain); Wang and Yin [95] use *ranking selection-based particle swarm optimization* (RSPSO); Savsani, Rao, and Vakharia [97] apply *biogeography based optimization* (BBO); Rao and Savsani [86] use multiple optimization algorithms including BBO, PSO, *artificial bee colony* (ABC), *differential evolution* (DE), and *artificial immune algorithm* (AIA); Savsani [155] makes good use of a hybrid optimization algorithm fusing BBO and ABC; Savsani and Savsani [156] utilize *passing vehicle search* (PVS) to solve GTP.

Without modifying the problem, finding a feasible solution requires tens to hundreds of thousands of constraint function evaluations [86, 95]. Note that a Monte Carlo Simulation with uniform random sampling of 10^8 samples failed to find a feasible solution for this problem in this study. This problem is very suitable as a benchmark for optimization algorithms dealing with highly-constrained problems. Problem definition is repeated in this chapter for the sake of self-containment while two new variants of the problem are introduced.

5.1.1 Original four-stage gear train problem

The problem[1] consists of the design of a four-stage gear train. Its objective is to minimize the weight (which is directly proportional to total of volumes of all pinions and gears) while conforming to a total of 86 constraints involving tooth bending fatigue strength, tooth contact strength, contact ratio, gear pitch, output speed, and geometric constraints. The aim is to place pinion and gears of each stage at discrete positions within a box of $127\,\mathrm{mm} \times 127\,\mathrm{mm}$ and to determine their sizes. Figure 5.1 shows a stage of the gear train and discrete shaft locations. Design variables include the positions of the gears in each stage, pinion position in the first stage (other stages have their pinions positioned at the position of the previous stage's gear), number of teeth for gears and pinions, and gear thickness for each stage.

GTP is formulated as a single-objective optimization problem which can be defined as to minimize

$$y = f(\mathbf{x}) = \pi \sum_{s=1}^{4} b_s (C_s)^2 \frac{(N_s^p)^2 + (N_s^g)^2}{(N_s^p + N_s^g)^2} \tag{5.1}$$

subject to

$$g_m(\mathbf{x}) \le 0, \quad m = 1, \dots, M \tag{5.2}$$

where $M = 86$,

$$\begin{aligned} \mathbf{x} =& [x_1 \quad \cdots \quad x_{22}]^T \\ =& [X_0 \quad Y_0 \quad \cdots \quad X_4 \quad Y_4 \quad b_1 \quad \cdots \quad b_4 \quad N_1^P \quad \cdots \quad N_4^P \quad N_1^G \quad \cdots \quad N_4^G]^T \end{aligned} \tag{5.3}$$

and C_s is the distance between pinion and gear centers such that

$$C_s = \sqrt{(X_s - X_{s-1})^2 + (Y_s - Y_{s-1})^2} \tag{5.4}$$

Therefore, $N = 22$ and the decision variables converted to problem domain are summarized in Table 5.1. Note that the four stages of the gear train are represented by

$$s \in \{1, 2, 3, 4\} \tag{5.5}$$

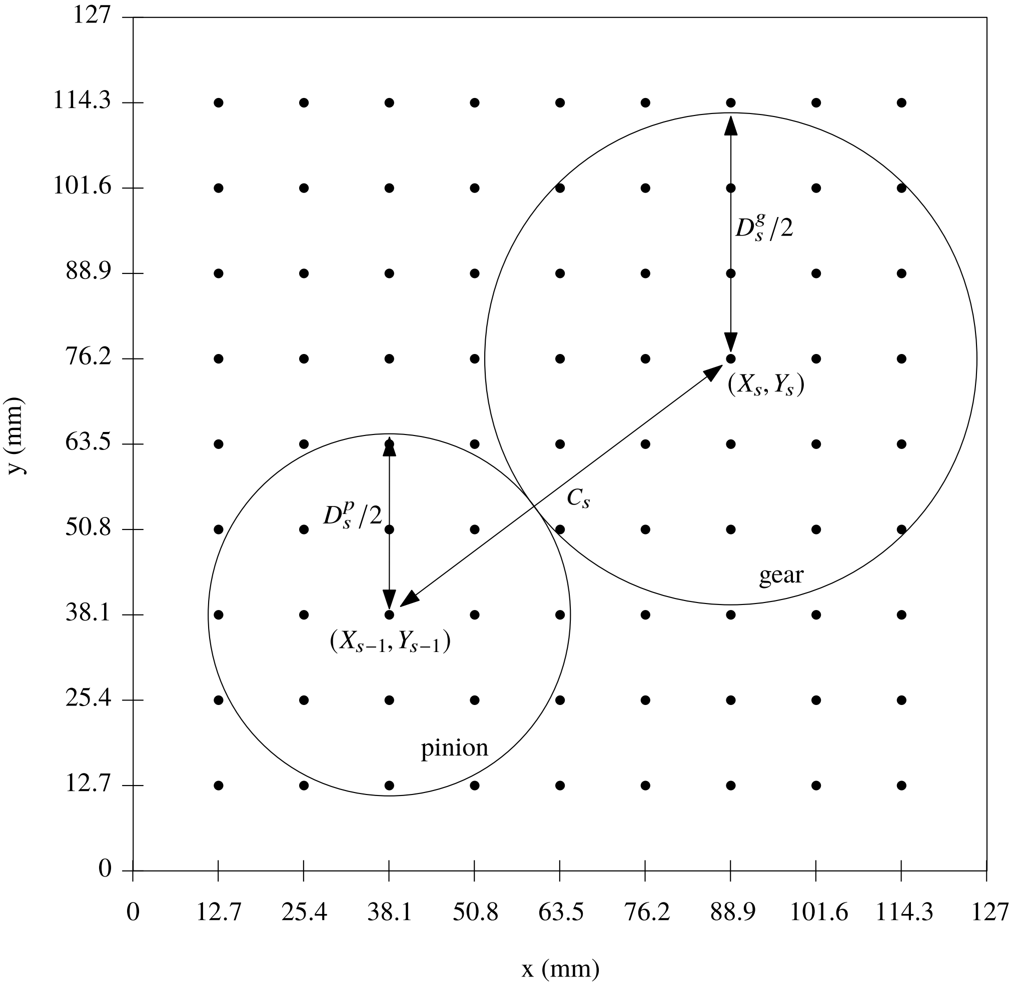


Figure 5.1: Representative pitch circles associated with discrete shaft locations of the pinion and gear at stage s

Table 5.1: Decision variables of GTP

Variable	**Description**	**Allowed values**
(X_0, Y_0)	First pinion position (mm)	$X_s, Y_s \in 12.7 \times \{1, 2, \cdots, 9\}$ $s \in \{0, 1, 2, 3, 4\}$
(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), (X_4, Y_4)	Gear positions of stages 1, 2, 3, and 4 (mm)	
b_1, b_2, b_3, b_4	Gear and pinion thickness (mm)	$b_s \in \{3.175, 5.715, 8.255, 12.7\}$ $s \in \{1, 2, 3, 4\}$
N_1^p, N_2^p, N_3^p, N_4^p	Pinion tooth numbers	$N_s^p, N_s^g \in \{7, 8, 9, ..., 60\}$ $s \in \{1, 2, 3, 4\}$
N_1^g, N_2^g, N_3^g, N_4^g	Gear tooth numbers	

Definitions of constraint functions $g_1, \ldots, g_{86}$ are given in Table 5.2. Calculation of parameters used in constraint function definitions are explained below. Some constants and pre-calculated limits required for these parameter calculations are tabulated in Table 5.3. Gear tooth bending fatigue stress ($\mathrm{kgf}/\mathrm{cm}^2$), gear tooth contact fatigue stress ($\mathrm{kgf}/\mathrm{cm}^2$), and gear tooth contact ratio take the following form

$$\sigma_s^{GTB} = \left(\frac{366000}{\pi \omega_{s-1}} + 2 \frac{C_s N_s^p}{N_s^p + N_s^g} \right) \frac{(N_s^p + N_s^g)^2}{4 b_s C_s^2 N_s^p} \tag{5.6}$$

$$\sigma_s^{GTC} = \left(\frac{366000}{\pi \omega_{s-1}} + 2 \frac{C_s N_s^p}{N_s^p + N_s^g} \right) \left(\frac{(N_s^p + N_s^g)^3}{4 b_s C_s^2 (N_s^p)^2 N_s^g} \right) \tag{5.7}$$

$$\begin{aligned} CR_s = \frac{1}{\pi \cos \phi} \Bigg[& N_s^p \sqrt{\frac{\sin^2 \phi}{4} + \frac{1}{N_s^p} + \frac{1}{(N_s^p)^2}} \\ & + N_s^g \sqrt{\frac{\sin^2 \phi}{4} + \frac{1}{N_s^g} + \frac{1}{(N_s^g)^2}} - (N_s^p + N_s^g) \frac{\sin \phi}{2} \Bigg] \end{aligned} \tag{5.8}$$

and they must satisfy

$$\sigma_s^{GTB} \leq S^{GTB} \tag{5.9}$$

$$\sigma_s^{GTC} \leq S^{GTC} \tag{5.10}$$

$$CR_s \geq CR^{min} \tag{5.11}$$

---

[1] The original four-stage gear train problem is abbreviated as GTP throughout this dissertation.

As shown in definitions of constraint functions $g_1, \ldots, g_{12}$ in Table 5.2. There are also constraints on diameters of pinions and gears ($g_{13}, \ldots, g_{20}$):

$$D_s^p = 2C_s \frac{N_s^p}{N_s^p + N_s^g} \geq D^{min} \tag{5.12}$$

$$D_s^g = 2C_s \frac{N_s^g}{N_s^p + N_s^g} \geq D^{min} \tag{5.13}$$

Table 5.2: Constraint functions for GTP

	m	g_m
Gear stage scope	1-4	$\sigma_s^{GTB} - S^{GTB}$
$s \in \{1, 2, 3, 4\}$	5-8	$\sigma_s^{GTC} - S^{GTC}$
	9-12	$CR^{min} - CR_s$
	13-16	$D^{min} - D_s^p$
	17-20	$D^{min} - D_s^g$
	21-24	$-\left(X_{s-1} - r_s^{po}\right)$
	25-28	$X_{s-1} + r_s^{po} - L^{max}$
	29-32	$-\left(Y_{s-1} - r_s^{po}\right)$
	33-36	$Y_{s-1} + r_s^{po} - L^{max}$
	37-40	$-\left(X_s - r_s^{go}\right)$
	41-44	$X_s + r_s^{go} - L^{max}$
	45-48	$-\left(Y_s - r_s^{go}\right)$
	49-52	$Y_s + r_s^{go} - L^{max}$
	53-68	$P_s^{min1}, P_s^{min2}, P_s^{min3}, P_s^{min4}$
	69-84	$P_s^{max1}, P_s^{max2}, P_s^{max3}, P_s^{max4}$
Gear train scope	85	$\omega^{min} - \omega_4$
	86	$\omega_4 - \omega^{max}$

The parameters needed for geometric constraints are outer radii of the pinions and gears for each stage s:

$$r_s^{po} = \frac{D_s^p}{2} + \frac{2C_s}{N_s^p + N_s^g} \tag{5.14}$$

$$r_s^{go} = \frac{D_s^g}{2} + \frac{2C_s}{N_s^p + N_s^g} \tag{5.15}$$

Table 5.3: Constants for GTP

Constant	Description
$CR^{min} = 1.4$	Allowable contact ratio
$C_p = 464.0$	Elastic coefficient
$\phi = 20°$	Pressure angle
$W = 55.9\,\mathrm{W}$	Input power
$J_R = 0.2$	Geometry factor
$K_M = 1.6$	Mounting factor
$K_O = 1.5$	Overload factor
$\sigma^H = 3290\,\mathrm{kgf/cm^2}$	Allowable fatigue stress
$\sigma^N = 2090\,\mathrm{kgf/cm^2}$	Allowable bending stress
$\omega_0 = 5000\,\mathrm{rpm}$	Input speed
$\omega^{min} = 245\,\mathrm{rpm}$	Minimum output speed
$\omega^{\max} = 255\,\mathrm{rpm}$	Maximum output speed
$D^{min} = 25.4\,\mathrm{mm}$	Minimum pinion/gear diameter
$L^{max} = 127\,\mathrm{mm}$	Maximum housing dimension

The geometric constraints are to confine pinions and gears in the box with dimensions $127\,\mathrm{mm} \times 127\,\mathrm{mm}$ ($g_{21}, \ldots, g_{52}$):

$$X_{s-1} - r_s^{po} \geq 0 \tag{5.16}$$

$$X_{s-1} + r_s^{po} \leq L^{max} \tag{5.17}$$

$$Y_{s-1} - r_s^{po} \geq 0 \tag{5.18}$$

$$Y_{s-1} + r_s^{po} \leq L^{max} \tag{5.19}$$

$$X_s - r_s^{go} \geq 0 \tag{5.20}$$

$$X_s + r_s^{go} \leq L^{max} \tag{5.21}$$

$$Y_s - r_s^{go} \geq 0 \tag{5.22}$$

$$Y_s + r_s^{go} \leq L^{max} \tag{5.23}$$

Gear pitch must also satisfy these constraints:

$$P_s^{min} \leq P_s \leq P_s^{max} \tag{5.24}$$

where P_s, P_s^{min}, and P_s^{max} are gear pitch, minimum gear pitch, and maximum gear pitch values for sth gear stage respectively. These values are calculated as follows:

$$P_s = \frac{N_s^p + N_s^g}{2C_s}, \tag{5.25}$$

$$P_s^{min} = \begin{cases} 0.472, & b_s = 3.175 \\ 0.323, & b_s = 5.715 \\ 0.252, & b_s = 8.255 \\ 0, & b_s = 12.7 \end{cases} \tag{5.26}$$

$$P_s^{max} = \begin{cases} 0.906, & b_s = 3.175 \\ 0.472, & b_s = 5.715 \\ 0.323, & b_s = 8.255 \\ 0.252, & b_s = 12.7 \end{cases} \tag{5.27}$$

As seen in constraints $g_{53}, \ldots, g_{84}$ in Table 5.2 separate criteria can be constructed for each gear thickness alternative. Derived gear pitch constraints used in GTP are as follows:

$$P_s^{min1} = -\left(0.472 - P_s\right)\left(b_s - 5.715\right)\left(b_s - 8.255\right)\left(b_s - 12.7\right) \leq 0 \tag{5.28}$$

$$P_s^{min2} = +\left(0.323 - P_s\right)\left(b_s - 3.175\right)\left(b_s - 8.255\right)\left(b_s - 12.7\right) \leq 0 \tag{5.29}$$

$$P_s^{min3} = -\left(0.252 - P_s\right)\left(b_s - 3.175\right)\left(b_s - 5.715\right)\left(b_s - 12.7\right) \leq 0 \tag{5.30}$$

$$P_s^{min4} = +\left(0 - P_s\right)\left(b_s - 3.175\right)\left(b_s - 5.715\right)\left(b_s - 8.255\right) \leq 0 \tag{5.31}$$

$$P_s^{max1} = -\left(P_s - 0.906\right)\left(b_s - 5.715\right)\left(b_s - 8.255\right)\left(b_s - 12.7\right) \leq 0 \tag{5.32}$$

$$P_s^{max2} = +\left(P_s - 0.472\right)\left(b_s - 3.175\right)\left(b_s - 8.255\right)\left(b_s - 12.7\right) \leq 0 \tag{5.33}$$

$$P_s^{max3} = -\left(P_s - 0.323\right)\left(b_s - 3.175\right)\left(b_s - 5.715\right)\left(b_s - 12.7\right) \leq 0 \tag{5.34}$$

$$P_s^{max4} = +\left(P_s - 0.252\right)\left(b_s - 3.175\right)\left(b_s - 5.715\right)\left(b_s - 8.255\right) \leq 0 \tag{5.35}$$

Finally, the output speed of each stage is computed as

$$\omega_s = \omega_{s-1}\frac{N_s^p}{N_s^g}. \tag{5.36}$$

Other than the stage scope constraints, two more constraints related with the output speed exist (g_{85}, g_{86}):

$$\omega^{min} \leq \omega_4 \leq \omega^{max} \tag{5.37}$$

5.1.2 Reduced constraints four-stage gear train problem

Reduced-constraint problem (RC-GTP) is actually same as the original problem. The only difference is the treatment of the gear pitch constraints. Reduced-constraint problem uses Eqn. 5.24 instead of Eqns. 5.28-5.35. Therefore, as seen in in Table 5.4 reduced-constraint problem has only 62 constraints ($M = 62$) whereas the original problem has 86.

Although the number of constraints decreases in this case, the problem actually becomes more implicit. The underlying reason is that the newly introduced constraints are actually harder to satisfy because they become coupled to gear thickness as indicated by Eqns. 5.26 and 5.27.

Table 5.4: Constraint functions for RC-GTP

	m	g_m
Gear stage scope	1-4	$\sigma_s^{GTB} - S^{GTB}$
$s \in \{1, 2, 3, 4\}$	5-8	$\sigma_s^{GTC} - S^{GTC}$
	9-12	$CR^{min} - CR_s$
	13-16	$D^{min} - D_s^p$
	17-20	$D^{min} - D_s^g$
	21-24	$-\left(X_{s-1} - r_s^{po}\right)$
	25-28	$X_{s-1} + r_s^{po} - L^{max}$
	29-32	$-\left(Y_{s-1} - r_s^{po}\right)$
	33-36	$Y_{s-1} + r_s^{po} - L^{max}$
	37-40	$-\left(X_s - r_s^{go}\right)$
	41-44	$X_s + r_s^{go} - L^{max}$
	45-48	$-\left(Y_s - r_s^{go}\right)$
	49-52	$Y_s + r_s^{go} - L^{max}$
	53-56	$p_s^{min} - p_s$
	57-60	$p_s - p_s^{max}$
Gear train scope	61	$\omega^{min} - \omega_4$
	62	$\omega_4 - \omega^{max}$

5.1.3 Mixed-valued four-stage gear train problem

Mixed-valued problem (MV-GTP) is similar to RC-GTP, only difference being that the gears are allowed to be located anywhere within the box of dimensions $127\,\mathrm{mm} \times 127\,\mathrm{mm}$ instead of the discrete shaft locations shown in Figure 5.1. This implies that

$$X_s, Y_s \in \{x \in \mathbb{R} : 12.7 \leq x \leq 114.3\} \tag{5.38}$$

All other decision variable definitions remain the same (Table 5.5). On the other hand, the constraint functions of the MV-GTP are same as RC-GTP (Table 5.4).

Table 5.5: Decision variables of MV-GTP

Variable	**Description**	**Allowed values**
(X_0, Y_0)	First pinion position (mm)	$X_s, Y_s \in$ $\{x \in \mathbb{R} : 12.7 \leq x \leq 114.3\}$ $s \in \{0, 1, 2, 3, 4\}$
(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), (X_4, Y_4)	Gear positions of stages 1, 2, 3, and 4 (mm)	
b_1, b_2, b_3, b_4	Gear and pinion thickness (mm)	$b_s \in \{3.175, 5.715, 8.255, 12.7\}$ $s \in \{1, 2, 3, 4\}$
N_1^p, N_2^p, N_3^p, N_4^p	Pinion tooth numbers	$N_s^p, N_s^g \in \{7, 8, 9, ..., 60\}$ $s \in \{1, 2, 3, 4\}$
N_1^g, N_2^g, N_3^g, N_4^g	Gear tooth numbers	

5.1.4 Results and discussion

FEPSO was run 50 times for each variant of four-stage gear train optimization problem: GTP, RC-GTP, and MV-GTP. The parameter configuration used in runs is summarized in Table 5.6. Except for a single unsuccessful run in the RC-GTP all FEPSO runs were able to find feasible solutions. A summary of simulation results is given in Table 5.7. In this table, number of *constraint function evaluations* (CFEs) when first feasible point is found, as well as the number of CFEs and *objective function evaluations* (OFEs) when the best value was found are presented. Design variable values of the best solutions found by FEPSO for each problem variant are given in Table 5.8 and are illustrated in Figure 5.2.

Table 5.6: FEPSO algorithm parameters used in simulations

Swarm Size	Max. Num. of Iterations	W	C_1	C_2	x^{tol}
200	500	0.9 - 0.4[(a)]	2	2	0.01

[(a)] The inertia factor (W) is linearly varied throughout the iterations.

Single objective algorithms DOPSO [65], HPSO [141], and GA (MI-LXPM) [157]; multi-objective algorithms MOGA2 and NSGA2 are used in all three problems for comparison purposes (see Tables 5.9, 5.10, and 5.11). Although it is unnecessary to utilize a multi-objective algorithm here, MOGA2 and NSGA2 are well known for their good performances and constraint handling capabilities. Since both algorithms can easily be employed in a single objective problem, they have been utilized for benchmarking FEPSO.

A comparison of solutions found in the literature for the GTP is also summarized in Table 5.9. In DOPSO inertia factor was taken $\omega = 1$, the objective optimization threshold was taken $\delta = 0$, and the swarm size was chosen as 200. The HPSO parameters were selected as follows: $M = 250$, $c_1 = 2.0$, $c_2 = 2.0$, $w = 0.9 \rightarrow 0.4$, $L = 20$, $\lambda = 0.94$, $\eta = 0.001$. Note that the standard mixed integer GA implementation of MATLAB® was used for MI-LXPM with a population size of 500 and a maximum number of generations of 200. MOGA2 and NSGA2 simulations were carried out in modeFRONTIER®. The selected MOGA2 and NSGA2 options are summarized in Tables 5.12 and 5.13 respectively.

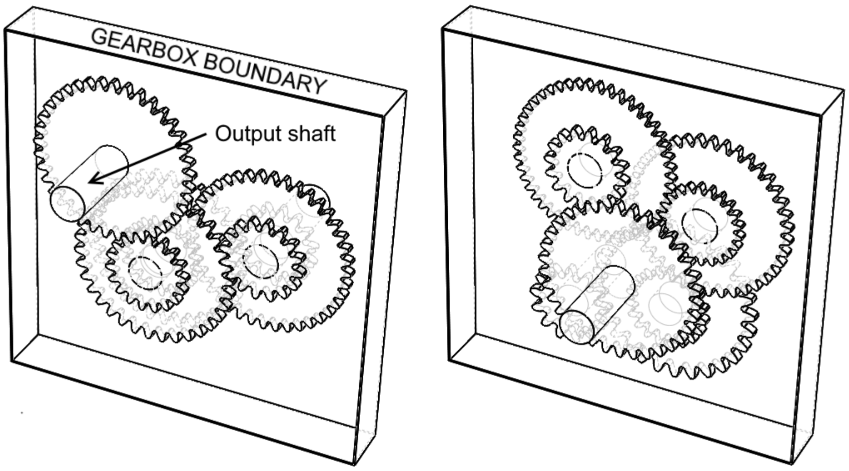
As can be seen in Table 5.9, the best FEPSO solution for the original problem is among the best three found in the literature including the papers where special solution space reduction [98] or designer interaction [96] methods are applied. Furthermore, it is one of the two techniques with 100% success rate. FEPSO has the best mean-objective-value among the studies that report a mean value. Although only one other paper [95] reports the worst objective value obtained in the simulations, the worst value obtained by FEPSO is considerably better (41.11 vs. 68.50).

Table 5.7: FEPSO simulation results summary of all three four-stage gear train problems

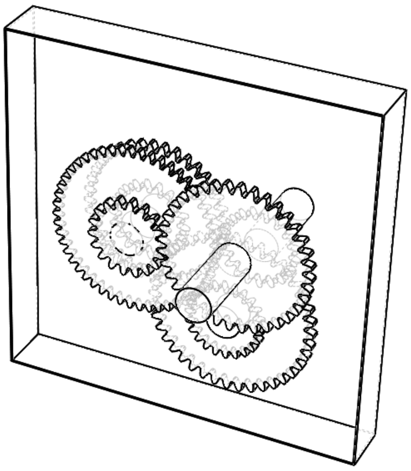
Results		**GTP**	**RC-GTP**	**MV-GTP**
# of variables		22	22	10: real 12: discrete
# of constraints		86	62	62
Runs		50	50	50
Successful Runs		50	49	50
# of CFEs when first feasible is found	Best	13 279	19 174	18 498
	Worst	69 275	49 254	93 327
	Mean	37 077	33 607	34 752
	Std. Dev.	11 007	7170	11 486
Objective value of first feasible (cm^3)	Best	39.199	38.340	38.910
	Worst	94.721	85.458	106.135
	Mean	52.830	55.779	61.060
	Std. Dev.	10.257	11.756	15.220
# of CFEs when best is found	Best	49 843	42 394	35 765
	Worst	203 455	226 532	215 409
	Mean	141 617	136 369	199 256
	Std. Dev.	43 479	49 139	31 929
# of OFEs when best is found	Best	2	2	2
	Worst	451	440	1841
	Mean	77	66	937
	Std. Dev.	77	76	455
Best value found (cm^3)	Best	36.250	36.254	35.226
	Worst	41.108	47.310	51.500
	Mean	38.056	38.611	37.947
	Std. Dev.	1.251	1.948	3.358

Table 5.8: Best solutions found during FEPSO simulations for all three problems

	GTP	RC-GTP	MV-GTP
X_0	88.9	38.1	83.395
Y_0	63.5	25.4	60.469
X_1	50.8	76.2	46.087
Y_1	50.8	38.1	63.507
X_2	88.9	88.9	76.903
Y_2	63.5	76.2	36.581
X_3	50.8	50.8	43.812
Y_3	50.8	88.9	61.731
X_4	38.1	63.5	81.800
Y_4	88.9	50.8	64.651
b_1	3.175	3.175	3.175
b_2	3.175	3.175	3.175
b_3	3.175	3.175	3.175
b_4	3.175	3.175	3.175
N_1^g	32	28	37
N_2^g	40	39	40
N_3^g	34	46	50
N_4^g	37	34	36
N_1^p	15	13	19
N_2^p	19	19	18
N_3^p	16	22	22
N_4^p	18	16	18
y	**36.250**	**36.254**	**35.226**



(a) GTP solution by FEPSO (b) GTP solution by FEPSO



(c) GTP solution by FEPSO

Figure 5.2: Best GTP, RC-GTP, and MV-GTP solutions found by FEPSO

Table 5.9: Comparison of results for GTP

Algorithm		Max OFE	Max CFE	Runs	Succ. runs	Best	Mean	Worst	Std. Dev.
Pomrehn (1995) [93]		N/A	N/A	N/A	N/A	91.87	N/A	N/A	N/A
Pomrehn(1) (1995) [94]		N/A	N/A	N/A	N/A	83.92	N/A	N/A	N/A
Khorsid(2) (1999) [96]		N/A	N/A	N/A	N/A	38.13	N/A	N/A	N/A
Dolen(1) (2005) [98]		10 000	10 000	100	60	35.40	39.78	N/A	N/A
	(without reduction)	10 000	10 000	100	0	-	-	-	-
Wang - RSPSO		10 000	10 000	30	3	44.58	55.37	68.50	12.13
(2008) [95]		80 000	80 000	30	29	38.11	51.84	76.08	10.42
Savsani	BBO	100 000	100 000	100	36	37.28	N/A	N/A	N/A
(2009) [97]	PSO	100 000	100 000	100	0	-	-	-	-
	DE	100 000	100 000	100	0	-	-	-	-
Rao	BBO	100 000	100 000	N/A	N/A	36.57	N/A	N/A	N/A
(2012) [86]	PSO	100 000	100 000	N/A	0	-	-	-	-
	ABC	100 000	100 000	N/A	0	-	-	-	-
	DE	100 000	100 000	N/A	0	-	-	-	-
	AIA	100 000	100 000	N/A	0	-	-	-	-
Savsani	HBBABC	5000	5000	100	8	46.20	N/A	N/A	N/A
(2012) [155]		10 000	10 000	100	100	35.36	52.49	N/A	N/A
	BBO	5000	5000	100	0	-	-	-	-
		10 000	10 000	100	24	43.64	N/A	N/A	N/A
	ABC	5000	5000	100	0	-	-	-	-
		10 000	10 000	100	36	40.41	N/A	N/A	N/A
Savsani (2016) - PVS [156]		25 000	25 000	50	44	37.27	N/A	N/A	N/A
DOPSO [65]		5000	400 000	20	2	97.93	99.21	100.49	1.81
HPSO [141]		2500	400 000	20	1	99.18	99.18	99.18	-
GA (MI-LXPM) [157]		100 000	100 000	20	0	-	-	-	-
MOGA2 [145]		100 000	100 000	5	0	-	-	-	-
NSGA2 [28]		100 000	100 000	5	0	-	-	-	-
FEPSO		**460**	**205 000**	**50**	**50**	**36.25**	**38.06**	**41.11**	**1.25**

(1) Applies special techniques for solution space reduction.

(2) Involves designer interaction.

Table 5.10: Comparison of results for RC-GTP

Algorithm	Max OFE	Max CFE	Runs	Succ. runs	Best	Mean	Worst	Std. Dev.
DOPSO	5000	400 000	20	1	74.96	74.96	74.96	-
HPSO	2500	400 000	20	0	-	-	-	-
MOGA2	75 000	75 000	5	5	36.51	39.76	43.58	2.89
NSGA2	100 000	100 000	5	3	49.09	60.98	80.52	17.05
FEPSO	**440**	**230 000**	**50**	**49**	**36.25**	**38.61**	**47.31**	**1.95**

Table 5.11: Comparison of results for MV-GTP

Algorithm	Max OFE	Max CFE	Runs	Succ. runs	Best	Mean	Worst	Std. Dev.
DOPSO	5000	400 000	20	2	68.13	97.64	127.15	41.73
HPSO	2500	400 000	20	0	-	-	-	-
GA (MI-LXPM)	100 000	100 000	20	15	43.24	52.23	62.69	6.37
MOGA2	100 000	100 000	5	5	36.35	39.63	44.41	3.84
NSGA2	100 000	100 000	5	4	51.28	54.07	56.87	3.95
FEPSO	**1900**	**220 000**	**50**	**50**	**35.23**	**37.95**	**51.50**	**3.36**

Table 5.12: Algorithm parameters used for MOGA2 in benchmark simulations

Number of Individuals	500 (200[(1)])
Number of Generations	200 (500[(1)])
Probability of Directional Cross-Over	0.5
Probability of Selection	0.05
Probability of Mutation	0.1
DNA String Mutation Ratio	0.05
Elitism	Enabled
Constraint Handling	Penalize Objectives

(1) Population size of 200 individuals with 500 generations is also tried but a population size of 500 with 200 generations is found to produce better results.

Table 5.13: Algorithm parameters used for NSGA2 in benchmark simulations

Number of Individuals	500 (200$^{(1)}$)
Number of Generations	200 (500$^{(1)}$)
Crossover Probability	0.9
Mutation Probability (Real-valued Vectors)	1
Mutation Probability (Binary Strings)	1

(1) Population size of 200 individuals with 500 generations has also been evaluated but a population size of 500 with 200 generations is found to produce better results.

If more CFEs are allowed, FEPSO is capable of finding a good solution for the original problem most of the times with less than 100 OFEs and every time when it is allowed to make 500 OFEs according to the simulations conducted. FEPSO manages to obtain a candidate optimum in approximately $140\,000$ CFEs on average. When allowed to make approximately $200\,000$ CFEs and 500 OFEs, FEPSO consistently achieves to attain good solutions. Standard deviation of the best solutions found is only 1.25, which signifies the consistency of FEPSO.

Only disadvantage of FEPSO seems to be the relatively high number of CFEs required. Although the GTP and its variants considered in this benchmark has constraints and objective functions with similar complexity; as argued in Section 2.4, many problems of the mechanical engineering domain have relatively easy-to-calculate constraints if compared to objective functions involving complex computations such as the FEM or computational fluid dynamics. Moreover, there may be cases where violation of some constraints render calculation of objective- or other constraint functions impossible. It might be necessary to handle these constraints separately. Therefore, the FEPSO approach is useful and is even necessary in some cases. Corriveau, Guilbault, and Tahan [158] reported that FEM corresponded approximately to $95\,\%$ of total time elapsed during the processing of an individual in their study where they coupled GA and FEM. Muc and Gurba [159] use a function representing a failure criterion which is calculated by FEM as the objective function and constraints consisting of geometric parameters in optimization of composite structures. On the other hand, Parasiliti et al. [160] define a highly-constrained problem that has different kinds of constraints:

- Constraints that are easier to calculate

- Constraints that make calculation of some other constraint- and objective functions impossible when violated

FEPSO not only finds some feasible particles, but also improves the constraint satisfaction level of the whole swarm. *Mean number of constraint violations* (MNCV) (number of constraints that a particle violates) can be used as an indication of the level of constraint violation (or satisfaction) in the swarm. In essence, MNCV can be employed as a measure to gauge the swarm's feasibility level. MNCV of 50 simulations executed for the RC-GTP is plotted as a function of iteration number in Figure 5.3. As can be seen from the figure, FEPSO progressively improves the constraint satisfaction level of the swarm. This suggests that FEPSO might be successfully employed to preprocess a population for another algorithm which requires a better population in terms of constraint satisfaction.

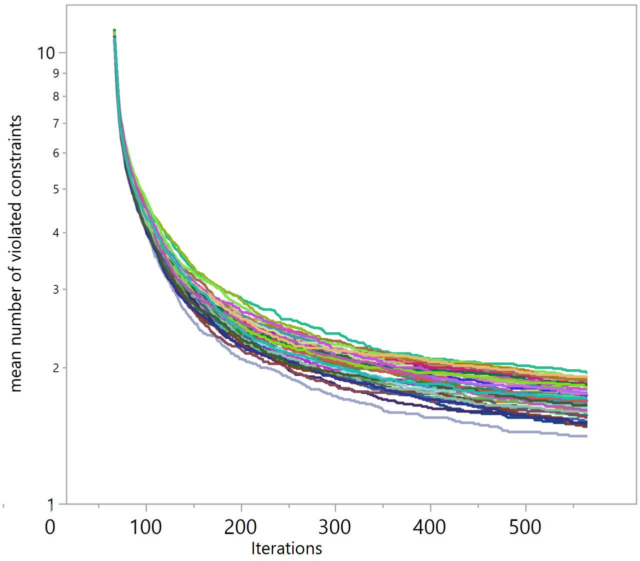


Figure 5.3: Mean number of constraint violations in the population during the RC-GTP runs.

5.2 Multi-objective four-stage gear train problem

The original GTP problem and its two variants were introduced in Section 5.1. The two variants are devised via combining several constraints (RC-GTP) and allowing several discrete decision variables to take real values (MV-GTP). In fact, the GTP can easily be handled as a multi-objective problem with some additional objectives. Dolen, Kaplan, and Seireg [98] introduced a multi-objective version of the GTP by adding another volume parameter as the second objective. Following this multi-objective approach and two new variants of the GTP, three multi-objective versions of the GTP are also introduced in this chapter:

- Multi-objective four-stage gear train problem (MOG)
- Reduced constraints multi-objective four-stage gear train problem (RC-MOG)
- Mixed valued multi-objective four-stage gear train problem (MV-MOG)

All three variants of the problem consist of the design of a four-stage gear train with two objectives:

- Minimize the weight (which is directly proportional to the total volume of all pinions and gears)
- Minimize the gearbox volume that encompasses all gears and pinions

Design variables of the problems are same as the single objective versions of the problems (see Section 5.1) and include the following items: the positions of the gears at each stage; pinion position in the first stage (other stages have their pinions positioned at the position of the previous stage's gear); the number of teeth of gears and pinion; gear thickness for each stage.

5.2.1 Problem definition for MOG, RC-MOG, and MV-MOG

All three variants of the problem can be defined as to *minimize*

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \equiv \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) \end{bmatrix}^{\mathbf{T}} \tag{5.39}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq M} \tag{5.40}$$

where $M = 86$ for MOG whereas $M = 62$ for RC-MOG and MV-MOG. Decision vector[2] can be defined as

$$\begin{aligned}\mathbf{x} =& [x_1 \quad \cdots \quad x_{22}]^T \\ =& [X_0 \quad Y_0 \quad \cdots \quad X_4 \quad Y_4 \quad b_1 \quad \cdots \quad b_4 \quad N_1^P \quad \cdots \quad N_4^P \quad N_1^G \quad \cdots \quad N_4^G]^T \end{aligned} \tag{5.41}$$

Similarly, objective functions can be written as follows:

$$y_1 = f_1(\mathbf{x}) = \pi \sum_{s=1}^{4} b_s (C_s)^2 \frac{(N_s^p)^2 + (N_s^g)^2}{(N_s^p + N_s^g)^2} \tag{5.42}$$

$$y_2 = f_2(\mathbf{x}) = L_x L_y \sum_{s=1}^{4} b_s \tag{5.43}$$

Decision variable properties of MOG and RC-MOG are same with GTP and RC-GTP whereas MV-MOG is similar to MV-GTP. While constraint functions of MOG are identical with GTP, RC-MOG as well as MV-MOG constraint functions are same as RC-GTP and MV-GTP. Decision variable- and constraint function similarities are summarized in Table 5.14.

Table 5.14: Similarities between decision variables- and constraint functions of single-objective and multi-objective variants of the GTP

Multi-objective Problem	Decision variable properties	Constraint functions
MOG	Same as GTP and RC-GTP (Table 5.1)	Same as GTP (Table 5.2)
RC-MOG	Same as GTP and RC-GTP (Table 5.1)	Same as RC-GTP and MV-GTP (Table 5.4)
MV-MOG	Same as MV-GTP (Table 5.5)	Same as RC-GTP and MV-GTP (Table 5.4)

All calculations explained for GTP in Section 5.1.1 are also valid for the multi-objective variants of the problem. Additionally, bounding box dimensions (L_x and L_y) used for

---

[2] Note that decision vector definition is similar to GTP (Eqn. 5.3).

the second objective function are defined as follows:

$$L_x = X^{max} - X^{min}, \quad L_y = Y^{max} - Y^{min} \tag{5.44}$$

$$X^{min} = \min\left(\min_s(X_{s-1} - r_s^{po}), \min_s(X_s - r_s^{go})\right) \tag{5.45}$$

$$X^{max} = \max\left(\max_s(X_{s-1} + r_s^{po}), \max_s(X_s + r_s^{go})\right) \tag{5.46}$$

$$Y^{min} = \min\left(\min_s(Y_{s-1} - r_s^{po}), \min_s(Y_s - r_s^{go})\right) \tag{5.47}$$

$$Y^{max} = \max\left(\max_s(Y_{s-1} + r_s^{po}), \max_s(Y_s + r_s^{go})\right) \tag{5.48}$$

where $s \in \{1, 2, 3, 4\}$ as before.

5.2.2 Results and discussion

To evaluate the performance of MOFEPSO, the method was run 50 times for each variant of MOG. Algorithm parameters were kept same with previous FEPSO runs (see Table 5.6). Briefly, the swarm size (I) was set to 200 while the maximum number of iterations was limited to 500. Note that the inertia factor W has been varied from 0.9 (in the beginning) to 0.4 (at the end) throughout the execution. Acceleration coefficients and the tolerance parameter were selected as $C_1 = C_2 = 2$, $x^{tol} = 0.01$.

Since MOFEPSO may perform many *constraint function evaluations* (CFE) during virtual boundary search, actual number of CFEs needed to be recorded. Similarly, *objective function evaluations* (OFE) that are only performed at feasible positions were also saved to analyze the performance of the proposed method.

Popular multi-objective algorithms MOGA2 [145], NSGA2 [58], and MOPSO [68] were also employed to solve all three problems so as to make a quantitative comparison for MOFEPSO. NSGA2 is one of the most popular algorithms used to compare PSO based multi-objective algorithms [161–163]. MOGA2 and NSGA2 simulations were carried out in modeFRONTIER® software package with options summarized in Tables 5.12 and 5.13 respectively. Note that MOPSO simulations were run in MATLAB® with the options given in Table 5.15. Other than these simulations, the results presented by Dolen, Kaplan, and Seireg [98] were also used for comparison of MOG solutions found by MOFEPSO.

Table 5.15: Algorithm parameters used for MOPSO

W	0.4
C_1	2
C_2	2
Swarm Size (I)	250
Mutation Rate	0.5
Number of Grids (N_{grid})	20
Max. Number of Iterations	1000

Summary of MOFEPSO runs for MOG, RC-MOG, and MV-MOG along with benchmark simulations are tabulated in Tables 5.16, 5.17, and 5.18. Note that all MOFEPSO runs performed on each problem variant were successful in finding feasible solutions[3].

Considering the mean values (μ) of objective functions, MOFEPSO apparently yields the best mean for both objectives in all problem variants. When objectives are examined separately, it can be seen that MOFEPSO has obtained the overall minimum for both objectives except that the result presented by Dolen, Kaplan, and Seireg [98] for the first objective (i.e. y_1) is slightly better. Moreover, the number of OFEs in MOFEPSO is significantly lower than all other approaches.

Since Pareto optimality based multi-objective algorithms provide multiple Pareto solutions at each run, many solution points could be generated by MOFEPSO and its contenders. Some selected Pareto solutions are presented in Table 5.19. Decision vectors for Pareto solutions (labeled in Table 5.19 as best MOFEPSO solutions) for each of the three problem variants are also given in Table 5.20. Figure 5.4 illustrates the results side by side along with their single-objective counterparts. In the illustrations the single objective versions of MOG, RC-MOG, and MV-MOG are labeled as GTP, RC-GTP, and MV-GTP respectively. Detailed results of single-objective versions are presented in Section 5.1. However, it is worth mentioning here that $y_1 = 36.25\,\mathrm{cm}^3$ for GTP and RC-GTP; $y_1 = 35.23\,\mathrm{cm}^3$ for MV-GTP.

It can be observed from Figure 5.4 that the second objective (y_2) ensures compactness

---

[3] This indicates a success rate of $100\,\%$

Table 5.16: Comparison of results obtained for MOG

					Gear volume - y_1 (cm^3)				Gearbox volume - y_2 (cm^3)			
Algorithm	$\mathrm{CFE}_{\max}$	$\mathrm{OFE}_{\max}$	Runs	Succ.	Best	Worst	μ	σ	Best	Worst	μ	σ
Dolen[(1)]	10 000	10 000	100	60	35.40	-	39.78	-	85.20	-	137.52	-
Dolen[(2)]	10 000	10 000	100	0	-	-	-	-	-	-	-	-
MOGA2	250 000	250 000	5	5	37.36	72.54	49.88	8.61	110.25	206.38	146.41	24.37
NSGA2	250 000	250 000	5	3	45.52	83.61	66.76	15.45	113.12	246.93	197.22	44.98
MOPSO	250 000	250 000	10	9	43.54	132.86	67.49	29.46	112.18	333.77	193.76	71.78
MOFEPSO	250 000	1000	50	50	36.25	49.43	39.14	2.22	84.53	149.62	113.53	12.24

(1) Applies solution space reduction [98]

(2) No solution space reduction [98]

Table 5.17: Comparison of results obtained for RC-MOG

					Gear volume - y_1 (cm^3)				Gearbox volume - y_2 (cm^3)			
Algorithm	$\mathrm{CFE}_{\mathrm{max}}$	$\mathrm{OFE}_{\mathrm{max}}$	Runs	Succ.	Best	Worst	μ	σ	Best	Worst	μ	σ
MOGA2	250 000	250 000	5	5	40.86	48.89	43.72	1.64	109.02	133.84	116.45	4.61
NSGA2	250 000	250 000	5	4	39.86	83.77	65.13	16.79	119.93	229.05	194.21	46.80
MOPSO	250 000	250 000	10	2	53.40	57.50	53.95	1.44	148.92	172.41	166.50	10.59
MOFEPSO	270 000	1000	50	50	36.25	44.45	39.35	1.67	85.16	152.18	115.41	11.28

Table 5.18: Comparison of results obtained for MV-MOG

					Gear volume - y_1 (cm^3)				Gearbox volume - y_2 (cm^3)			
Algorithm	$\mathrm{CFE}_{\mathrm{max}}$	$\mathrm{OFE}_{\mathrm{max}}$	Runs	Succ.	Best	Worst	μ	σ	Best	Worst	μ	σ
MOGA2	250 000	250 000	5	5	35.46	50.87	41.72	6.62	81.87	140.56	107.55	18.67
NSGA2	250 000	250 000	5	5	35.96	90.14	53.50	19.38	105.31	242.30	149.85	49.18
MOPSO	250 000	250 000	10	10	35.64	85.87	58.93	19.50	95.80	275.25	180.91	71.86
MOFEPSO	250 000	2500	50	50	35.13	44.89	37.04	2.50	74.83	137.21	101.22	15.50

of the gear train as intended. In the absence of y_2, the gears are free to assume any position without affecting the value of y_1. Notice that the presence of y_2 encourages repetitive use of same positions in successive gear train stages.

Progression of Pareto fronts throughout MOFEPSO runs are plotted in Figure 5.5. More solutions are obtained in MV-MOG due to the relaxation provided by unrestricted real-valued variables.

5.2.2.1 Statistical evaluation of simulation results

Pareto solutions found by all algorithms were used to statistically compare the MOFEPSO results with those of the others (see Figure 5.6[4]). Note that each objective was treated separately and that nonparametric comparisons were performed with Wilcoxon method [164] (Table 5.21). As can be seen from the Table 5.21, MOFEPSO yields significantly better results for both objectives if compared to other algorithms with only one exception (i.e. y_2 of MOGA2).

As in FEPSO, the only disadvantage of MOFEPSO seems to be the high number of CFEs due to its unique constraint handling mechanism. However, MOFEPSO only performs OFEs at feasible points. Therefore, high number of CFEs limit OFEs in highly constrained problems. This issue might be especially advantageous when objective functions are computationally more expensive.

5.2.2.2 Exploration capability

Rate of congruence for pinions and gears at each stage can be calculated to illustrate the distribution of Pareto solutions on the design space. It is critical to note that the rate of congruence is a function defining the rate for a position at any stage to be occupied by a specific type of gear among all Pareto solutions of MOFEPSO during simulations (50 runs). Rate of congruence of an arbitrary position for pinions and gears at stage s

---

[4] Note that since the MOG and its variants are minimization problems, lower values for both y_1 and y_2 are better.

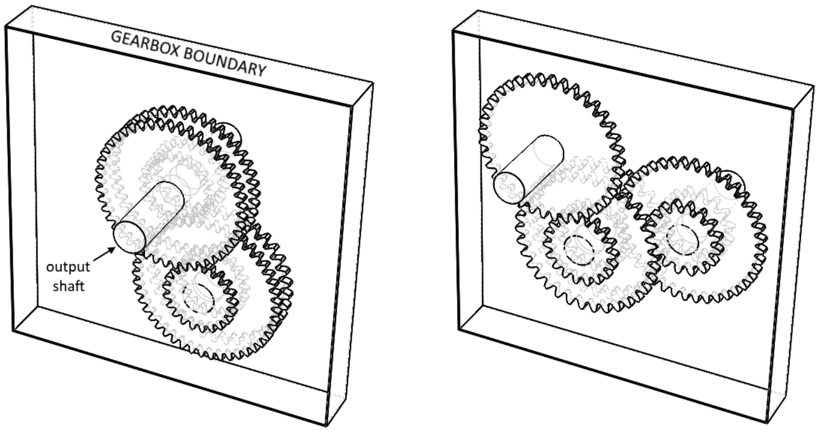
Table 5.19: Some selected Pareto solutions

	MOG		RC-MOG		MV-MOG	
Algorithm	y_1	y_2	y_1	y_2	y_1	y_2
MOGA2	37.36	123.34	45.22	109.02	35.46	81.88
	39.10	110.75	40.86	111.07	35.47	81.87
	39.12	110.42				
NSGA2	52.56	113.12	39.86	131.82	35.96	105.80
	45.57	161.85	40.69	119.93	36.59	105.31
MOPSO	43.54	132.82	53.40	172.41	35.64	97.49
	52.62	112.18	53.70	149.78	36.29	95.80
MOFEPSO(1)	36.26	85.16	36.25	131.14	35.13	79.50
	36.26	99.85	**36.26**	**85.16**	35.14	79.48
	36.26	**84.60**	36.26	115.76	**35.15**	**75.41**
	36.27	101.00	36.27	115.53	35.38	74.83
	36.28	84.53	39.03	94.13	35.38	74.84
	36.26	85.16	37.41	95.14	35.33	74.87
	36.29	85.16	38.18	95.34	35.27	74.89
	38.96	93.45	37.44	95.66		

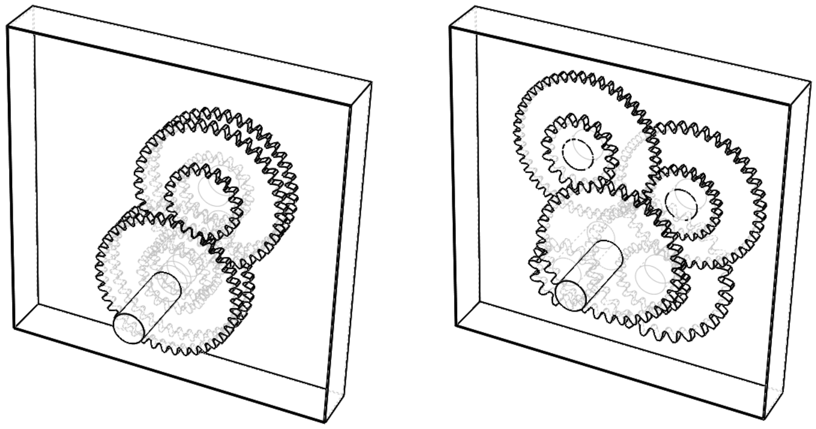
(1) Marked solutions are given in Table 5.20 and illustrated in Figure 5.4

Table 5.20: Decision vectors and objective values for some selected Pareto solutions found by MOFEPSO

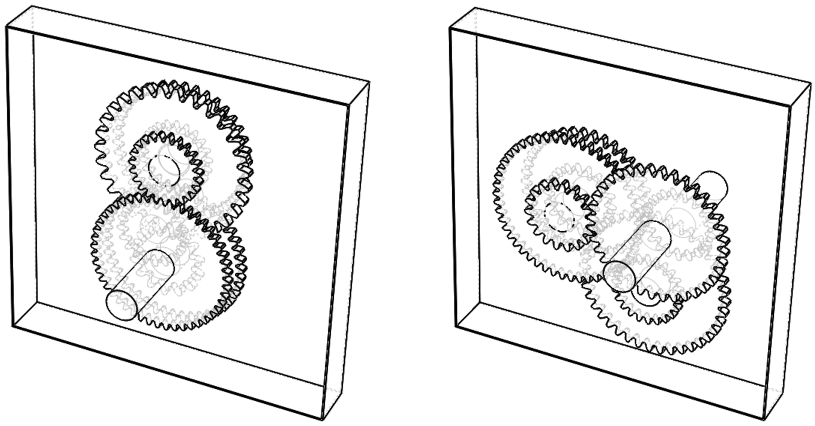
	MOG	RC-MOG	MV-MOG
X_0	63.5	63.5	50.3283
Y_0	76.2	38.1	47.9472
X_1	76.2	76.2	60.8033
Y_1	38.1	76.2	84.5796
X_2	63.5	63.5	61.1880
Y_2	76.2	38.1	44.6666
X_3	76.2	76.2	62.1840
Y_3	38.1	76.2	85.4760
X_4	63.5	63.5	61.1601
Y_4	76.2	38.1	46.2847
b_1	3.175	3.175	3.175
b_2	3.175	3.175	3.175
b_3	3.175	3.175	3.175
b_4	3.175	3.175	3.175
N_1^g	36	41	38
N_2^g	45	41	45
N_3^g	45	38	31
N_4^g	45	40	48
N_1^p	18	20	19
N_2^p	21	19	21
N_3^p	21	18	14
N_4^p	21	19	23
y_1	36.26	36.26	35.15
y_2	84.60	85.16	75.41



(a) MOG solution by MOFEPSO (b) GTP solution by FEPSO

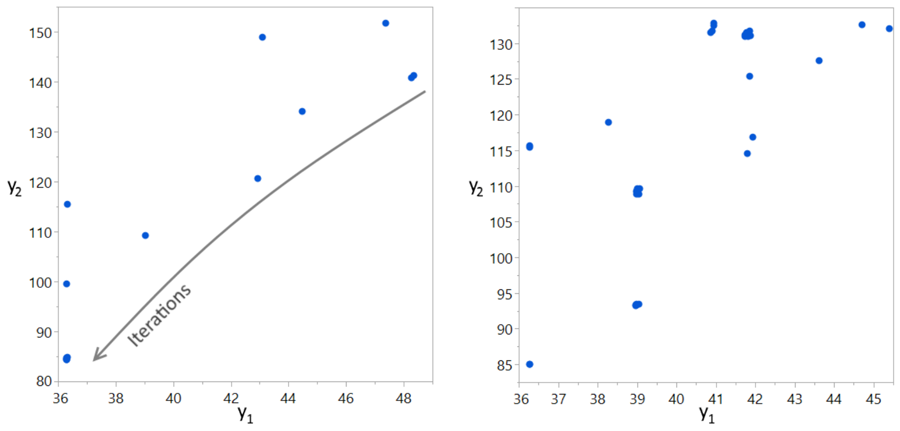


(c) RC-MOG solution by MOFEPSO (d) RC-GTP solution by FEPSO

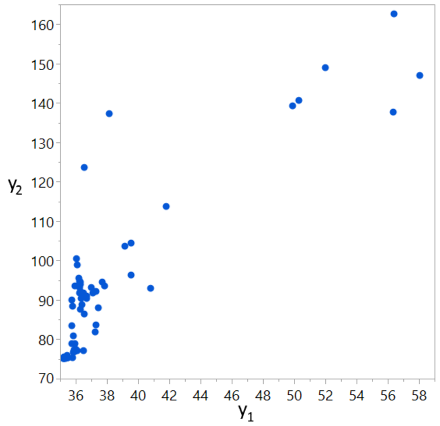


(e) MV-MOG solution by MOFEPSO (f) MV-GTP solution by FEPSO

Figure 5.4: Illustrations of some selected Pareto solutions found by MOFEPSO and their comparison to single-objective solutions found by FEPSO



(a) MOG (b) RC-MOG



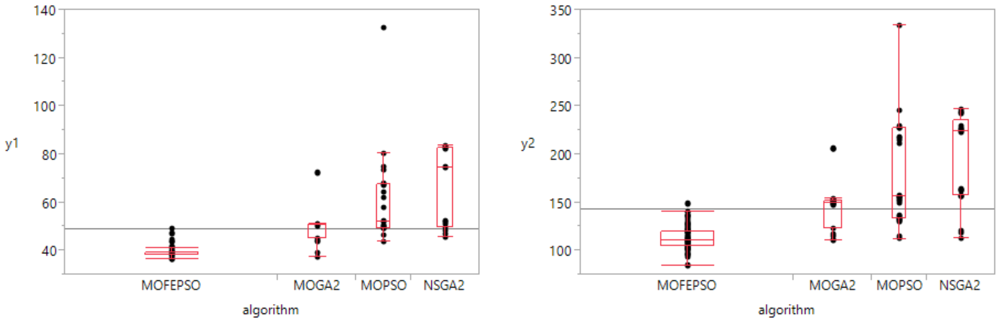
(c) MV-MOG

Figure 5.5: Progression of Pareto solutions along iterations of selected MOFEPSO runs

Table 5.21: Statistical results (p) for nonparametric comparisons of MOGA2, MOPSO, and NSGA2 Pareto solutions with MOFEPSO Pareto solutions using Wilcoxon method

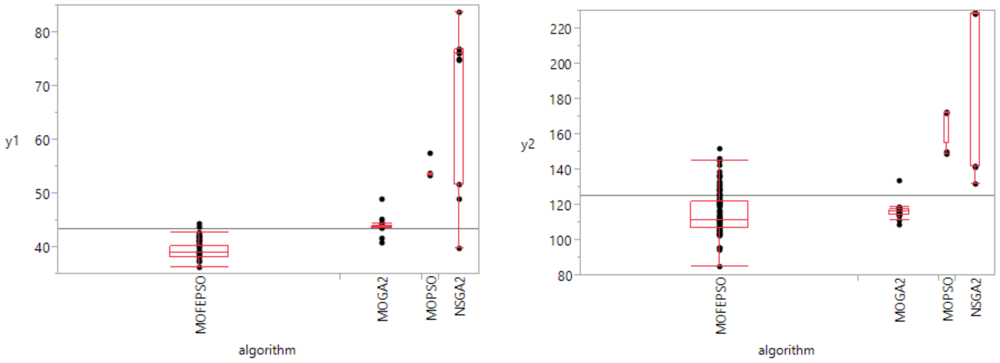
	MOG		RC-MOG		MV-MOG	
Algorithm	y_1	y_2	y_1	y_2	y_1	y_2
MOGA2	< 0.0001	< 0.0001	< 0.0001	0.0837	< 0.0001	0.035
MOPSO	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001
NSGA2	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

H_0: No difference between MOFEPSO and the listed algorithm. $p < 0.05$ rejects H_0



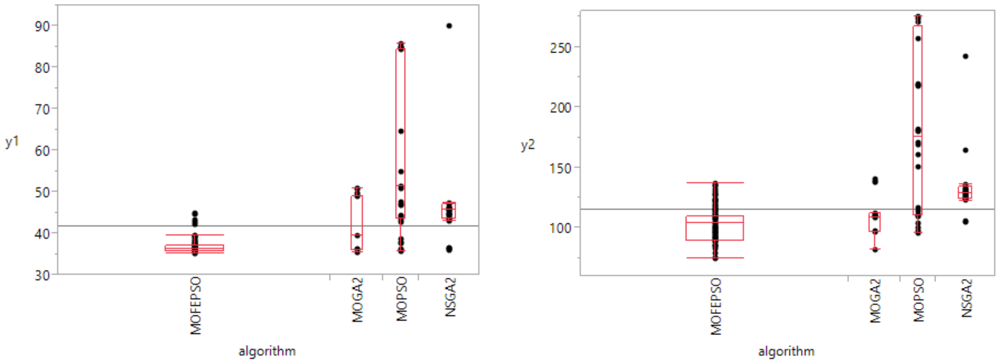
(a) y_1, MOG

(b) y_2, MOG



(c) y_1, RC-MOG

(d) y_2, RC-MOG



(e) y_1, MV-MOG

(f) y_2, MV-MOG

Figure 5.6: Graphical comparison of Pareto solutions obtained by different algorithms for MOG, RC-MOG, and MV-MOG

can be defined as follows:

$$\rho^\star(x,y,s) \triangleq \frac{\sum\limits_{\phi=1}^{|\boldsymbol{\Upsilon}|} \xi_\phi^\star(x,y,s)}{|\boldsymbol{\Upsilon}|} \tag{5.49}$$

where $\star$ is a placeholder for the letters p (pinion) and g (gear); $\boldsymbol{\Upsilon}$ denotes the union of all final non-dominated sets (i.e. Pareto solutions, $\mathbf{B}_i$) found throughout simulation runs:

$$\boldsymbol{\Upsilon} = \bigcup_{i=1}^{50} \mathbf{B}_i \triangleq \left\{ \boldsymbol{v}_\phi = \left(\boldsymbol{v}_\phi^x, \boldsymbol{v}_\phi^y\right) : \phi \in \mathbb{N}_{>0}^{\leq|\boldsymbol{\Upsilon}|} \right\} \tag{5.50}$$

Congruence functions in Eqn. (5.49) can be defined as follows:

$$\xi_\phi^\star(x,y,s) \triangleq \begin{cases} 1, & \sqrt{(x-\bar{X}_{s-\sigma})^2 + (y-\bar{Y}_{s-\sigma})^2} \leq \dfrac{\bar{D}_s^\star}{2} \\ 0, & \text{otherwise} \end{cases} \tag{5.51}$$

where $\sigma = 1$ for pinions ($\star = p$) while $\sigma = 0$ for gears ($\star = g$). Similarly,

$$\boldsymbol{v}_\phi^x = [\bar{X}_0 \quad \bar{Y}_0 \quad \cdots \quad \bar{X}_4 \quad \bar{Y}_4 \quad \bar{b}_1 \quad \cdots \quad \bar{b}_4 \quad \bar{N}_1^P \quad \cdots \quad \bar{N}_4^P \quad \bar{N}_1^G \quad \cdots \quad \bar{N}_4^G]^T \tag{5.52}$$

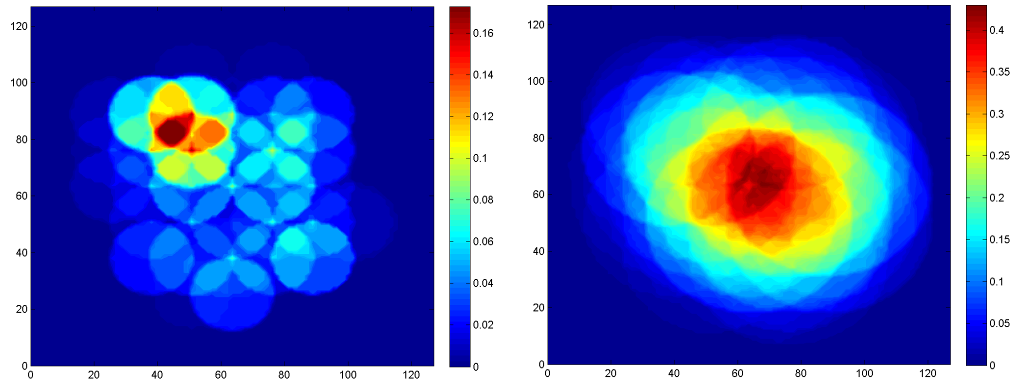
$$\boldsymbol{v}_\phi^y = \boldsymbol{f}(\boldsymbol{v}_\phi^x) \tag{5.53}$$

Definitions for $\bar{D}_s^p$ and $\bar{D}_s^g$ can be found in Eqns. 5.12 and 5.13.

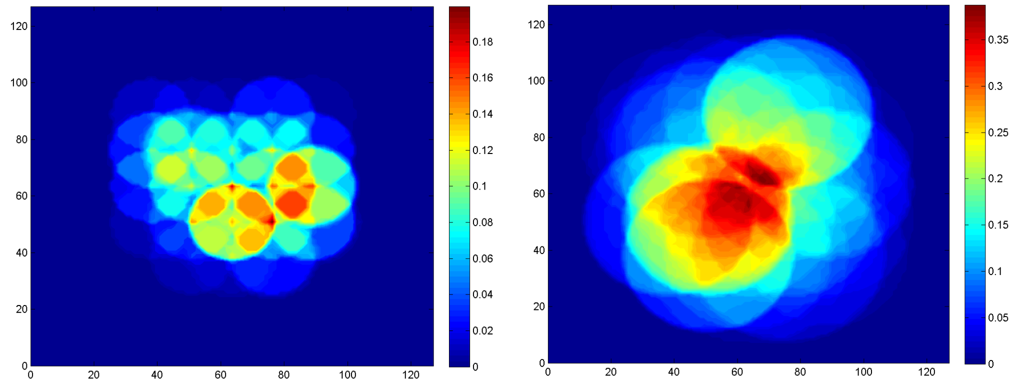
Congruence rates of first two stages for MOG Pareto solutions are shown in Figure 5.7. As seen in the figure, gear positions in different Pareto solutions are distributed among available locations. Hence, MOFEPSO can be said to have explored a wide portion of the search space.

5.3 Closure

A very highly constrained mechanical design problem (GTP) is addressed in this chapter. Two distinct versions of the problem which introduce different features are defined (RC-GTP and MV-GTP). FEPSO was applied to all three versions (including the original GTP) of the problem to assess its performance. Other algorithms (DOPSO, HPSO, GA, MOGA2, and NSGA2) were also employed where applicable for benchmarking purposes. Additionally, results for the GTP available in the literature [86, 93–98, 155, 156] were also compared with results obtained by FEPSO simulations.



(a) pinion, $s = 1$ (b) gear, $s = 1$



(c) pinion, $s = 2$ (d) gear, $s = 2$

Figure 5.7: Rates of congruence at first two stages for MOG Pareto solutions found by MOFEPSO

Based on the simulations and comparisons, following points related with FEPSO were observed:

- FEPSO is well suited for highly-constrained problems where feasible region of the search space is relatively small. FESPO was able to perform with a high success rate where other algorithms failed to consistently provide a solution.
- Due to the fact that FEPSO requires relatively less OFEs, it could serve well when the problem has constraints that are relatively easier to evaluate in comparison to objective functions (or other constraints[5]).
- FEPSO was observed to continuously improve the feasibility of the swarm. Therefore, FEPSO can also be utilized to improve solely the constraint satisfaction level of the swarm (i.e. to reduce the mean number of violated constraints). This feature might be useful to preprocess the population for another algorithm that needs a better population in terms of constraint satisfaction.

In the second part of this chapter, the multi-objective forms of the four-stage gear train problems are introduced: MOG, RC-MOG, and MV-MOG. These problems share common properties with their single-objective counterparts with addition of an objective function that addresses the total size of the gearbox encompassing all pinions and gears of all stages.

MOFEPSO, MOGA2 [145], NSGA2 [58], and MOPSO [68] were utilized to solve all three multi-objective problems for benchmarking purposes. In addition to remarks related with FEPSO's ability in constraint handling, Following results regarding MOFEPSO were observed:

- MOFEPSO was found to perform better than the approaches it was benchmarked with both in terms of objective values of the solutions and success rate.
- Techniques developed for handling highly constrained problems were found to be relevant and effective also in the multi-objective context.

---

[5] A case where separate sets of constraints with differing complexities exist has not been demonstrated in this study. However, FEPSO could easily be modified to handle such problems. It would only require additional internal Virtual Boundary Search routines to be implemented.

- Design space exploration capability of the algorithm was found to be sufficient even in the existence of discrete decision variables.

CHAPTER 6

HEAT PIPE DESIGN PROBLEM

This chapter presents an application of FEPSO to design of a mesh wick type heat pipe. The design problem consists of minimization of heat pipe mass with respect to eight decision variables including geometric properties, mesh wick attributes, and working fluid type along with constraints ensuring heat pipe performance at multiple operating conditions. Performance of FEPSO in this problem is comparatively evaluated against NSGA2. Random design space exploration for this problem may reveal some feasible solutions. Hence, this design problem can not be categorized as a highly constrained problem. The aim of this application is to investigate FEPSO behavior in a less constrained problem of the mechanical engineering domain.

Heat pipes are thermal devices that transfer heat at a high transfer rate over long distances with low temperature gradient. A heat pipe is basically a hermetically sealed tube with a porous structure (called wick) placed on its inner walls (perimeter). It is filled with a working fluid which is evaporated at the evaporator side and flows through the center of the tube towards the condenser side when the heat pipe is operated. Simultaneously, the condensed liquid moves through the wick from the condenser side towards the evaporator side due to capillary forces. Even with a low temperature gradient between the evaporator and the condenser sides, heat pipes can transfer great amount of heat.

Some researchers have worked on optimization of heat pipes in earlier papers. One of the first efforts for design optimization of heat pipes was due to Buksa and Williams [165]. They developed an integrated analytical tool for use in designing optimized space-based heat pipe radiator systems.

Wu et al. [166] optimized L-ratio (i.e. ratio of the evaporator section length to the condenser section length) of a micro heat pipe heat sink. Liang and Hung [167] also optimized the L-ratio of the U-shape heat pipe using thermal resistance function as the objective function. Kim, Seo, and Do [168] performed thermal optimization of a miniature heat pipe through a mathematical model they derived. Vlassov, Sousa, and Takahashi [169] and Sousa, Vlassov, and Ramos [170, 171] optimized the heat pipe mass for space applications considering several different working fluids. They repeated the optimization procedure at several different operating conditions characterized by temperature and heat transfer rates.

Jeong, Kobayashi, and Yoshimura [172] carried out multi-objective optimization of mass and thermal conductance of a satellite heat pipe. Zhang et al. [173] utilized genetic algorithm to optimize the total thermal resistance of a heat pipe using geometric parameters as decision variables. Kiseev, Vlassov, and Muraoka [174, 175] used extensive experimental data to analyze and optimize capillary heat pipe structures.

Rao, Savsani, and Vakharia [139] maximized heat transfer rate while minimizing resistance of a heat pipe using a novel optimization algorithm. Maheshkumar and Muraleedharan [176] minimized entropy generation in a flat heat pipe. Roper [177] modeled and optimized a sandwich panel heat pipe for density, compressive modulus, compressive strength, and maximum heat flux. Wan, Wang, and Tang [178] employed finite element method for the condenser section optimization of a loop heat pipe.

Similar to several other multi-objective approaches in the literature, Rao and More [179] also used two objectives (mass and thermal resistance) but employed a weighted aggregation function instead of directly applying a multi-objective algorithm. Jokar et al. [180] employed a surrogate assisted approach through artificial neural networks and used genetic algorithm to optimize the operating point of a pulsating heat pipe. Song et al. [181] optimized the exergy efficiency of a heat pipe used in solar dynamic space power system. Patel [182] investigated a satellite heat pipe operated with ammonia and methanol for the multi-objective optimization of weight and thermal resistance. Lurie, Rabinskiy, and Solyaev [183] presented a topology optimization approach proposed to determine an optimal geometry of a wick sintered inside a flat plate heat pipe.

There are not many studies incorporating advanced optimization techniques. Only a few papers related with truly multi-objective optimization of heat pipes exist. These multi-objective approaches generally use the mass and the thermal resistance (or thermal conductivity) of the heat pipe as objective functions. For this study, design of a heat pipe with a wick formed by wire screen meshes for satellite thermal control is considered. Both single-objective and multi-objective approaches are investigated. The heat pipe mass is minimized in both approaches. Unlike other studies in the literature, the operating range (in terms of maximum heat transfer rate that can be achieved) is used as the second objective to be optimized (i.e. maximized). It is critical to note that the design- and optimization methodology presented here can easily be adapted for other types of heat pipes.

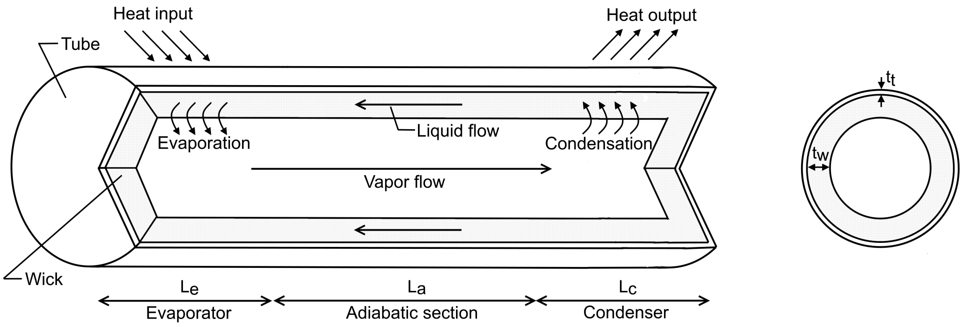


Figure 6.1: Heat pipe

6.1 Single-objective heat pipe design problem

6.1.1 Problem definition

This section describes a methodology for heat pipe design for a given operating condition defined by the heat transfer rate (Q) and heat sink temperature (T_{si}). The approach originates from Rajesh and Ravindran [184]. It is adapted as a heat pipe design problem for satellite thermal management by Sousa, Vlassov, and Ramos [170, 171]. It is also revisited by several recent optimization studies [185, 186]. The objective of the design problem is to minimize total mass of the heat pipe (m_{tot}) while satisfying the constraints. Therefore, the problem can be formulated as to minimize

$$y = f(\mathbf{x}) = m_{tot} \tag{6.1}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq M} \tag{6.2}$$

where $M = 14$ for the operating condition of interest and

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_8 \end{bmatrix}^T \tag{6.3}$$

$$= \begin{bmatrix} N & \varepsilon & d_v & t_w & L_e & L_c & t_t & M_f \end{bmatrix}^T \tag{6.4}$$

The elements of the decision vector $\mathbf{x}$ are defined in Table 6.1.

Table 6.1: Decision variables

Variable	Description	Values
N	Wick mesh number	$314 \leq N \in \mathbb{N} \leq 15\,000$
ε	Porosity	$0.0001 \leq \varepsilon \in \mathbb{R} \leq 0.9999$
d_v	Vapor core diameter [m]	$5 \times 10^{-3} \leq d_v \in \mathbb{R} \leq 80 \times 10^{-3}$
t_w	Wick thickness [m]	$0.05 \times 10^{-3} \leq t_w \in \mathbb{R} \leq 10 \times 10^{-3}$
L_e	Evaporator length [m]	$50 \times 10^{-3} \leq L_e \in \mathbb{R} \leq 400 \times 10^{-3}$
L_c	Condenser length [m]	$50 \times 10^{-3} \leq L_c \in \mathbb{R} \leq 400 \times 10^{-3}$
t_t	Container wall thickness [m]	$0.3 \times 10^{-3} \leq t_t \in \mathbb{R} \leq 3 \times 10^{-3}$
M_f	Working fluid	$M_f \in \{\mathrm{ethanol}, \mathrm{methanol}, \mathrm{ammonia}\}$

Many of the calculations required to obtain objective functions and constraints are dependent on the vapor temperature (T_v). Therefore, before elaborating the procedure, calculation of T_v will be explained. When thermal resistances associated with various elements of the heat pipe are considered, all “axial” resistances except for that of (axially flowing) vapor can be presumed to be high so that the resulting thermal circuit could be idealized as an open one [187]. Furthermore, axial resistance of the vapor and liquid-vapor interface resistances can be neglected since they are relatively low [187]. Hence, a simplified model of thermal resistances between the evaporator and condenser sections will only include thermal resistance of the heat pipe wall at evaporator and condenser sections (R_{te} and R_{tc} respectively) as well as thermal resistance of the wick (including that of the liquid) at evaporator and condenser sections (R_{we} and R_{wc} respectively) as shown in Figure 6.2. That is,

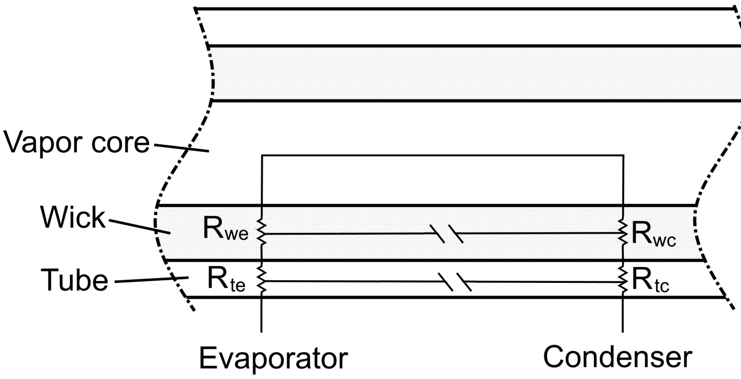


Figure 6.2: Thermal resistance model

$$R_{te} = \frac{\ln(d_o/d_i)}{2\pi L_e k_t} \tag{6.5}$$

$$R_{tc} = \frac{\ln(d_o/d_i)}{2\pi L_c k_t} \tag{6.6}$$

where $d_i = d_v + 2t_w$ is the inner diameter, $d_o = d_i + 2t_t$ is the outer diameter of the heat pipe, and k_t is the thermal conductivity of the heat pipe tube material. Note that tube and wick materials are assumed to be made from SAE 304 stainless steel. Similarly,

$$R_{we} = \frac{\ln(d_i/d_v)}{2\pi L_e k_{eq}} \tag{6.7}$$

$$R_{wc} = \frac{\ln(d_i/d_v)}{2\pi L_c k_{eq}} \tag{6.8}$$

The equivalent thermal conductivity (k_{eq}) in Eqn. 6.7 and 6.8 is defined as

$$k_{eq} = \frac{k_l\left[(k_l + k_w) - (1-\varepsilon)(k_l - k_w)\right]}{(k_l + k_w) + (1-\varepsilon)(k_l - k_w)} \tag{6.9}$$

Here, k_l and k_w denote the thermal conductivities of the liquid and wick material respectively. Note that all fluid properties (for both liquid and vapor) depend on T_v itself. Therefore, despite its apparent simplicity, the calculation of Eqn. 6.10

$$T_v = T_{si} + (R_{tc} + R_{wc})Q \tag{6.10}$$

embodies an iterative procedure as follows:

1. Make an initial guess: $T_v^* := T_{si}$

2. Obtain fluid properties k_l and k_w at T_v^* to calculate k_{eq}, R_{we}, and R_{wc} using Eqns. 6.7, 6.8, and 6.9.
3. Calculate $T_v = T_{si} + (R_{tc} + R_{wc})Q$. Note that the condenser temperature is assumed to be equal to T_{si}.
4. If $|T_v - T_v^*| \geq T^{tol}$ then set $T_v^* := T_v$ and return to step 2.

$T^{tol} = 0.01\ K$ is the convergence tolerance. Following fluid properties are also calculated at T_v: latent heat of vaporization (λ), liquid density (ρ_l), vapor density (ρ_v), liquid thermal conductivity (k_l), liquid viscosity (μ_l), vapor viscosity (μ_v), vapor pressure (P_v), and liquid surface tension (σ). Note that all material- and fluid properties are tabulated in Appendix A[1].

The objective function, total mass of the heat pipe, can be expressed as

$$f(\mathbf{x}) = m_{tot} = m_t + m_{wd} + m_l + m_v \tag{6.11}$$

where m_t, m_{wd}, m_l, and m_v are the masses of the tube, dry wick, the liquid in the wick, and the vapor in the core respectively. In Eqn. 6.11,

$$m_t = \pi t_t \left(d_i + t_t\right) L_{tot} \rho_t \tag{6.12}$$

$$m_{wd} = \pi t_w \left(d_v + t_w\right) \left(1 - \varepsilon\right) L_{tot} \rho_w \tag{6.13}$$

where $L_{tot} = L_a + L_e + L_c$ is the total length of the heat pipe, ρ_t is the tube material density, ρ_w is the wick material density, and $L_a = 0.5\,\mathrm{m}$ is the length of the adiabatic section. Note that when a specific requirement related with the total length of the heat pipe exists, it is also possible to take L_a as a decision variable and define a constraint function for L_{tot}. However, if the distance between evaporator- and condenser sections is fixed by design [170, 171, 186], adiabatic section length is constant as explained above.

$$m_l = \pi t_w \left(d_v + t_w\right) \varepsilon L_{tot} \rho_l \tag{6.14}$$

$$m_v = \frac{\pi d_v^2 \rho_v L_{tot}}{4} \tag{6.15}$$

Due to the capillary limit

$$g_1(\mathbf{x}) = Q - Q_c \leq 0 \tag{6.16}$$

---

[1] Note that working fluid properties are obtained by interpolating values given in property tables with vapor temperature (T_v).

The capillary heat transfer rate limit Q_c is defined as

$$Q_c = \frac{P_c + P_g}{\left(F_l + F_v\right) L_{eff}} \tag{6.17}$$

where P_c is the capillary pressure drop, P_g is the gravitational pressure difference, F_l and F_v are the liquid and vapor friction coefficients respectively. $L_{eff} = \frac{L_e + L_c}{2} + L_a$ is the effective length.

$$P_c = \frac{2\sigma}{r_c} \tag{6.18}$$

where capillary radius $r_c = {}^1/_{2N}$. Similarly,

$$P_g = \rho_l g \left(L_{tot} \sin\alpha - d_v \cos\alpha\right) \tag{6.19}$$

where α is the angle of inclination. Note that for satellite applications $P_g = 0$.

$$F_l = \frac{\mu_l}{K \left(\pi \frac{d_i^2 - d_v^2}{4}\right) \rho_l \lambda} \tag{6.20}$$

where K is permeability which is defined as follows:

$$K = \frac{d^2 \varepsilon^3}{122 (1 - \varepsilon)^2} \tag{6.21}$$

$$F_v = \frac{128 \mu_v}{\pi d_v^4 \rho_v \lambda} \tag{6.22}$$

Due to the operational temperature range of electronic equipment, limits on the source temperature (T_{so}) exist, such that

$$T_{so}^{\min} \leq T_{so} \leq T_{so}^{max} \tag{6.23}$$

$$g_2(\mathbf{x}) = T_{so}^{min} - T_{so} \leq 0 \tag{6.24}$$

$$g_3(\mathbf{x}) = T_{so} - T_{so}^{\max} \leq 0 \tag{6.25}$$

where

$$T_{so} = \left(R_{te} + R_{tc} + R_{we} + R_{wc}\right) Q + T_{si} \tag{6.26}$$

The working fluid shall not reach the boiling point such that

$$g_4(\mathbf{x}) = Q - Q_b \leq 0 \tag{6.27}$$

$$Q_b = \frac{2\pi L_e k_{eq} T_v}{\lambda \rho_v \ln\left(\frac{d_i}{d_v}\right)} \left(\frac{2\sigma}{r_n} - P_c\right) \tag{6.28}$$

where $r_n = 2.5 \times 10^{-7}\,\mathrm{m}$ is the nucleation radius.

High vapor velocities may entrain the liquid returning to the evaporator. Entrainment limit may especially become significant at low mesh numbers [184]. Therefore,

$$g_5(\mathbf{x}) = Q - Q_e \leq 0 \tag{6.29}$$

$$Q_e = \frac{\pi d_v^2}{4}\lambda\sqrt{\frac{\sigma\rho_v}{2r_{hs}}} \tag{6.30}$$

where hydraulic radius (r_{hs}) can be defined as

$$r_{hs} = \frac{1}{2N} - \frac{d}{2} \tag{6.31}$$

For Eqn. 6.30 to be valid,

$$g_6(\mathbf{x}) = d - \frac{1}{N} \leq 0 \tag{6.32}$$

must be satisfied. The wick wire diameter (d) can be calculated using mesh number and porosity such that

$$d = 4\frac{1-\varepsilon}{1.05\pi N} \tag{6.33}$$

At lower than optimal operating temperatures, saturation vapor pressure in the heat pipe may become comparable to the pressure drop required for the vapor to flow towards the condenser. As a consequence, the following constraint called the viscous limit shall be satisfied:

$$g_7(\mathbf{x}) = Q - Q_v \leq 0 \tag{6.34}$$

$$Q_v = \frac{\pi d_v^4 \rho_v \lambda P_v}{256\mu_v L_{eff}} \tag{6.35}$$

Furthermore, Since Eqns. 6.20 and 6.22 are only valid for laminar and incompressible flow, limits on Mach number and Reynolds number must be imposed:

$$g_8(\mathbf{x}) = M_v - 0.2 \leq 0 \tag{6.36}$$

$$g_9(\mathbf{x}) = \mathrm{Re}_v - 2300 \leq 0 \tag{6.37}$$

$$M_v = \frac{4Q}{\pi\rho_v d_v^2 \lambda\sqrt{\gamma R_v T_v}} \tag{6.38}$$

$$\mathrm{Re}_v = \frac{4Q}{\pi d_v \mu_v \lambda} \tag{6.39}$$

where γ is the specific heat ratio of the working fluid and R_v is the specific gas constant[2].

---

[2] Note that $R_v = R/M$ where $R = 8.3144598$ J/molK is the gas constant and M is the molar mass.

Wick wire diameter shall also be constrained such that

$$0.025 \times 10^{-3}\,\mathrm{m} \leq d \leq 1 \times 10^{-3}\,\mathrm{m} \tag{6.40}$$

$$g_{10}(\mathbf{x}) = 0.025 \times 10^{-3} - d \tag{6.41}$$

$$g_{11}(\mathbf{x}) = d - 1 \times 10^{-3} \tag{6.42}$$

A limit for the wick thickness should also be defined as

$$g_{12}(\mathbf{x}) = 2d(1 + \beta) - t_w \leq 0 \tag{6.43}$$

where $\beta = 0.2$ is the technology parameter [170]. This parameter ensures that very low values of wire diameter that violate technological feasibility for mesh type wicks [171] are not attained.

Finally, the following constraints are defined to ensure that heat pipe tube wall and end caps withstand the working fluid pressure.

$$g_{13}(\mathbf{x}) = 4\frac{\Delta P\left(d_o^2 + d_i^2\right)}{d_o^2 - d_i^2} - \sigma_y \leq 0 \tag{6.44}$$

$$g_{14}(\mathbf{x}) = 4\frac{\Delta P\left(d_o^3 + 2d_i^3\right)}{2\left(d_o^3 - d_i^3\right)} - \sigma_y \leq 0 \tag{6.45}$$

where σ_y is the yield strength of the tube material and $\Delta P = P_v - P_{amb}$. Note that $P_{amb} = 0$ for heat pipes used in space applications.

6.1.2 Results and discussion

The operating condition of the heat pipe can be parametrized by the heat transfer rate (Q) and the heat sink temperature (T_{si}). The heat pipe would often be expected to function in different operating conditions. For instance, in a satellite application the heat transfer rate would depend on the duty cycle and loading regime of the equipment being cooled. Likewise, the heat sink temperature would vary based on the position of the satellite and total heat generated by its components. Five different operating conditions summarized in Table 6.2 were chosen for the heat pipe design problem and all constraints explained in the previous section were imposed for every operating condition on designs obtained in the optimization. Therefore, the obtained designs are expected to function in a range of operating conditions.

Table 6.2: Selected operating conditions

#	T_{si} (°C)	Q (W)
1	0	25
2	0	100
3	15	62.5
4	30	25
5	30	100

FEPSO was employed for 50 separate design optimization runs to minimize the total mass of the heat pipe while satisfying all constraints (g_1 - g_{14}) for all selected operating conditions. Furthermore, 5 different population size (i.e. swarm size) and number of maximum iterations combinations were selected to repeat the runs. For a comparative analysis of the results the constrained evolutionary optimizer called NSGA2 [28] was also utilized to obtain results. A statistical summary of all results obtained by both algorithms can be seen in Table 6.3. Statistical data on number of *constraint-* and *objective function evaluations* (CFE and OFE respectively) performed when the best value of m_{tot} is found are also presented in this table. A selected result found by FEPSO is given in Table 6.4.

Results obtained by FEPSO are found to be better in both average and minimum when compared with NSGA2 results obtained with same population size and iteration number. The difference between solutions of the two algorithms were statistically significant ($p < 0.0001$ for each of the 5 combinations) according to nonparametric Wilcoxon method [164][3]. The difference between results found by the two algorithms for each case are illustrated in Figure 6.3.

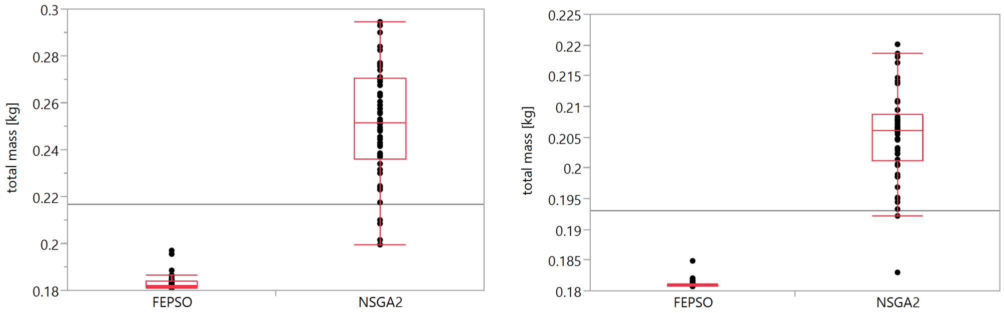
Note that FEPSO solutions required more CFEs but less OFEs when population size and termination criterion was kept same. On the other hand, selecting different termination criteria for the two algorithms resulted in similar number of CFEs. For instance, at $popSize = 150$, selecting $maxIter = 50$ for FEPSO and $maxIter = 150$ for NSGA2 resulted in similar amount of CFEs. Likewise, at $popSize = 250$,

---

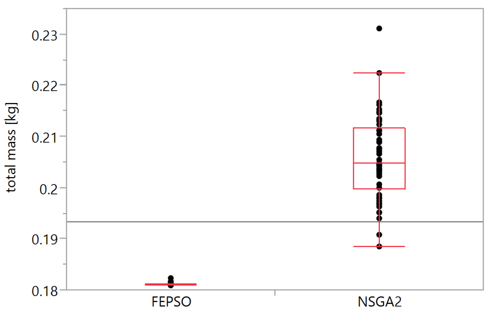
[3] Wilcoxon method was used due to non-normally distributed FEPSO results. Normality was tested with Shapiro-Wilk test [188].

Table 6.3: All results obtained in simulations for the heat pipe problem

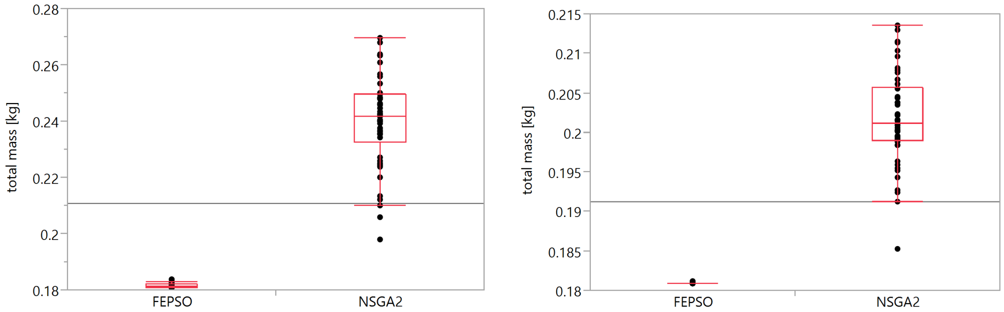
Population	Maximum		Total mass (kg)				CFE			OFE		
size	iterations	Algorithm	Best	Worst	μ	σ	Best	Worst	μ	Best	Worst	μ
150	50	NSGA2	0.1994	0.2947	0.2508	2.38×10^{-2}	6600	7500	7365	6600	7500	7365
		FEPSO	0.1809	0.1969	0.1831	3.14×10^{-3}	18 530	21 304	19 584	4160	5474	4813
150	150	NSGA2	0.1830	0.2201	0.2052	7.20×10^{-3}	18 300	22 500	21 684	18 300	22 500	21 684
		FEPSO	0.1808	0.1849	0.1811	6.24×10^{-4}	41 969	47 076	45 814	13 833	18 328	16 642
200	100	NSGA2	0.1884	0.2312	0.2059	8.26×10^{-3}	17 800	20 000	19 572	17 800	20 000	19 572
		FEPSO	0.1808	0.1822	0.1810	2.57×10^{-4}	42 043	45 854	44 202	12 323	16 461	14 401
250	50	NSGA2	0.1978	0.2696	0.2403	1.62×10^{-2}	10 750	12 500	12 200	10 750	12 500	12 200
		FEPSO	0.1808	0.1839	0.1816	7.54×10^{-4}	31 042	33 810	32 638	6819	8957	8001
250	150	NSGA2	0.1852	0.2135	0.2017	5.80×10^{-3}	33 500	37 500	36 805	33 500	37 500	36 805
		FEPSO	0.1808	0.1812	0.1809	7.47×10^{-5}	73 088	78 733	76 147	23 170	30 588	27 865



(a) $popSize = 150$, $maxIter = 50$ (b) $popSize = 150$, $maxIter = 150$



(c) $popSize = 200$, $maxIter = 100$



(d) $popSize = 250$, $maxIter = 50$ (e) $popSize = 250$, $maxIter = 150$

Figure 6.3: Graphical comparison of results obtained by NSGA2 and FEPSO for the heat pipe problem using different maximum iterations ($maxIter$) and population sizes ($popSize$)

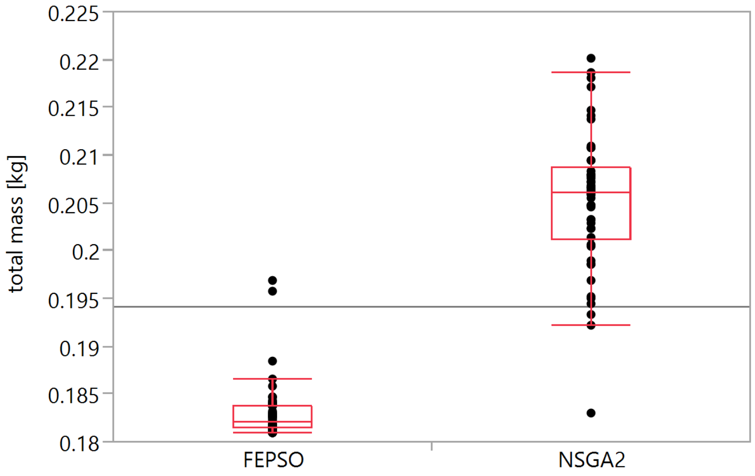
Table 6.4: A selected design obtained by FEPSO

Variable	Value
N	439
ε	0.9909
d_v	$36.69\,\mathrm{mm}$
t_w	$0.12\,\mathrm{mm}$
L_e	$50.00\,\mathrm{mm}$
L_c	$50.00\,\mathrm{mm}$
t_t	$0.30\,\mathrm{mm}$
M_f	ammonia
Tube/wick material	SAE 304 stainless steel
m_{tot}	$0.1808\,\mathrm{kg}$

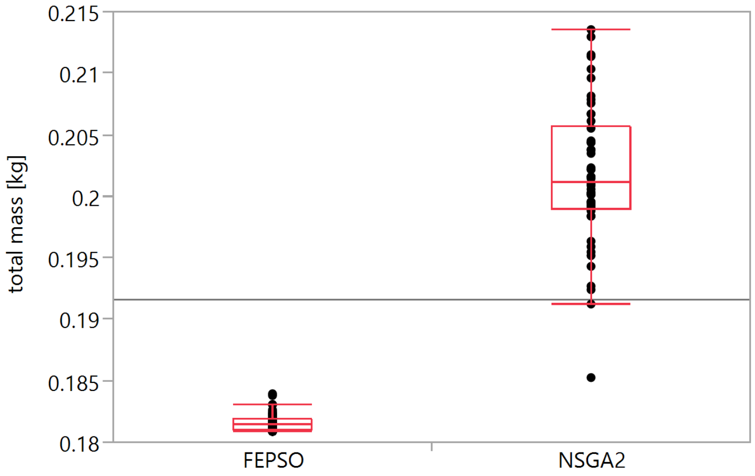
$maxIter = 50$ for FEPSO and $maxIter = 150$ for NSGA2 has the same effect. When results obtained for these cases were compared, FEPSO was found to have yielded in significantly better results (($p < 0.0001$ for both cases using Wilcoxon method). The difference between the results is illustrated in Figure 6.4. As a result, one can assume that FEPSO obtains better solutions when both algorithms are restricted to only make a certain number of CFEs.

6.2 Multi-objective heat pipe design problem

In the previous section the heat pipe design problem is handled as a single-objective problem with the objective being to minimize the weight. Design of the heat pipe can also be approached in a multi-objective manner. While minimizing the weight of the heat pipe, the amount of heat it can dissipate can also be maximized.



(a) $popSize = 150$; $maxIter = 50$ for FEPSO, $maxIter = 150$ for NSGA2



(b) $popSize = 250$; $maxIter = 50$ for FEPSO, $maxIter = 150$ for NSGA2

Figure 6.4: Graphical comparison of results obtained by NSGA2 and FEPSO which have similar number of CFEs

6.2.1 Problem definition

The multi-objective heat pipe design problem can formally be stated as to *minimize*

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} m_{tot} & -Q_{max} \end{bmatrix} \tag{6.46}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq M} \tag{6.47}$$

where $M = 28$ and Q_{max} represents the maximum amount of heat that the heat pipe can transfer. The decision vector of the problem is as shown in Eqn. 6.4. In the multi-objective context, the aim is to satisfy all constraints described by Eqns. 6.16 to 6.45 at two operating conditions that represent the minimum heat transfer rate and both ends of the heat sink temperature range. These two operating conditions can be denoted as (Q_{min}, T_{si}^{min}) and (Q_{min}, T_{si}^{max}). Where $Q_{min} = 25\,\mathrm{W}$ is the minimum heat transfer rate the heat pipe is expected to operate. Half of the constraints (g_1-g_{14}) are to ensure the heat pipe operates in one of these conditions while the rest (g_{14}-g_{28}) is for the other condition. The second objective ($y_2 = -Q_{max} = f_2(\mathbf{x})$) is the negative[4] of the maximum heat transfer rate the heat pipe can operate. In other words, Q_{max} is the maximum Q where all working conditions defined by Eqns. 6.16-6.45 are met. This requires the satisfaction of all conditions at both ends of the heat sink temperature range $[T_{si}^{min}, T_{si}^{max}]$. Maximization of the operating range of the heat pipe is shown in Figure 6.5.

The maximum allowable heat transfer rate (Q_{max}) is found numerically using the bisection method. The technique involves gradually increasing Q starting from Q_{min} to find a Q that violates working conditions. Then, bisection method is applied to find Q_{max} with a certain accuracy ($Q^{tol} = 0.1\,\mathrm{W}$). Note that this approach assumes working conditions are met at (Q_{min}, T_{si}^{min}) and (Q_{min}, T_{si}^{max}) since this objective is only relevant when the constraints of the problem are satisfied. In the meantime, the step size with which Q is increased has been taken as $\Delta Q = 10\,\mathrm{W}$. Likewise, the heat sink temperature range is defined with $T_{si}^{min} = 0\,^{\circ}\mathrm{C}$ and $T_{si}^{max} = 30\,^{\circ}\mathrm{C}$. A flowchart summarizing the procedure is given in Figure 6.6

---

[4] The maximum heat transfer rate is negated because it is to be maximized.

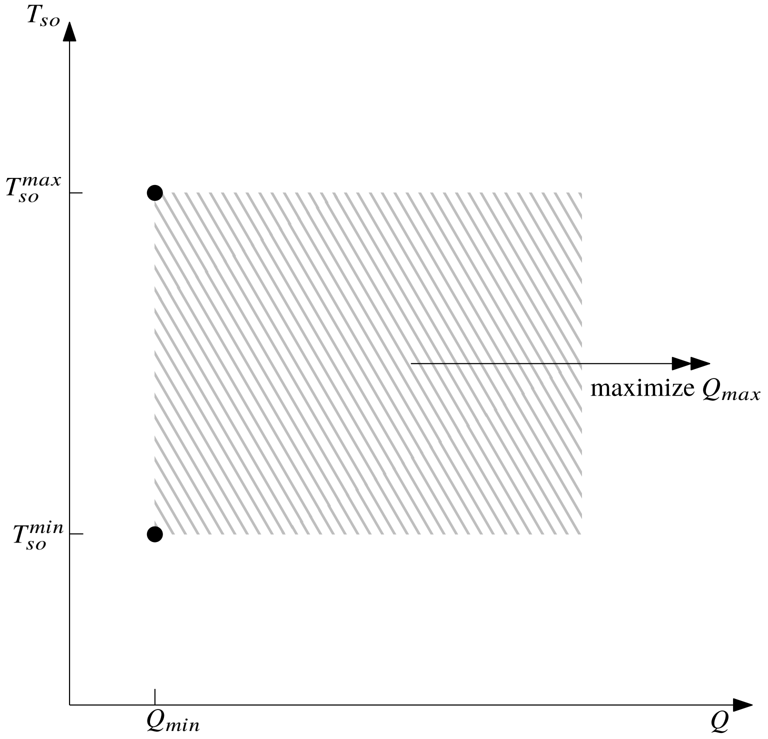


Figure 6.5: Maximization of the heat pipe operating condition range

6.2.2 Results and discussion

The multi-objective heat pipe design problem was solved by MOFEPSO, NSGA2, and MOGA2 with different combinations of population size and number of generations. Statistical summary of the results obtained during these runs are given in Table 6.5. However, since the problem is multi-objective, separate statistical properties evaluated for each objective are not completely meaningful. Graphical comparison of Pareto fronts is an alternative approach for examining results obtained in multi-objective problems. This technique is efficient when there are only two objectives. When there are more than three objectives, graphical comparison of Pareto fronts becomes practically impossible due to the fact that Pareto fronts being hyper-surfaces of a degree higher than three which are very complicated to visualize. This is one of the reasons why multi-objective techniques are not recommended for problems with more than three objectives. This type of problems are generally categorized as many-objective problems and require some handling techniques. Since the multi-objective heat pipe problem has only two objectives, it makes sense to make an evaluation based on

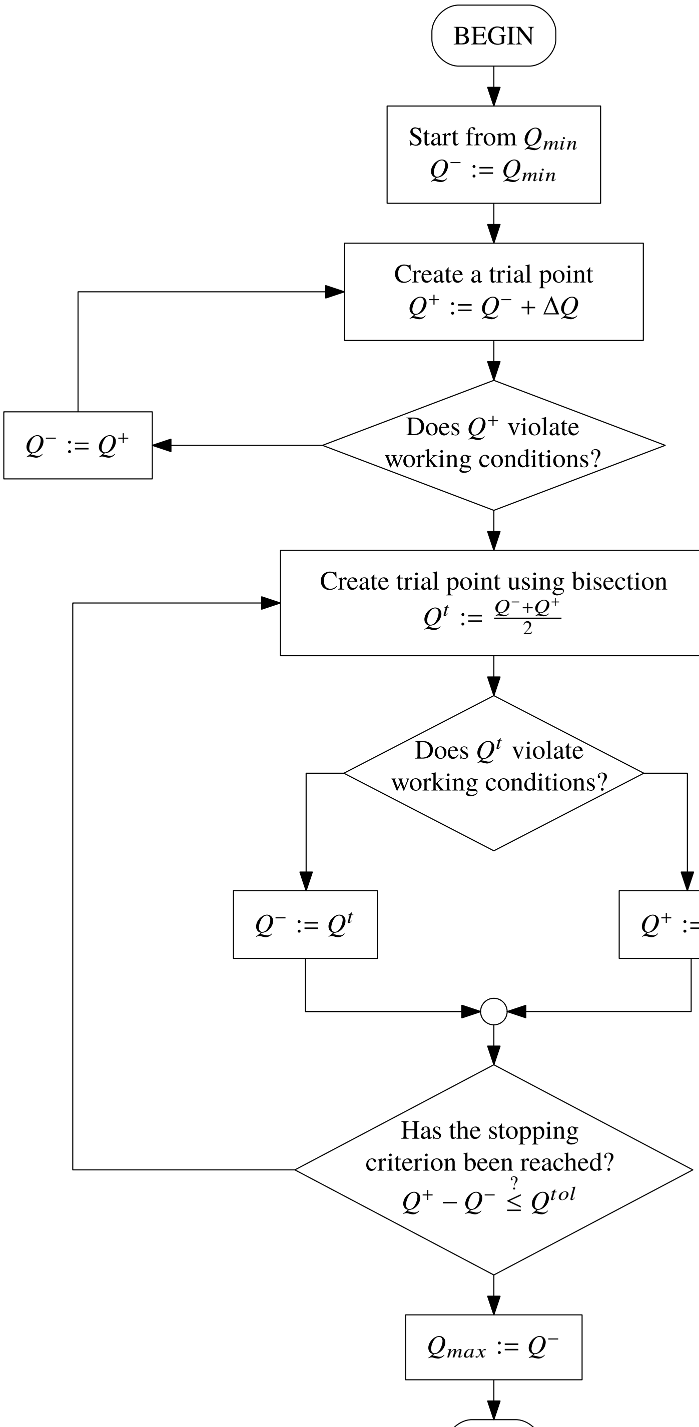

BEGIN
Start from Q_{min}
$Q^- := Q_{min}$
Create a trial point
$Q^+ := Q^- + \Delta Q$
Does Q^+ violate working conditions?
$Q^- := Q^+$
Create trial point using bisection
$Q^t := \frac{Q^- + Q^+}{2}$
Does Q^t violate working conditions?
$Q^- := Q^t$
$Q^+ := Q^t$
Has the stopping criterion been reached?
$Q^+ - Q^- \overset{?}{\leq} Q^{tol}$
$Q_{max} := Q^-$
END

Figure 6.6: Flowchart of the numerical approach to evaluate Q_{max} of a heat pipe design

images of the Pareto fronts obtained by different algorithms. Nevertheless, the solution procedure was repeated several times for each algorithm (10 times for MOFEPSO and NSGA2, while 5 times for MOGA2 at each population size and maximum iterations combination) and it is very difficult to compare every solution combination. Therefore, four randomly selected solution combinations for each population size and maximum iterations configuration containing a solution from each algorithm was plotted in Figures 6.7 thru 6.10.

MOFEPSO is found to produce the best Pareto front among the algorithms especially at lower numbers of maximum iterations. When the algorithms are allowed to produce more generations, the resulting Pareto fronts become similar. However, MOFEPSO tends to find better results at the upper left corner of the Pareto front even at higher number of generations. This advantage is not observed at the lower right corner of the Pareto front and there are some cases where MOGA2 or NSGA2 perform better than MOFEPSO in this region. When compared with MOGA2 and NSGA2, MOFEPSO seems to produce less number of solutions in the Pareto front for the multi-objective heat pipe design problem. Another disadvantage of MOFEPSO seems to be the higher number of constraint function evaluations it requires. Although, MOFEPSO makes less number of objective function evaluations, the difference is not significant. MOFEPSO is known to perform with significantly low number of objective function evaluations in highly-constrained problems (see Chapter 5). However, when the problem has relatively easier constraints as in the heat pipe problem, the difference in the number of objective function evaluations is not as dramatic. Broadly speaking, MOFEPSO was found to be able to perform satisfactorily in a multi-objective problem that is somewhat less constrained than problems that it is designed to deal with. In fact, MOFEPSO was better in some aspects from widely accepted algorithms.

Solutions for the multi-objective heat pipe problem present a large number of alternatives to the designer. These not only include solutions that can substitute the solutions found for the single-objective version but also provide valuable insight about available design options and the aspects of performance that these designs achieve in the objective space. For instance from this data, the maximum rate of heat transfer that can be achieved for a given heat pipe mass can be extracted. Furthermore, the designer can judge how much gain they would have in terms of maximum heat transfer rate

Table 6.5: Statistical summary of results obtained in simulations for the multi-objective heat pipe problem

Population	Maximum					y_1				y_2			
size	iterations	Algorithm	N	CFE	OFE	Best	Worst	μ	σ	Best	Worst	μ	σ
100	50	MOGA2	1663	5000	5000	0.1445	1.5677	0.4421	0.2677	−382.81	−168.98	−25.59	85.68
		NSGA2	3144	5000	5000	0.1197	1.7019	0.3994	0.2664	−594.22	−167.57	−26.03	103.16
		MOFEPSO	509	14 859	4866	0.1059	1.5562	0.3551	0.2660	−571.41	−165.45	−25.00	114.35
	100	MOGA2	7396	10 000	10 000	0.1057	1.7279	0.4126	0.2689	−619.30	−217.06	−25.07	140.72
		NSGA2	12 727	10 000	10 000	0.1045	1.4492	0.3920	0.2722	−610.70	−205.24	−25.73	149.05
		MOFEPSO	729	26 823	9887	0.1001	1.7031	0.3204	0.2466	−568.13	−167.41	−25.00	128.85
200	50	MOGA2	1727	10 000	10 000	0.1106	1.4564	0.4403	0.2935	−518.91	−207.33	−26.10	133.50
		NSGA2	4173	10 000	10 000	0.1167	1.7225	0.4346	0.2724	−580.31	−208.44	−25.59	136.83
		MOFEPSO	733	29 473	9721	0.1024	1.4116	0.3242	0.2360	−559.69	−172.08	−25.07	127.76
	100	MOGA2	8235	20 000	20 000	0.0996	1.4542	0.4057	0.3154	−624.06	−215.62	−25.15	174.53
		NSGA2	19 478	20 000	20 000	0.1009	1.3568	0.3839	0.2666	−626.25	−214.76	−25.66	160.03
		MOFEPSO	974	50 845	19 737	0.1007	1.5848	0.3225	0.2491	−595.08	−180.72	−25.07	139.64

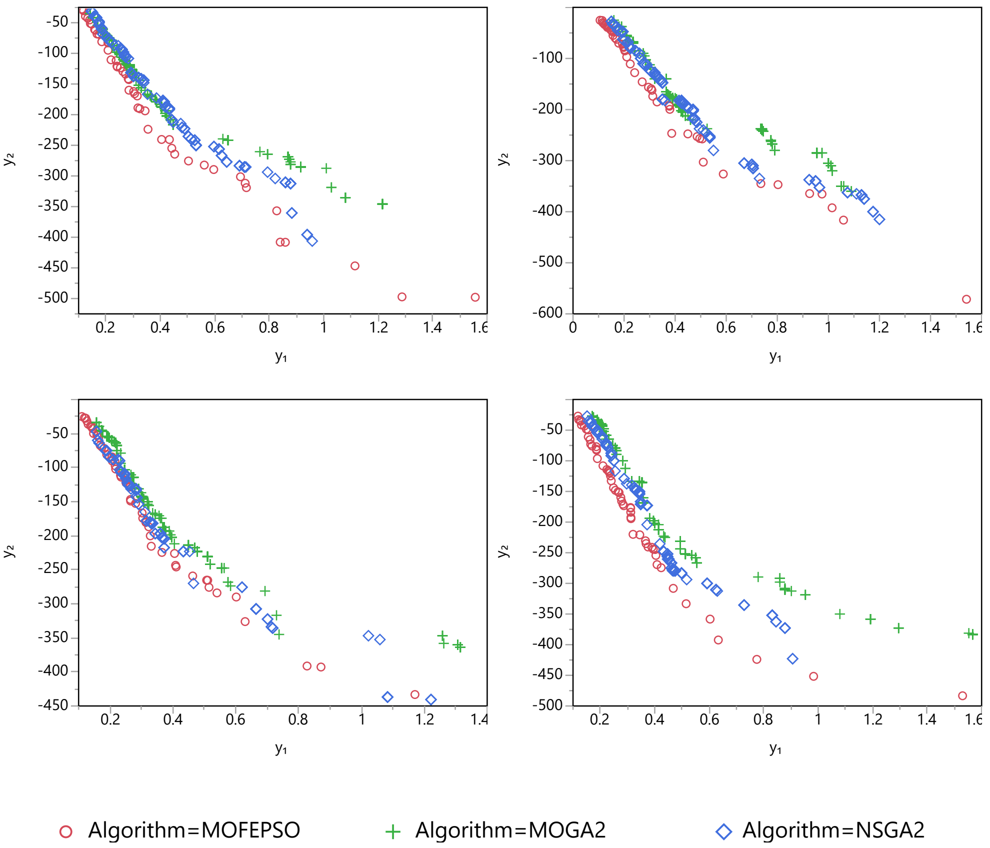


Figure 6.7: Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 100, number of iterations: 50)

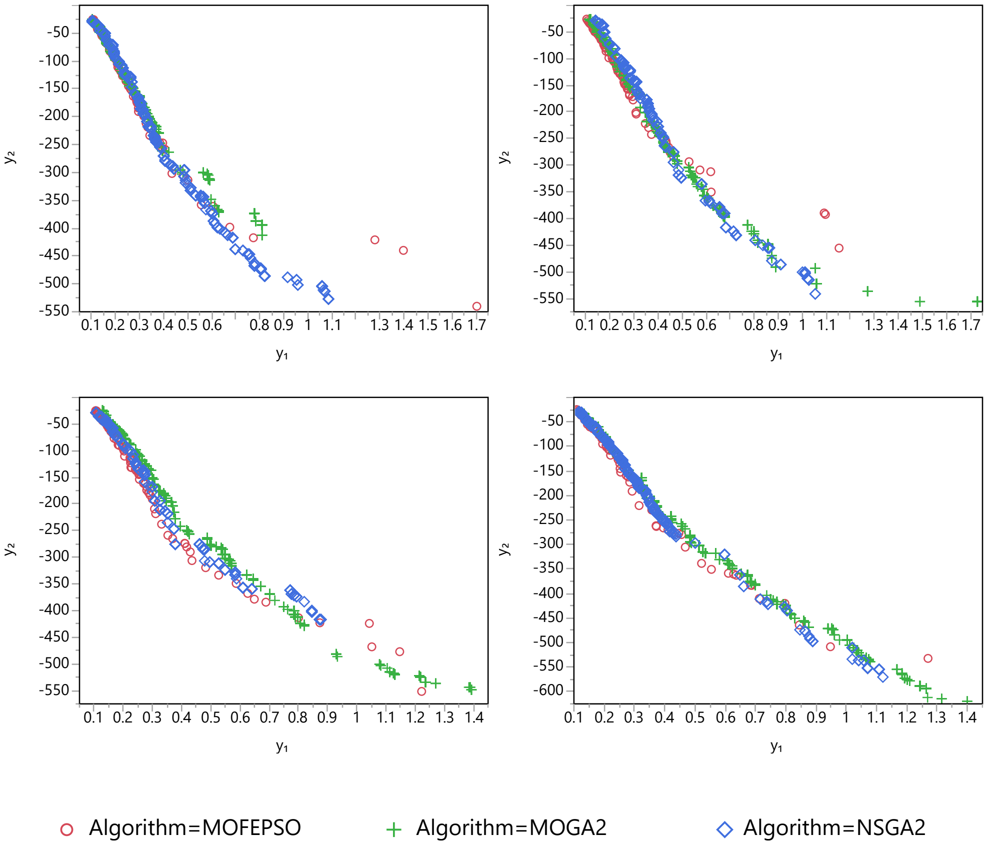


Figure 6.8: Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 100, number of iterations: 100)

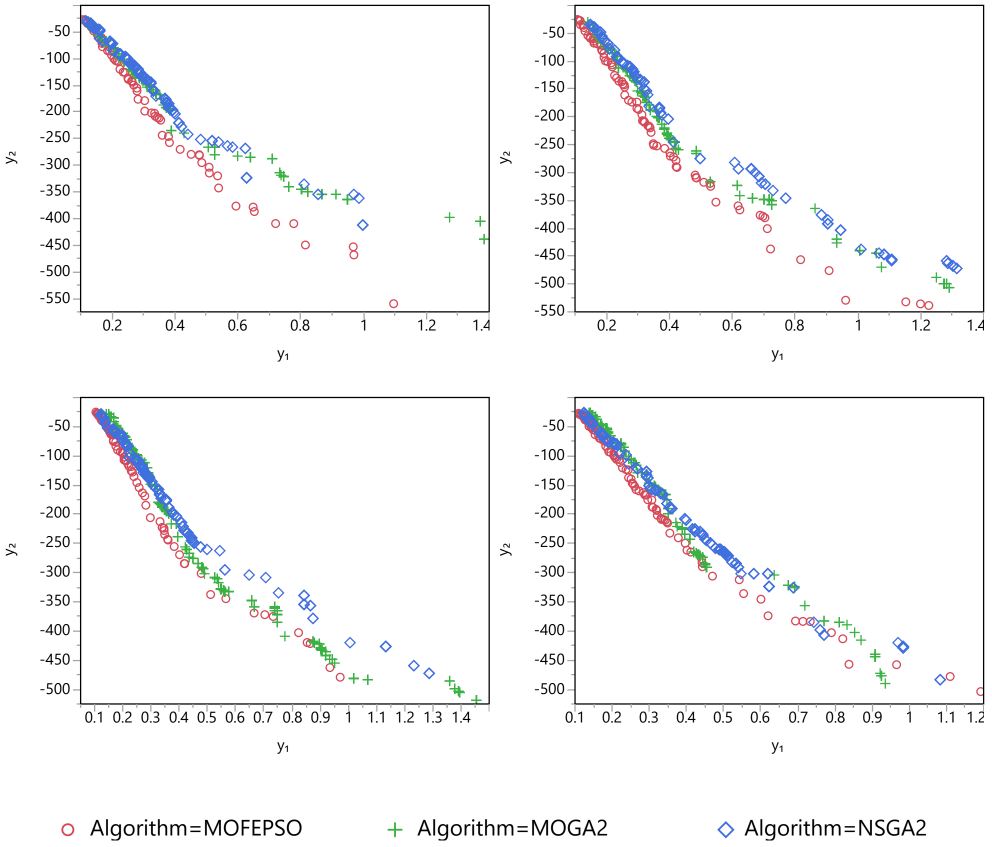


Figure 6.9: Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 200, number of iterations: 50)

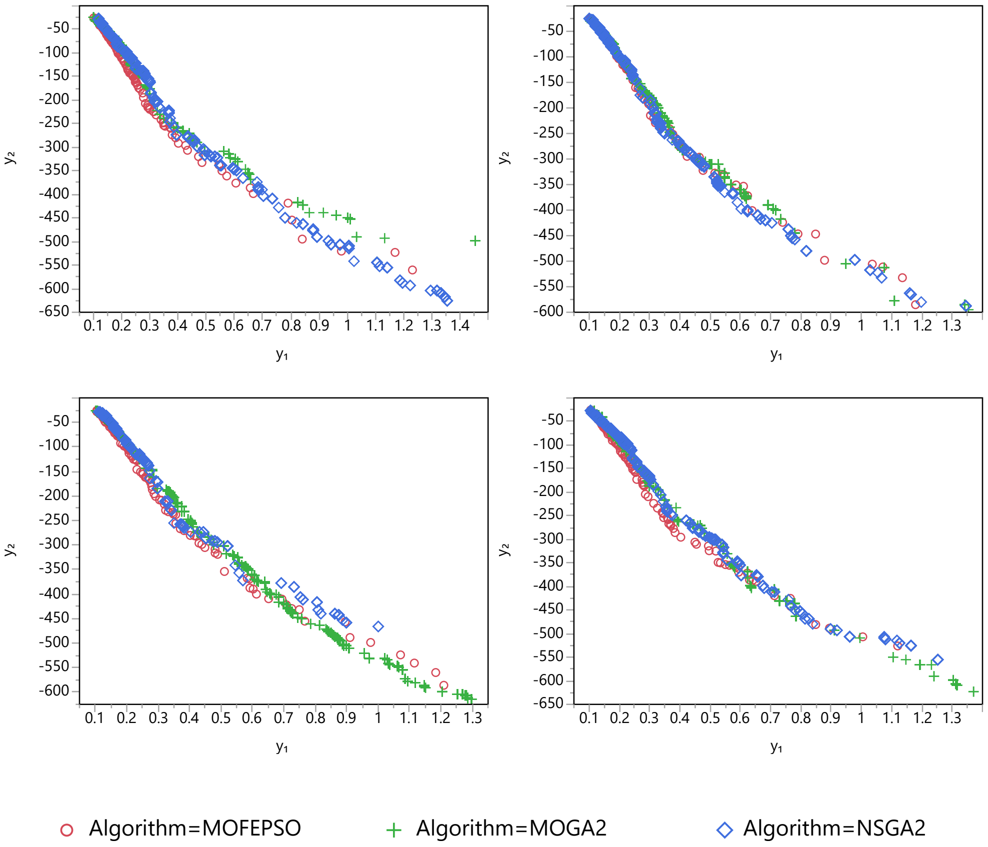


Figure 6.10: Selected Pareto fronts for the multi-objective heat pipe problem obtained by MOFEPSO, MOGA2, and NSGA2 (population size: 200, number of iterations: 100)

in the case of allowing a higher heat pipe mass. Therefore, a design can be chosen by making a trade-off between these two objectives and their relation given by the solutions obtained through the multi-objective optimization [189]. A third objective, such as the cost, may provide additional leverage.

6.3 Closure

In the first part of this chapter, the single-objective heat pipe design problem and the solution procedure was described. Multiple operating conditions were taken into consideration. FEPSO and NSGA2 were used to minimize the heat pipe mass while satisfying all constraints for all operating conditions. Feasible solutions were found by both evolutionary algorithms. However, results obtained by FEPSO were significantly better in both of the following cases:

- When population size and maximum number of iterations are same for both algorithms
- When CFEs and OFEs are limited to the same number for both algorithms

In the second part, the multi-objective heat pipe design problem was introduced. A new objective (Q_{max}) was defined to maximize the operating range of the heat pipe. problem was solved by MOFEPSO, NSGA2, and MOGA2 with different combinations of population size and number of generations. MOFEPSO was found to produce the best Pareto front among the algorithms especially at lower numbers of maximum iterations. When the algorithms are allowed to produce more generations, the resulting Pareto fronts become similar. Results presented a plethora of solutions to chose from through a trade-off between the two objectives and their relation given by the solutions obtained through the multi-objective optimization.

The results presented in this chapter show that FEPSO and MOFEPSO not only perform well in highly constrained problems that they are designed for but also have promising effectiveness in relatively less constrained problems.

CHAPTER 7

ALGORITHM CHARACTERISTICS

7.1 Qualitative and empirical analysis of complexity

Time complexity is a type of computational complexity that characterizes the amount of time the algorithm needs to run. Therefore, time complexity is an important parameter in the selection and assessment of an algorithm.

Time complexity of MOFEPSO has been investigated as well[1]. Since MOFEPSO is an enhanced version of the original PSO, the time complexity of the algorithm has a close resemblance to that of the basic PSO algorithm. However, since the original PSO algorithm is not multi-objective and cannot handle constraints, MOFEPSO is compared to a generic multi-objective PSO constituting a constraint handling mechanism that is compatible with Deb's approach [58].

Initialization steps and computations performed in the main loop are taken into consideration in the complexity analysis presented in Table 7.1. Operations (such as empty array initializations or simple operations performed on variables/arrays) which are computationally insignificant if compared to the others, are disregarded.

At the initialization phase, the only operation which introduces significant computational burden to MOFEPSO is the constraint sensitivity calculation. This step requires each particle be temporarily moved along each fundamental direction to detect constraint sensitivities. Naturally, the procedure involves CFEs. That is, additional computational cost is directly related to the computational complexity of constraint

---

[1] In the complexity analysis MOFEPSO was chosen over FEPSO due to the fact that MOFEPSO is more complex. Additional complexity is a consequence of mechanisms implemented in MOFEPSO for handling multiple objectives.

Table 7.1: Complexity comparison of MOFEPSO with generic multi-objective PSO with a constraint handling mechanism

	MOFEPSO	Generic multi-objective PSO
Initialization	Random postion and velocity initialization	Random postion and velocity initialization
	Evaluation of constraint functions	Evaluation of constraint functions
	Initialization of PBEST and GBEST(1)	Initialization of PBEST and GBEST(1)
	Constraint sensitivities calculation(2)	-
Every iteration for each particle	Activated constraint selection(3)	-
	Leader (guide) selection	Leader (guide) selection
	Calculate velocity and candidate position	Calculate velocity and new position
	Virtual Boundary Search(4,5)	-
	Update particle position	Update particle position
	Update constraint priorities(3)	-
	Calculate objective function(6)	Calculate objective function(6)
	Update PBEST and GBEST	Update PBEST and GBEST

(1)Includes calculation of objective functions for feasible points

(2)Involves $I \times N$ temporary particle movements and constraint function evaluations

(3)Only for infeasible particles

(4)Only for virtually infeasible particles

(5)Involves consecutive constraint function evaluations until virtual boundary is found

(6) Only for feasible particles and particles which became feasible within the current iteration

functions. For complex constraint functions, a simpler approach could be facilitated like OCCURM (i.e. occurrence matrix) procedure of Khorshid and Seireg [96]. Note that if constraint sensitivities are known beforehand, this initialization step could be avoided altogether.

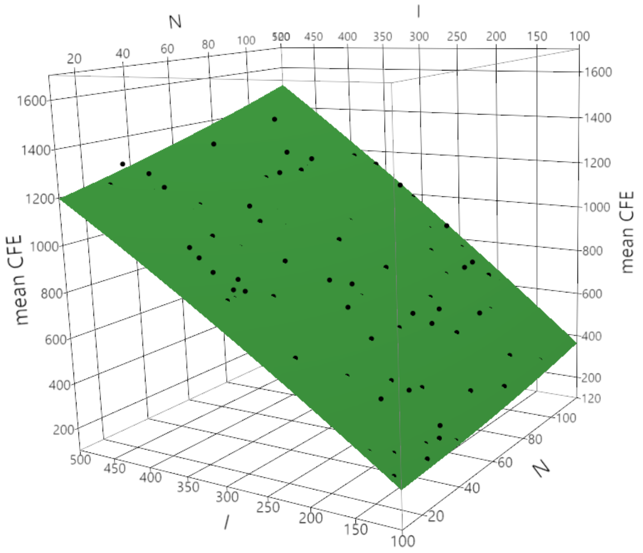
Complex operations are more critical within the main function of the algorithm. Additional computational load in the main loop of MOFEPSO is attributed to the constraint selection-, VBS, and constraint priority updating procedures (see Table 7.1). Activated constraint selection and constraint priority selection are performed on infeasible particles only. These procedures involve the update of several elements in initialized arrays. Therefore, the corresponding computational complexity may be neglected for all practical purposes. However, VBS requires consecutive CFEs until it finds a legitimate virtual boundary. As expected, this is the major source of computational load especially when the constraint functions are costly. For instance, it has been observed in this study that VBS for the MOG[2] problem requires, on average, 2 extra constraint evaluations (within the range of 1 to 27). Consequently, the time complexity of MOFEPSO may be assessed by the number of CFEs performed since most operations involve constraints.

After this qualitative analysis, a numerical evaluation of the time complexity associated with the proposed method is in order. The benchmark problem titled C14 of Mallipeddi and Suganthan [190] is utilized for this purpose. Figure 7.1 illustrates average number of CFE values at sampled data points and the surface fitted on the data. Swarm size and number of decision variables were changed using uniform random sampling with a sample size of 100. Consistent with the qualitative analysis above, the total computation time at each run is found to be correlated with number of CFEs ($p < 0.0001$). Therefore, the average number of CFEs per iteration can be used as a measure for time complexity.

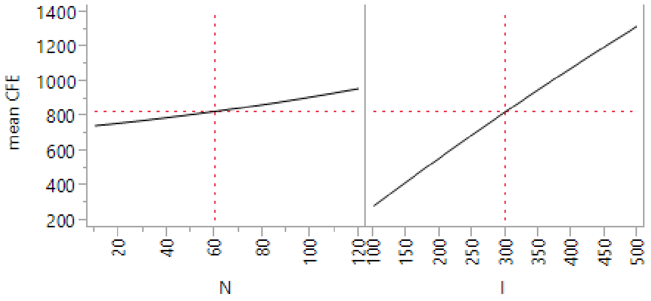
As seen in Figure 7.1, a high order increase in time complexity with neither of the parameters (N: number of decision variables and I: swarm size) have been observed. Therefore, increase in time complexity of MOFEPSO with respect to these parameters can be said to be benign.

---

[2] The MOG problem is described in Section 5.2.



(a) Response surface



(b) Factor effects

Figure 7.1: Mean number of CFEs versus number of decision variables (N) and swarm size (I)

7.2 Comparison of multi-objective and single-objective approaches

MOFEPSO is a Pareto based multi-objective optimization approach while FEPSO is a single-objective technique. Single-objective algorithms can also be used in multi-objective problems via special methods. Most commonly used method is utilization of fixed weight linear aggregation functions to convert the problem into a single-objective one. In this chapter, several multi-objective problems are solved by both MOFEPSO and FEPSO (with different aggregation functions) to compare these approaches.

When multi-objective problems are converted into single-objective problems via aggregation, objective functions need to be prioritized- and weighted before the optimization algorithm is employed. This carries the risk of introducing bias, especially when insufficient information related with the problem exists. On the contrary, in multi-objective approaches all objectives can be optimized simultaneously. However, in this case unless all objective functions have their optimums at the exact same point, a single optimum can not exist. Hence, the scalar concept of optimality in single-objective problems doesn't apply to multi-objective context. Instead, multi-objective optimization results in a set of Pareto optimal[3] solutions from which a solution can be selected by the designer through a trade-off of objectives.

There are some other studies comparing Pareto based approaches with others. Zitzler and Thiele [191] compare Pareto based NSGA [26] with a fixed weight aggregation approach and show superiority of NSGA in several problems. Coello [192] states that the major shortcoming of single-objective methods using fixed weight aggregation functions is that the weights are determined before sufficient information is available for the problem and the solutions become dependent to these predetermined weights. Meanwhile, Ishibuchi, Nojima, and Doi [193] discuss methods for comparing multi-objective and single-objective optimization algorithms and report some results of computational experiments where better results are obtained with multi-objective approaches.

---

[3] Please see Section 2.1.1 for a more extensive review of Pareto optimality and Pareto sets.

7.2.1 Comparison method

SRN [28], RC-MOG, and MV-MOG[4] problems were used for comparing single- and multi-objective approaches. SRN can be defined as to *minimize*

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \equiv \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) \end{bmatrix}^T \tag{7.1}$$

subject to

$$g_m(\mathbf{x}) \leq 0, \quad m \in \mathbb{N}_{>0}^{\leq 2} \tag{7.2}$$

where

$$f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \tag{7.3}$$

$$f_2(\mathbf{x}) = 9x_1 - (x_2 - 1)^2 \tag{7.4}$$

$$g_1(\mathbf{x}) = x_1^2 + x_2^2 - 225 \tag{7.5}$$

$$g_2(\mathbf{x}) = x_1 - 2x_2 + 10 \tag{7.6}$$

Figure 7.2 illustrate constraint- and objective functions of the SRN problem.

MOFEPSO was run 50 times for each of the problems to obtain Pareto solution sets. Additionally, following aggregation functions were utilized to convert all three problems into separate single-objective problems ($3 \times 3 = 9$ problems in total):

$$y^1 = 0.5y_1 + 0.5y_2 \tag{7.7}$$

$$y^2 = 0.6y_1 + 0.4y_2 \tag{7.8}$$

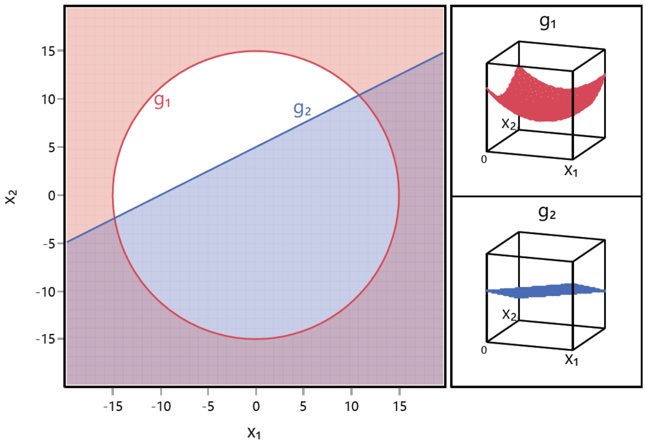
$$y^3 = 0.7y_1 + 0.3y_2 \tag{7.9}$$

Therefore, single objective approach (i.e. FEPSO) has been applied 50 times for each aggregation function of each problem ($3 \times 3 \times 50$).

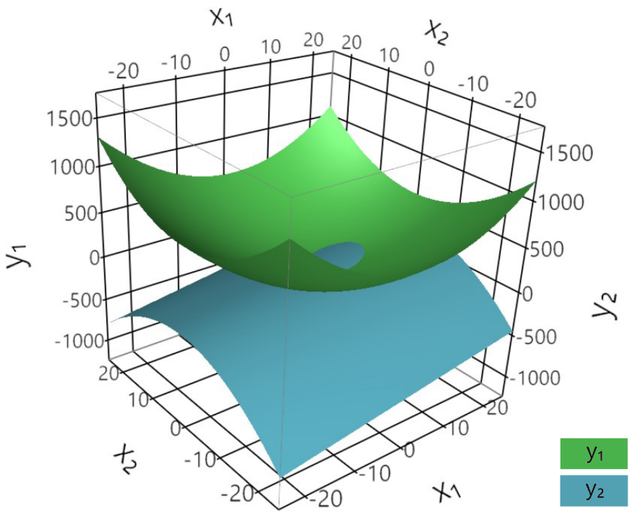
Note that while single-objective runs yield a single best solution, multi-objective runs produce a set of non-dominated solutions. Multi-objective approaches don't require predetermined objective weights and the trade-off between objectives can be performed after the optimization. To be able to compare resulting multi-objective Pareto solution sets with single objective best solutions, each of the three aggregation functions (Eqns. 7.7, 7.8, and 7.9) were evaluated for every element of the Pareto

---

[4] See Section 5.2 for detailed description of RC-MOG and MV-MOG problems.



(a) Constraint functions



(b) Objective functions

Figure 7.2: Constraint- and objective functions of the SRN problem

set. When a multi-objective result set is compared with a single-objective result, The solution that has the best corresponding aggregation function (i.e. the function used in single-objective run) value among the Pareto set is used. Furthermore, instead of directly comparing objective values, values of aggregation functions are compared. This approach ensures an objective comparison.

7.2.2 Results and discussion

Algorithm parameters given in Table 5.6 were used in FEPSO and MOFEPSO runs. Likewise, both population size and maximum number of generations were chosen as 100 for SRN.

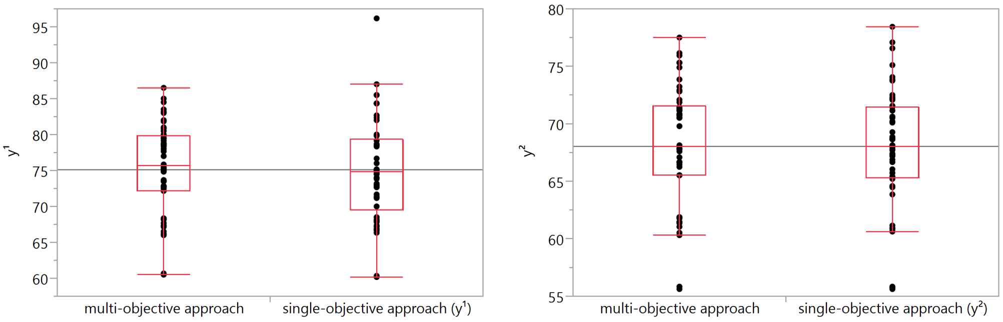
When RC-MOG and MV-MOG results are examined with respect to single-objective criteria y^1 and y^2, they were found to be normally distributed whereas y^3 criterion did not fit to normal distribution[5]. Accordingly, analysis involving y^1 and y^2 criteria were performed via Student's t-test while Wilcoxon signed rank test [164] was used for analysis related with y^3. On the other hand, none of the results obtained by SRN were distributed normally. Hence, Wilcoxon signed rank test was used in all SRN comparisons.

Upon comparison of results obtained for RC-MOG, neither the multi-objective approach nor the single-objective approach was found to be better than the other (Figure 7.3). On the other hand, results of multi-objective approach for MV-MOG were significantly better than the single-objective results in terms of y^2 and y^3 criteria ($p = 0.0434$ and $p = 0.0196$ respectively). Graphical comparisons of MV-MOG results with respect to each of the three criteria are given in Figure 7.4. Likewise, SRN results obtained with MOFEPSO were found to be at least as good as results obtained by FEPSO runs performed separately for each of the criteria (y^1, y^2, and y^3).

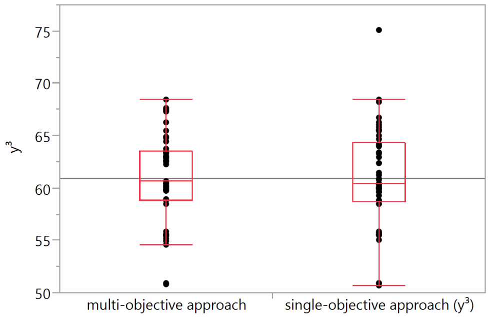
In the light of these facts, it is safe to say MOFEPSO, with no preset weights on the objectives, have yielded results that are at least as good as the results obtained with separate FEPSO runs for each of the three criteria. Hence, a single MOFEPSO run is worth three FEPSO runs for the problems considered. Like other Pareto based

---

[5] Shapiro-Wilk test [188] was used to test for normality.



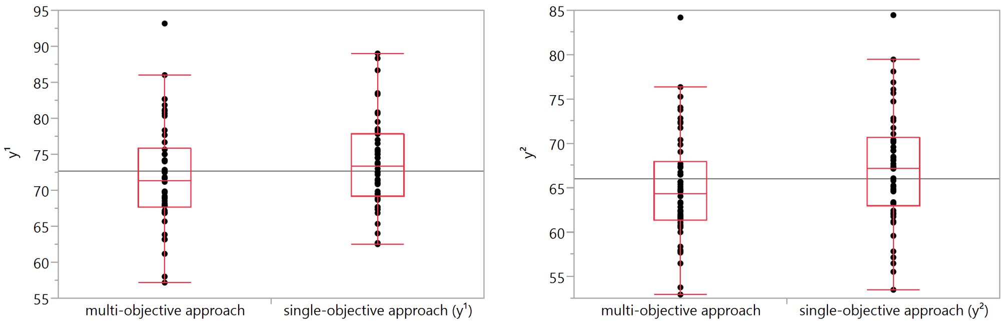
(a) Criterion y^1 (b) Criterion y^2



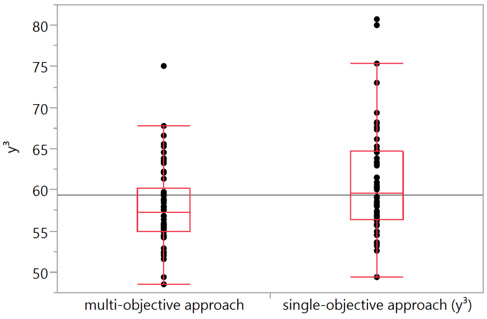
(c) Criterion y^3

Figure 7.3: Comparison of multi-objective results obtained for RC-MOG with results obtained by single-objective approaches using y^1, y^2, and y^3 criteria

.



(a) Criterion y^1 (b) Criterion y^2



(c) Criterion y^3

Figure 7.4: Comparison of multi-objective results obtained for MV-MOG with results obtained by single-objective approaches using y^1, y^2, and y^3 criteria

.

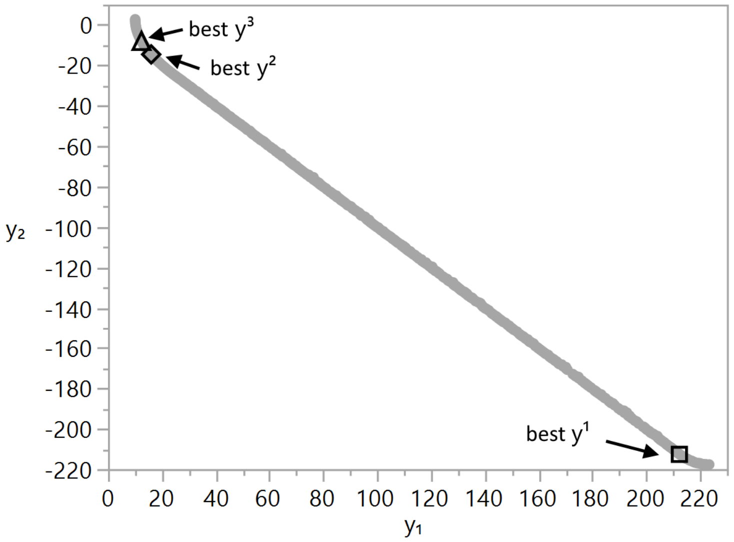


Figure 7.5: All Pareto solutions obtained for the SRN problem in a single selected multi-objective optimization run (Pareto front - 1879 points) and solutions that are the best (have the lowest value) in terms of y^1, y^2, and y^3 criteria

algorithms, MOFEPSO produces a Pareto set at the en of an optimization run. Designer can select a solution considering all solutions available in the set. With this approach, weights can be assigned to objectives after the optimization run and these weights can be change without requiring a rerun of the algorithm. For instance Figure 7.5 illustrates all non-dominated solutions obtained in a multi-objective MOFEPSO run and solutions among these that are determined to have the best (lowest) values of y^1, y^2, and y^3 criteria. As seen here, each solution in the Pareto set obtained by a single multi-objective optimization run actually corresponds to a single-objective run performed for a fixed weight aggregation function. Therefore, although multi-objective algorithms contain steps that introduce additional complexity such as special selection techniques, due to the fact that a single multi-objective optimization run yields solutions that would otherwise require multiple runs of single-objective algorithms, they provide a substantial advantage in terms of computational complexity. The size of the Pareto set obtained by a multi-objective optimization run largely depends on the problem. In Figure 7.6, number of solutions in the Pareto set is plotted against number of iterations and population size for the SRN problem.

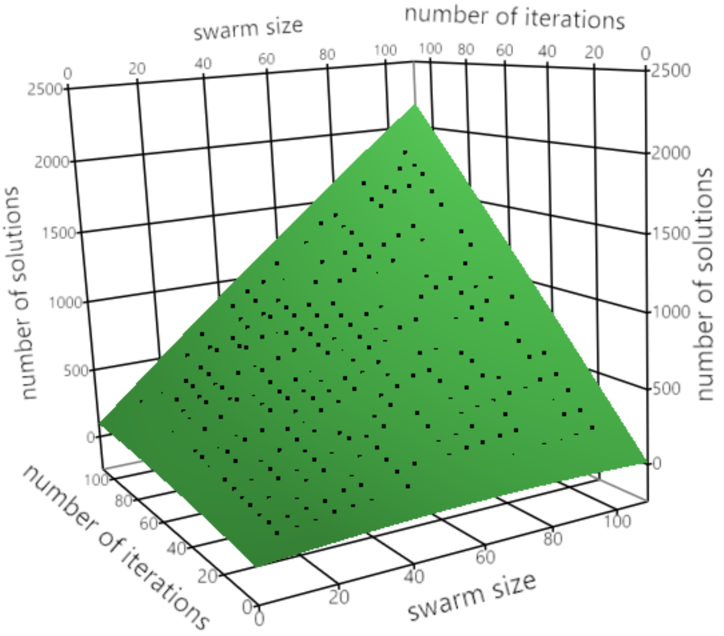


Figure 7.6: Number of solutions obtained in the Pareto set with varying number of iterations- and swarm size

7.3 Particle flight characteristics

Most unique aspect of the FEPSO particle movement is the Virtual Boundary Search. It entails limiting the particle movement to an implied boundary. If this boundary is passed, the particle moves into a less feasible position (i.e. some of its previously satisfied constraints become violated). FEPSO prohibits such a position change to allow progressive improvement of constraint satisfaction. Therefore, particles can only move towards a region where all of their already satisfied constraints remain intact and favorably some others become satisfied too. For feasible particles (i.e. particles that already satisfy all constraints), this implies that returning to an infeasible region is not possible.

Another distinct feature of FEPSO is that infeasible particles only move along decision space dimensions that a selected constraint (i.e. activated constraint) is sensitive to. This ensures a fast convergence to regions where high priority constraints become feasible while maintaining the diversity of the swarm.

Some aspects of these unique properties can be visualized and examined with an artificial scalable optimization problem. Suppose that a small feasible region in the shape of a hypercube exists in the center of the N-dimensional decision space. This can be achieved through a couple of linear constraints for every decision space dimension. Since each of these constraints only depend on a single decision vector, infeasible FEPSO particles shall only move along a single dimension. For the sake of convenience, optimum is also assumed to be at the center. The problem can be defined as to *minimize*

$$f(\mathbf{x}) = \sum_{n=1}^{N} |x_n| \tag{7.10}$$

subject to

$$g_n(\mathbf{x}) = -\delta - x_n \leq 0 \tag{7.11}$$

$$g_{2n}(\mathbf{x}) = x_n - \delta \leq 0 \tag{7.12}$$

where $\delta = 0.1$ and $M = 2N$. In two dimensions the constraint functions and the feasible region can be visualized as shown in Figure 7.7. Note that the problem is scalable in terms of number of decision variables. Definition of the problem is valid for any number of variables.

Initially, the problem was employed in two dimensions. FEPSO was used with a swarm size of five and particle movements in the first several iterations were examined. Traces of particle movements in the first six iterations are plotted in Figure 7.8. As seen in this figure, the particles are initialized at infeasible locations. They quickly move towards feasible regions. This particular example is designed for ease of visualization and particles move along a single dimension due to the fact that each constraint depends only on a single decision variable. Notice that the particles stick to constraint boundaries if any of their already satisfied constraints are to become violated. Another point worth mentioning is the importance of activating different constraints at each iteration. Overemphasizing highest priority constraints might cause a more stagnant swarm since many particles quickly move towards and stick to boundaries related with those constraints. It can be observed in Figure 7.8 that within the first six iterations, all five particles converge to the feasible region.

Further particle movements within the feasible region are also plotted in Figures 7.9 and 7.10 for iterations 2 through 13. Once the particles move into the feasible region

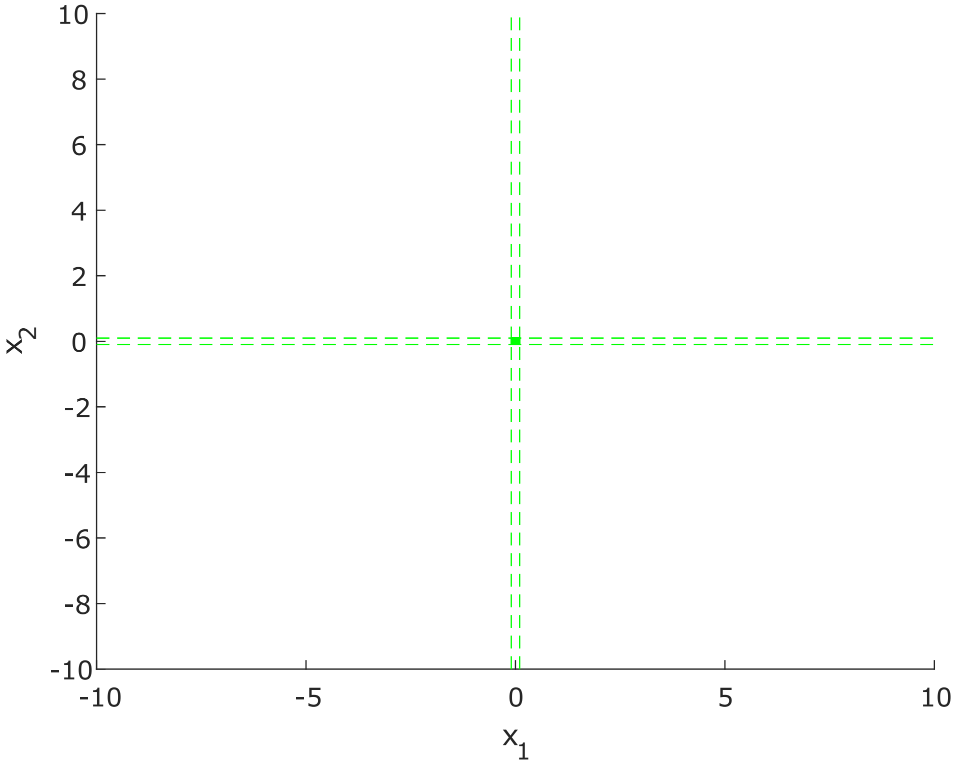
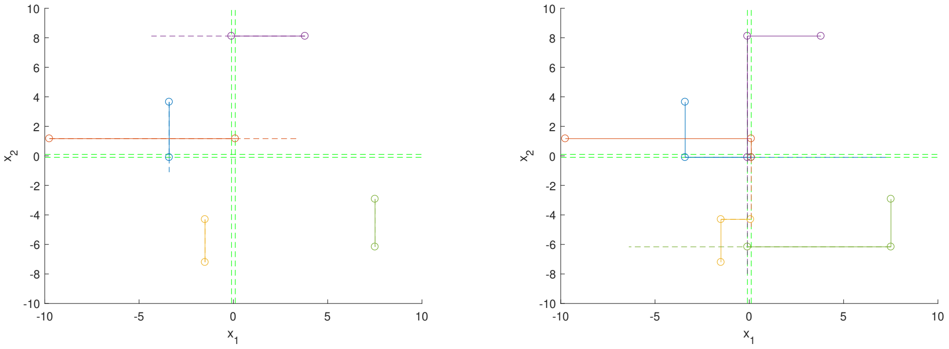


Figure 7.7: Constraint functions and the feasible regions of the hypercube problem in two dimensions

they converge to the center. Note that the particles are never allowed to leave this region and they bounce from the constraint boundaries.

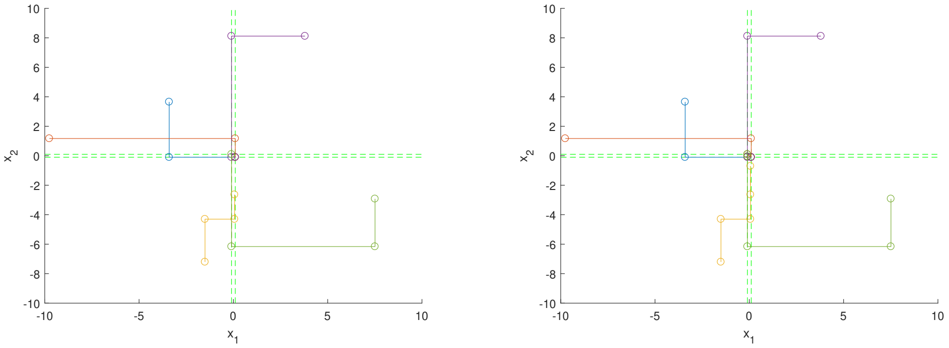
The three dimensional version of the problem was also examined. Once again, FEPSO was employed with only five particles for ease of visualization. As seen in Figure 7.11 all particles enter the feasible region within 6 iterations. Similar to the two dimensional case, once in the feasible region, particles begin moving towards the center bouncing off from the boundary walls. Figures 7.12 and 7.13 depict the behavior in the feasible region for the first several iterations after particles begin entering the feasible region. Note that, although the constraint boundaries are not explicitly shown in these figures, axis limits of the plots constitute the boundaries of the feasible region.

As discussed in the previous chapters, FEPSO uses the information produced by the constraint functions to search for the virtual boundary. Therefore, constraint functions providing a clear measure of how much they are violated and a margin of satisfaction



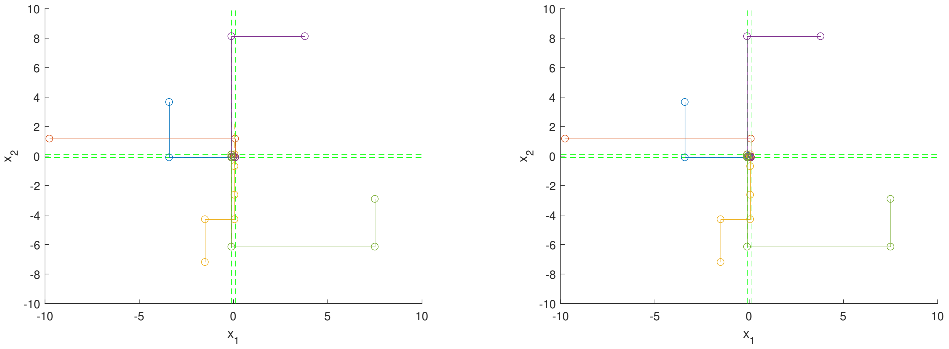
(a) Iteration 1

(b) Iteration 2



(c) Iteration 3

(d) Iteration 4

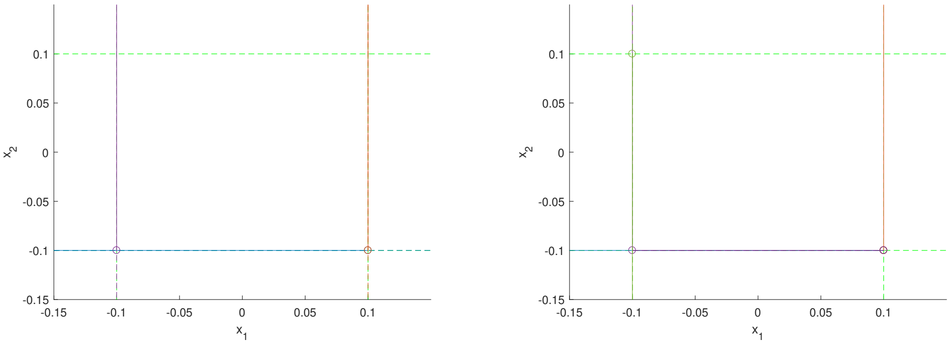


(e) Iteration 5

(f) Iteration 6

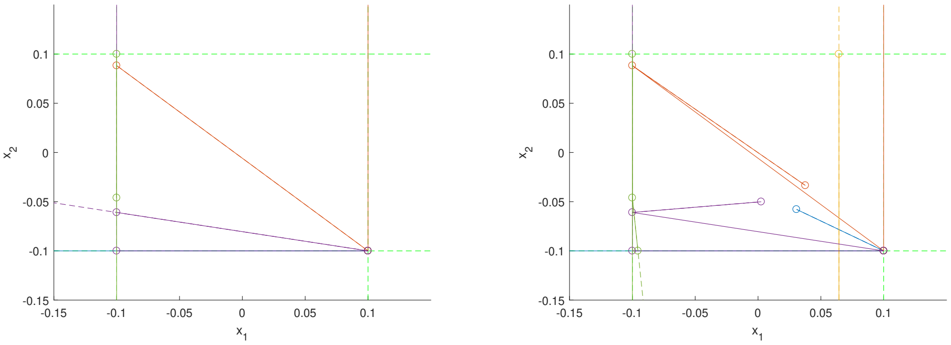
───	Flight path
- - -	Velocity vector
- - -	Constraint boundary

Figure 7.8: Progression of particles in the 2D design space (iterations 1 to 6)



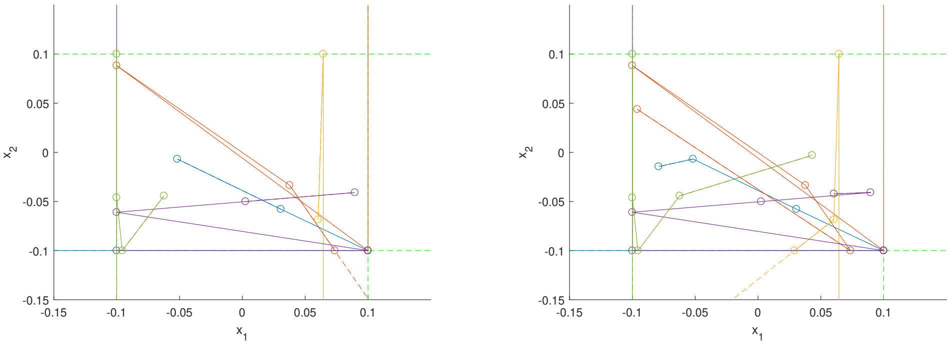
(a) Iteration 2

(b) Iteration 3



(c) Iteration 4

(d) Iteration 5



(e) Iteration 6

(f) Iteration 7

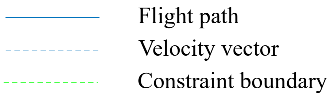
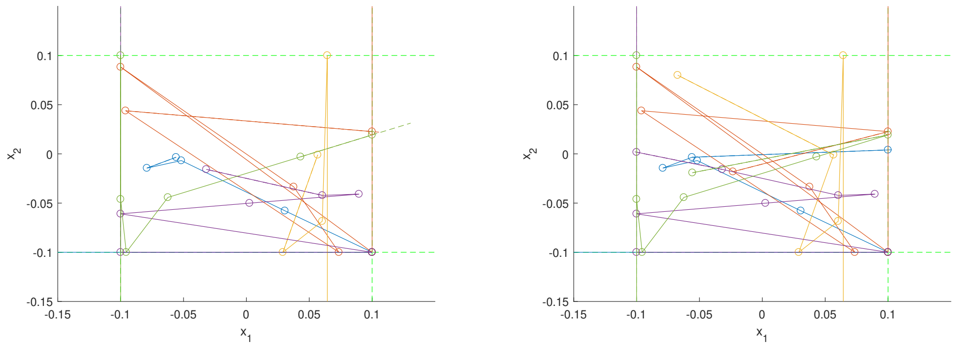
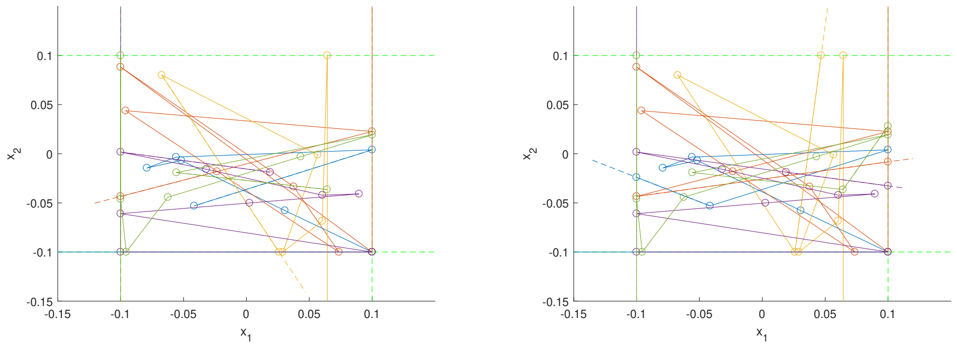


Figure 7.9: Progression of particles in the 2D feasible region (iterations 2 to 7)



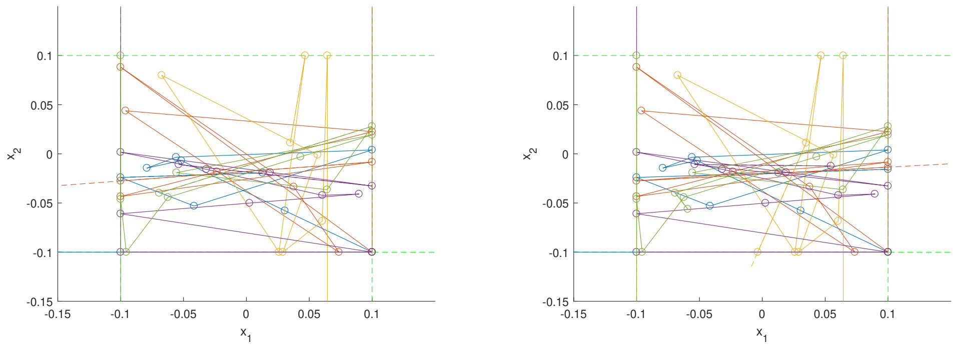
(a) Iteration 8

(b) Iteration 9



(c) Iteration 10

(d) Iteration 11



(e) Iteration 12

(f) Iteration 13

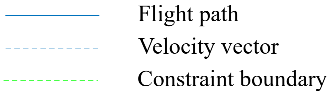
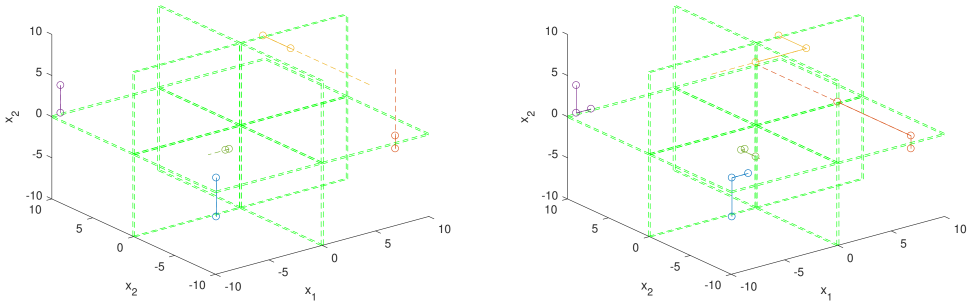
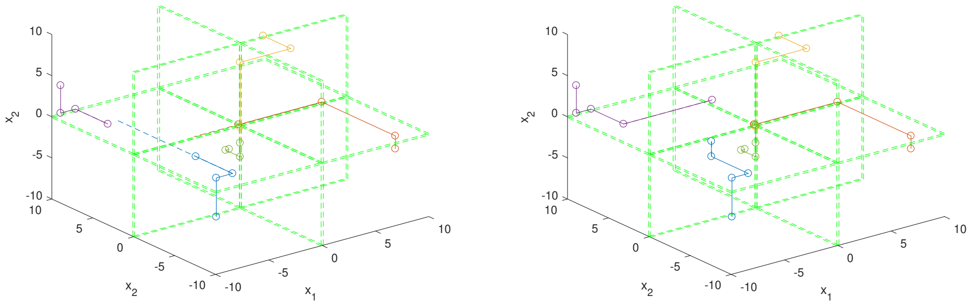


Figure 7.10: Progression of particles in the 2D feasible region (iterations 8 to 13)



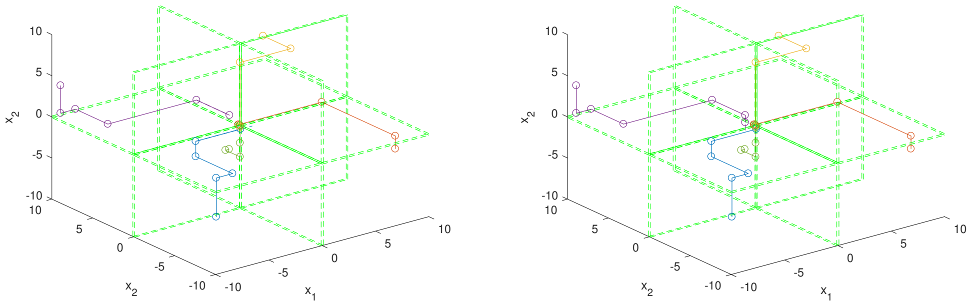
(a) Iteration 1

(b) Iteration 2



(c) Iteration 3

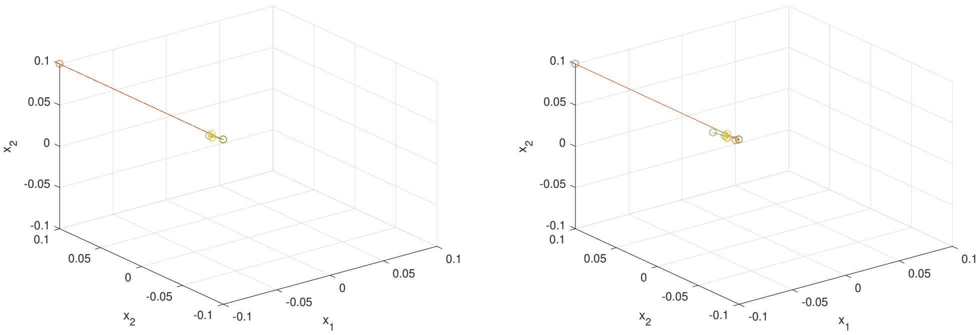
(d) Iteration 4



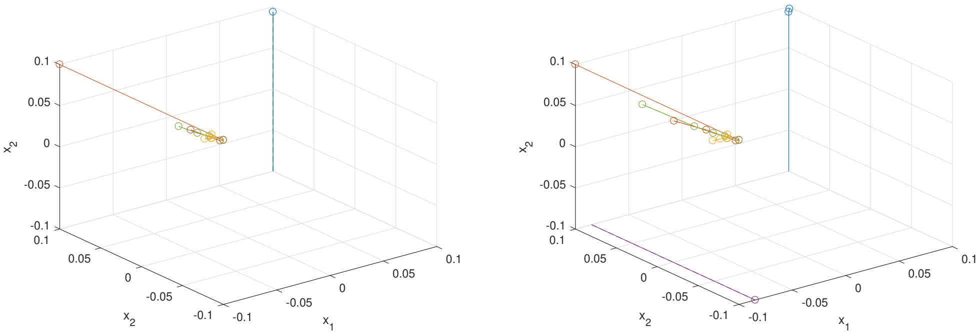
(e) Iteration 5

(f) Iteration 6

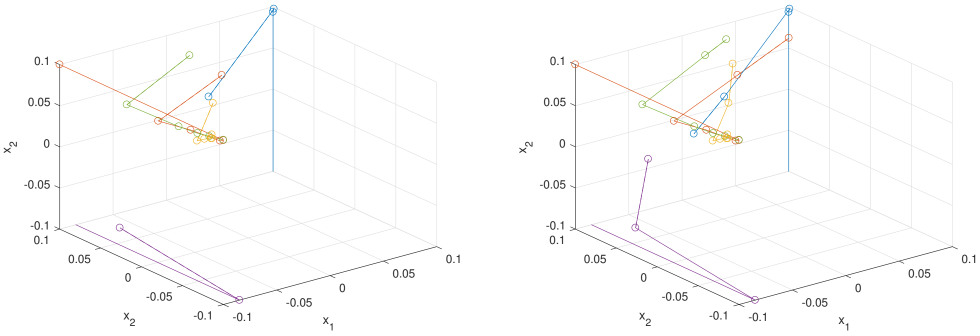
Figure 7.11: Progression of particles in the 3d design space (iterations 1 to 6)



(a) Iteration 5 (b) Iteration 6

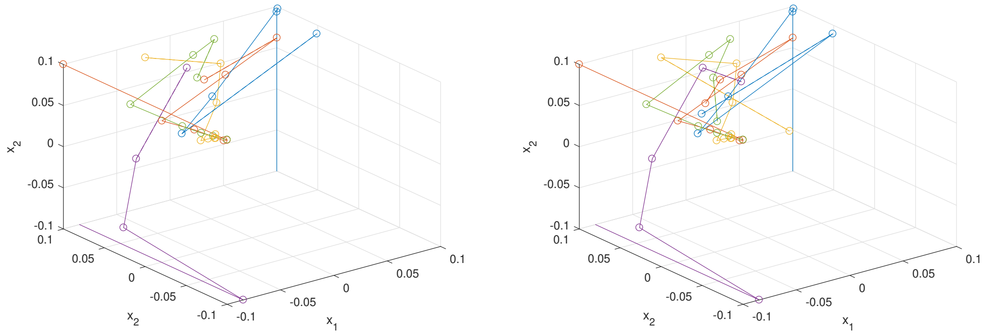


(c) Iteration 7 (d) Iteration 8



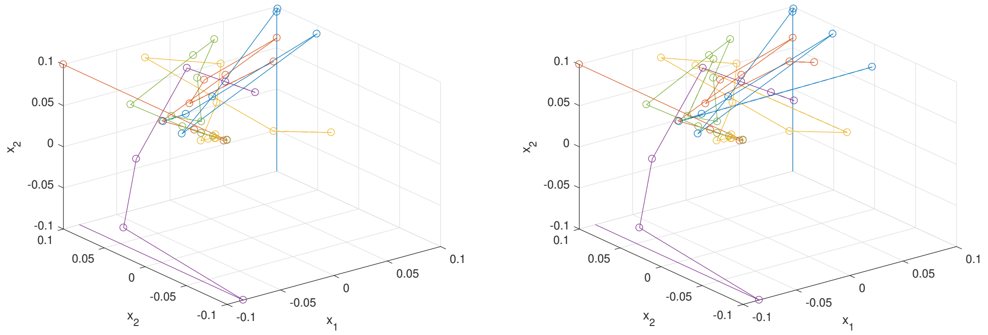
(e) Iteration 9 (f) Iteration 10

Figure 7.12: Progression of particles in the 3D feasible region (iterations 5 to 10)



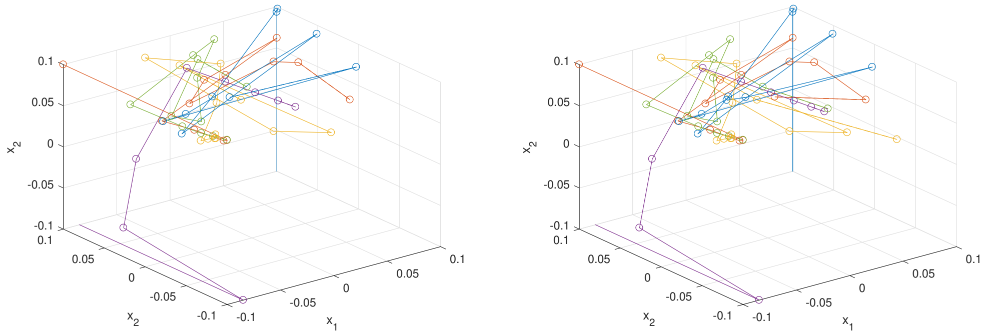
(a) Iteration 11

(b) Iteration 12



(c) Iteration 13

(d) Iteration 14



(e) Iteration 15

(f) Iteration 16

Figure 7.13: Progression of particles in the 3D feasible region (iterations 11 to 16)

when they are not violated are handled more efficiently by FEPSO. To illustrate the effects of this information on the optimization process the RC-MOG problem introduced in Section 5.2 was used. A boolean constraint version of the problem was defined where each constraint function was converted such that

$$g_n^*(\mathbf{x}) = \begin{cases} 1, & g_n(\mathbf{x}) > 0 \\ 0, & g_n(\mathbf{x}) \leq 0 \end{cases} \tag{7.13}$$

This modification essentially removes the information related with how far a point may be from each constraint boundary. When the boolean constrained RC-MOG was solved with MOFEPSO with a swarm size of 200 and maximum iterations of 200, no feasible solutions were found. Whereas, MOFEPSO with the same algorithm parameters is able to find good solutions to the original RC-MOG problem. Both the original- and boolean constrained versions of the problem were solved with FEPSO five times after setting the maximum iterations to 400. While FEPSO found feasible solutions for the original RC-MOG problem in all five of the runs, it was able to find feasible solutions in only three of the boolean constrained runs. A comparison of non-dominated solutions found in these runs are given in Figure 7.14. As seen in this figure, results found for the original RC-MOG problem are significantly better when compared to the boolean constrained version. All these results point to the fact that more iterations are required when there is less information related with constraints or in the case of implicit constraint functions.

The importance of constraint function characteristics were further examined using the artificial hypercube problem. Since the hypercube problem is a single-objective problem, it is easier to statistically compare results obtained by different approaches. Once again a boolean constrained version of the problem was defined by Eqn. 7.13. Number of decision variables was taken as 10 ($N = 10$ and $M = 2N$) while swarm size and maximum iterations were both set to 100. When results of 20 runs for the each problem variant were examined, solutions found for the boolean constrained version were found to be significantly worse than the solutions of the original hypercube problem ($p < 0.0001$)[6]. This result also indicates to the fact that more iterations are required as less information is provided by the constraint functions. Another conclusion is that if decomposition of some complex constraints are possible, FEPSO

---

[6] Nonparametric comparisons were performed with Wilcoxon method [164].

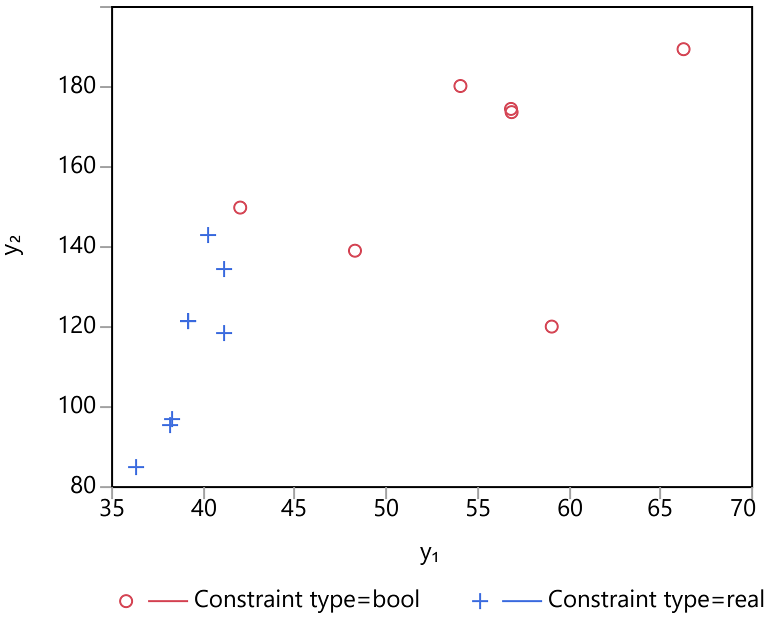


Figure 7.14: Comparison of boolean or real valued constraint functions for the RC-MOG problem (swarm size: 200, maximum iterations: 400, $n = 5$)

performance can be improved.

The hypercube problem was also utilized to observe the variation of number of constraint violations in the swarm as iterations progress. Three different number of variables ($N = 20, 25, 30$) were used in FEPSO runs with swarm sizes of 50 and 100. As seen in Figure 7.15, number of constraint violations rapidly reduce down to a fully satisfying swarm.

7.4 Closure

First section of this chapter describes the analysis performed in order to reveal the time complexity characteristics of MOFEPSO. Important differences that cause additional computational burden when compared with canonical PSO are explained. Main source of additional complexity introduced by MOFEPSO arises from the CFEs required by the VBS algorithm. However, when the relation of time complexity with swarm size and number of decision variables is examined a high order increase in complexity is not observed.

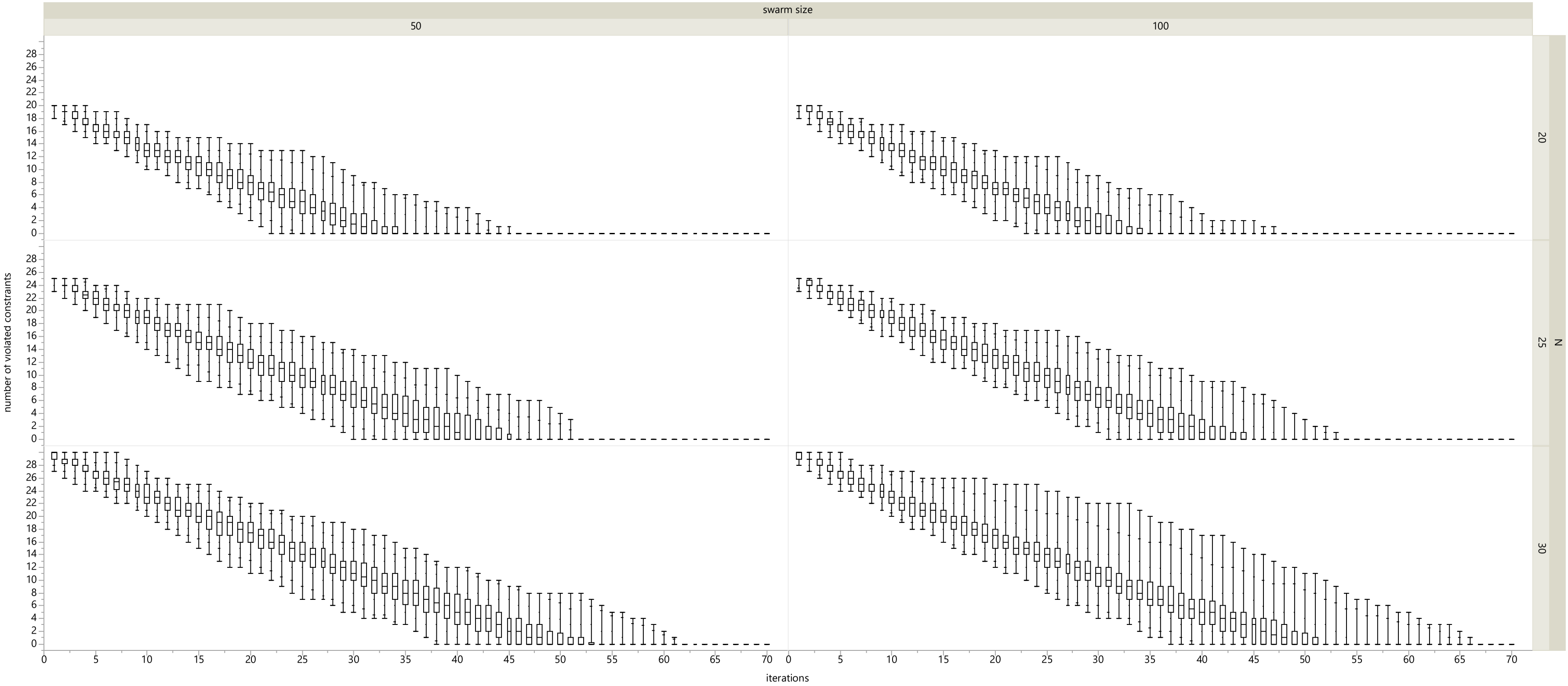


Figure 7.15: Variation of number of constraint violations as iterations progress in hypercube problem of different dimensions ($N = 20, 25, 30$)

Second part of the chapter puts forward an extensive comparison of single-objective and multi-objective approaches. Each of the multi-objective problems RC-MOG, MV-MOG, and SRN were solved 50 times with the MOFEPSO algorithm. To be able to compare with the results obtained, these problems were converted to single objective problems using three different fixed weight aggregation functions (y^1, y^2, and y^3) and solved with FEPSO. When results obtained by single- and multi objective algorithms were compared with respect to y^1, y^2, and y^3 criteria, single-objective solutions were not superior. Therefore, solutions obtained with MOFEPSO in a single run were found to be either equivalent to or better than the results obtained by separate FEPSO runs performed for different aggregation functions. Moreover, results obtained by MOFEPSO (i.e. the Pareto set) were observed to contain solutions that correspond to optimum values of different weight combinations of objectives. Therefore:

- Multi-objective optimization runs performed without assigning fixed weights to objective functions reveal solutions that virtually correspond to any weight combination.
- If sufficient information related with the problem does not exist, one should refrain from assigning fixed weights to objectives in order to employ a single-objective optimization algorithm.
- Due to the fact that the results of a single multi-objective optimization run can correspond to multiple single-objective runs, multi-objective approach is found to be more efficient. In other words, multi-objective algorithms can offer considerable advantage in terms of computational cost. This advantage may especially become important in the case of complex objective functions such as the finite element method.

In the final section the particle motion characteristics of FEPSO (and MOFEPSO) were examined. The mechanisms contributing fast convergence to feasible regions were discussed. The importance of cyclic constraint activation for infeasible particles was emphasized. Finally, the effects of the nature of constraint functions were addressed and decomposition of implicit constraint functions was pointed out as a potential approach for improving algorithm performance.

CHAPTER 8

CONCLUSION

This study presents the FEPSO method and its multi-objective counterpart MOFEPSO to solve constrained optimization problems in engineering design. This new approach treats objective functions and constraint functions separately. The proposed method does not require objective function evaluation at infeasible points. Moreover, it does not require the swarm to have feasible particles when initialized. It performs satisfactorily when the problem is highly-constrained and progressively improves the level of constraint satisfaction in the swarm at each iteration. Some key attributes of (MO)FEPSO can be summarized as follows:

- (MO)FEPSO does not require any feasible positions to exist in the initialized swarm. It only requires the presence of at least one non-violating particle for each constraint.
- Infeasible particles fly to become feasible. Therefore, they activate a violated constraint and become socially attracted to a particle that satisfies the activated constraint (AC). AC is selected based on constraint priorities and number of times constraints have been selected as the AC.
- If a particle is in a feasible position, it is not allowed to fly into an infeasible position.
- An infeasible particle is not allowed to fly into a position that violates one of its previously satisfied constraints.
- Flight of infeasible particles only occur along dimensions to which AC is sensitive to.

- Objective functions are only evaluated for feasible positions.

All of these features help (MO)FEPSO to gradually increase overall feasibility of the swarm and finally attain feasible solutions.

Rigorous simulations have been performed to compare FEPSO and MOFEPSO with other popular algorithms. Additionally, certain characteristics of the algorithms such as the design space exploration capability and the time complexity were also investigated. Simulation results suggest that FEPSO and MOFEPSO are efficiently able to solve single and multi-objective constrained optimization problems even if the feasible portion of the design space is extremely small. Furthermore, in comparisons performed, FEPSO and MOFEPSO were found to perform better than the competing algorithms. All problems[1] used in comparative studies in this dissertation are listed in Table 8.1 together with the algorithms that have been applied.

Table 8.1: All problems and algorithms used in the dissertation

	FEPSO	DOPSO	HPSO	GA	MOFEPSO	MOGA2	NSGA2	MOPSO
GTP	×	×	×	×		×	×	
RC-GTP	×	×	×			×	×	
MV-GTP	×	×	×	×		×	×	
Heat Pipe	×				×	×	×	
C14					×			
MOG					×	×	×	×
RC-MOG					×	×	×	×
MV-MOG					×	×	×	×
SRN	×				×			

Following points stand out as the main contributions of this work.

- A new evolutionary optimization algorithm and its multi-objective version have been developed.
  - The algorithms are capable of solving highly-constrained problems with a high success rate and consistency.

---

[1] Benchmark problems used for initial assessments in Section 3.8 are not included.

  - The algorithms were shown to progressively enhance a population's feasibility as a whole.
  - When applied to less constrained problems, the developed method was found to perform satisfactorily. Hence, usage is not limited to highly-constrained problems and utilization as a general purpose constrained optimizer is practical.
  - Solutions found by the algorithms were statistically better than its counterparts in the performed simulations. The time complexity of the algorithm was also found to be reasonable during these simulations.
- The virtual boundary search (VBS) and the concept of virtual boundaries in constraint handling were devised as a part of FEPSO development work. This innovative approach provides an efficient way to handle constraints especially for hard-to-satisfy constraint functions.
- A methodology for comparison of multi-objective- and single-objective approaches is formed. Results support the usage of multi-objective algorithms (such as MOFEPSO) in the presence of multiple objective functions.
- Several mechanical design problems have been formed. These problems can be utilized as test problems for constrained optimizers.
  - Four-stage gear train problem and its variants are especially suitable as highly-constrained optimizer testbeds.
  - The multi-objective version of the heat pipe design problem (see Section 6.2) is proposed within this study. This approach has certain advantages considering its ability to provide a multitude of non-dominated solutions.

8.1 Future work

The following points may provide insight to some future studies.

- VBS can be employed in many evolutionary algorithms that have a notion of movement in the decision hyper-space (such as the artificial bee colony or gravitational search algorithm). More importantly, special swarm intelligence algorithms could be developed to exploit capabilities of VBS.

- Prioritization of constraints play a critical role in FEPSO. Categorization of constraints with respect to their difficulties can also be useful. In that case, some trivial constraints can be left out of mechanisms driving infeasible particle flight and be included in the objective optimization flights. This modification can offer a substantial advantage in terms of computational burden. Moreover, automation of the prioritization and priority-aware categorization of constraints may bring significant benefits, and hence, are worth studying.

- The techniques developed in this study could find applications in many areas including new topics such as topology optimization and target shape design optimization.

- Hybridization is becoming more widely spread in the optimization literature. Hybridization of FEPSO (or MOFEPSO) with other compatible algorithms may provide formidable strengths.

- Although FEPSO and MOFEPSO do not introduce many algorithm parameters, they inherit parameters from the canonical PSO. These parameters influence particle behavior, and therefore, are factors of the algorithm performance. PSO is inspired from behavioral patterns and social interactions observed in animal swarms of nature. In reality, these patterns both influence and are influenced by the underlying evolutionary process. Therefore, evolving particles that adapt to their environment (the problem of interest in the context of evolutionary optimization) and adjust their behaviors may offer an interesting research topic.

REFERENCES

[1] Alfredo Arias-Montano, Carlos A. Coello Coello, and Efrén Mezura-Montes. “Evolutionary algorithms applied to multi-objective aerodynamic shape optimization”. In: *Computational Optimization, Methods and Algorithms*. Springer, 2011, pp. 211–240.

[2] Carlos A. Coello Coello. “Use of a self-adaptive penalty approach for engineering optimization problems”. In: *Computers in Industry* 41.2 (Mar. 2000), pp. 113–127.

[3] Kalyanmoy Deb and Mayank Goyal. “A Combined Genetic Adaptive Search (GeneAS) for Engineering Design”. In: *Computer Science and Informatics* 26 (1996), pp. 30–45.

[4] Mehmet Sinan Hasanoglu and Melik Dolen. “Feasibility enhanced particle swarm optimization for constrained mechanical design problems”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 232.2 (2018), pp. 381–400.

[5] Mehmet Sinan Hasanoglu and Melik Dolen. “Multi-objective feasibility enhanced particle swarm optimization”. In: *Engineering Optimization* 50.12 (Feb. 2018), pp. 2013–2037.

[6] Kalyanmoy Deb. “Practical Optimization Using Evolutionary Methods”. In: *International Workshop on Neural Networks and Genetic Algorithm in Material Science and Engineering, New Delhi*. 2006, pp. 26–43.

[7] Massimiliano Caramia and Paolo Dell’Olmo. *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms*. Springer Science & Business Media, 2008.

[8] Aimin Zhou et al. “Multiobjective evolutionary algorithms: A survey of the state of the art”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49.

[9] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[10] Randy L. Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms*. Wiley-Interscience, 2004.

[11] E. J. Borowski and Jonathan M. Borwein. *Harpercollins Dictionary of Mathematics*. Collins reference. Harpercollins, 1991.

[12] Singiresu S. Rao. *Engineering Optimization: Theory and Practice*. JOHN WILEY & SONS INC, July 11, 2009. 813 pp.

[13] Marshall K. Wood and George B. Dantzig. “Programming of Interdependent Activities: I General Discussion”. In: *Econometrica* 17.3/4 (July 1949), p. 193.

[14] George B. Dantzig. “Programming of Interdependent Activities: II Mathematical Model”. In: *Econometrica* 17.3/4 (July 1949), p. 200.

[15] J. A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313.

[16] M. J. Box. “A Comparison of Several Current Optimization Methods, and the use of Transformations in Constrained Problems”. In: *The Computer Journal* 9.1 (May 1966), pp. 67–77.

[17] A. Cauchy. “Methode generale pour la resolution des systemes d’equations simultanees”. In: *C.R. Acad. Sci. Paris* 25 (1847), pp. 536–538.

[18] M. J. D. Powell. “An efficient method for finding the minimum of a function of several variables without calculating derivatives”. In: *The Computer Journal* 7.2 (Feb. 1964), pp. 155–162.

[19] C. G. Broyden. “A class of methods for solving nonlinear simultaneous equations”. In: *Mathematics of Computation* 19.92 (1965), pp. 577–577.

[20] R. Fletcher. “Generalized inverses for nonlinear equations and optimization”. In: *Numerical Methods for Non-linear Algebraic Equations*. Ed. by R. Rabinowitz. 1963.

[21] Donald Goldfarb and Leon Lapidus. “Conjugate Gradient Method for Nonlinear Programming Problems with Linear Constraints”. In: *Industrial & Engineering Chemistry Fundamentals* 7.1 (Feb. 1968), pp. 142–151.

[22] D. F. Shanno. “An Accelerated Gradient Projection Method for Linearly Constrained Nonlinear Estimation”. In: *SIAM Journal on Applied Mathematics* 18.2 (1970), pp. 322–334.

[23] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming (International Series in Operations Research & Management Science)*. Springer, 2008.

[24] J. David Schaffer. “Multiple objective optimization with vector evaluated genetic algorithms.” In: *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985*. 1985, pp. 93–100.

[25] John H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[26] Nidamarthi Srinivas and Kalyanmoy Deb. “Muiltiobjective optimization using nondominated sorting in genetic algorithms”. In: *Evolutionary computation* 2.3 (1994), pp. 221–248.

[27] Kalyanmoy Deb et al. “A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II”. In: *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 849–858.

[28] Kalyanmoy Deb et al. “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[29] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680.

[30] Rainer Storn and Kenneth Price. “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces”. In: *Journal of Global Optimization* 11.4 (Dec. 1997), pp. 341–359.

[31] Christian von Lücken, Benjamín Barán, and Carlos Brizuela. “A survey on multi-objective evolutionary algorithms for many-objective problems”. en. In: *Computational Optimization and Applications* 58.3 (Feb. 2014), pp. 707–756.

[32] C. A. C. Coello. “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art”. In: *Computer Methods in Applied Mechanics and Engineering* 191.11 (2002), pp. 1245–1287.

[33] Efrén Mezura-Montes and Carlos A. Coello Coello. “Constraint-handling in nature-inspired numerical optimization: Past, present and future”. In: *Swarm and Evolutionary Computation* 1.4 (2011), pp. 173–194.

[34] Laurence Giraud-Moreau and Pascal Lafon. “A Comparison of Evolutionary Algorithms for Mechanical Design Components”. In: *Engineering Optimization* 34.3 (Jan. 2002), pp. 307–322.

[35] Gustavo R. Zavala et al. “A survey of multi-objective metaheuristics applied to structural optimization”. In: *Structural and Multidisciplinary Optimization* 49.4 (Oct. 2013), pp. 537–558.

[36] Alfredo Arias-Montano, Carlos A. Coello Coello, and Efrén Mezura Montes. “Multiobjective evolutionary algorithms in aeronautical and aerospace engineering”. In: *Evolutionary Computation, IEEE Transactions on* 16.5 (2012), pp. 662–694.

[37] Mathieu Balesdent et al. “A survey of multidisciplinary design optimization methods in launch vehicle design”. en. In: *Structural and Multidisciplinary Optimization* 45.5 (May 2012), pp. 619–642.

[38] Christian Horoba and Frank Neumann. “Approximating Pareto-Optimal Sets Using Diversity Strategies in Evolutionary Multi-Objective Optimization”. In: *Advances in Multi-Objective Nature Inspired Computing*. Springer Berlin Heidelberg, 2010, pp. 23–44.

[39] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. “Evaluating the ϵ-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions”. In: *Evolutionary Computation* 13.4 (Dec. 2005), pp. 501–525.

[40] K.I. Smith et al. “Dominance-Based Multiobjective Simulated Annealing”. In: *IEEE Transactions on Evolutionary Computation* 12.3 (June 2008), pp. 323–342.

[41] E. Aggelogiannaki and H. Sarimveis. “A Simulated Annealing Algorithm for Prioritized Multiobjective Optimization—Implementation in an Adaptive Model Predictive Control Configuration”. In: *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* 37.4 (Aug. 2007), pp. 902–915.

[42] Melanie Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press, 1996.

[43] Carlos M. Fonseca and Peter J. Fleming. “Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization”. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 416–423.

[44] J. Horn, N. Nafpliotis, and D. E. Goldberg. “A niched Pareto genetic algorithm for multiobjective optimization”. In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE, 1994.

[45] R. Eberhart and J. Kennedy. “A new optimizer using particle swarm theory”. In: *MHS 95., Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995*. Oct. 1995, pp. 39–43.

[46] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of IEEE International Conference on Neural Networks, 1995*. Vol. 4. Nov. 1995, 1942–1948 vol.4.

[47] D. Bratton and J. Kennedy. “Defining a Standard for Particle Swarm Optimization”. In: *2007 IEEE Swarm Intelligence Symposium*. 2007 IEEE Swarm Intelligence Symposium. Apr. 2007, pp. 120–127.

[48] Soniya Lalwani et al. “A comprehensive survey: Applications of multi-objective particle swarm optimization (MOPSO) algorithm”. In: *Transactions on Combinatorics* 2.1 (2013), pp. 39–101.

[49] Riccardo Poli. “Analysis of the Publications on the Applications of Particle Swarm Optimisation”. In: *J. Artif. Evol. App.* 2008 (Jan. 2008), 4:1–4:10.

[50] Riccardo Poli, James Kennedy, and Tim Blackwell. “Particle swarm optimization”. In: *Swarm Intelligence* 1.1 (June 2007), pp. 33–57.

[51] Margarita Reyes-sierra and Carlos A. Coello Coello. “Multi-objective particle swarm optimizers: A survey of the state-of-the-art”. In: *International Journal of Computational Intelligence Research* 2.3 (2006), pp. 287–308.

[52] Eberhart and Yuhui Shi. “Particle swarm optimization: developments, applications and resources”. In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. Vol. 1. 2001, 81–86 vol. 1.

[53] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. “A review of particle swarm optimization. Part I: background and development”. In: *Natural Computing* 6.4 (Dec. 2007), pp. 467–484.

[54] Alec Banks, Jonathan Vincent, and Chukwudi Anyakoha. “A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications”. In: *Natural Computing* 7.1 (Mar. 2008), pp. 109–124.

[55] R. Mendes, J. Kennedy, and J. Neves. “The fully informed particle swarm: simpler, maybe better”. In: *IEEE Transactions on Evolutionary Computation* 8.3 (June 2004), pp. 204–210.

[56] C. A. Coello Coello and M. S. Lechuga. “MOPSO: a proposal for multiple objective particle swarm optimization”. In: *Evolutionary Computation, 2002. CEC ’02. Proceedings of the 2002 Congress on*. Vol. 2. 2002, pp. 1051–1056.

[57] Tapabrata Ray, Kang Tai, and Kin Chye Seow. “Multiobjective design optimization by an evolutionary algorithm”. In: *Engineering Optimization* 33.4 (2001), pp. 399–424.

[58] Kalyanmoy Deb. “An efficient constraint handling method for genetic algorithms”. In: *Computer Methods in Applied Mechanics and Engineering* 186.2 (2000), pp. 311–338.

[59] T. Takahama and S. Sakai. “Constrained Optimization by the ε Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites”. In: *2006 IEEE International Conference on Evolutionary Computation*. July 2006, pp. 1–8.

[60] K. E. Parsopoulos and M. N. Vrahatis. “Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 582–591.

[61] J. Zheng, Q. Wu, and W. Song. “An Improved Particle Swarm Algorithm for Solving Nonlinear Constrained Optimization Problems”. In: *Third International Conference on Natural Computation (ICNC 2007)*. Vol. 4. Aug. 2007, pp. 112–117.

[62] Tetsuyuki Takahama and Setsuko Sakai. “Constrained Optimization by Combining the α Constrained Method with Particle Swarm Optimization”. In: *Proc. of Joint 2nd International Conference on Soft Computing and Intelligent Systems and 5th International Symposium on Advanced Intelligent Systems*. 2004.

[63] L. Jian, C. Peng, and L. Zhiming. “Solving Constrained Optimization via Dual Particle Swarm Optimization with Stochastic Ranking”. In: *2008 International Conference on Computer Science and Software Engineering*. Vol. 1. Dec. 2008, pp. 1215–1218.

[64] E. Nasr Azadani, S.H. Hosseinian, and B. Moradzadeh. “Generation and reserve dispatch in a competitive market using constrained particle swarm optimization”. In: *International Journal of Electrical Power & Energy Systems* 32.1 (2010), pp. 79–86.

[65] Haiyan Lu and Weiqi Chen. “Dynamic-objective particle swarm optimization for constrained optimization problems”. In: *Journal of Combinatorial Optimization* 12.4 (Dec. 2006), pp. 409–419.

[66] A. Rezaee Jordehi. “A review on constraint handling strategies in particle swarm optimisation”. In: *Neural Computing and Applications* 26.6 (Aug. 2015), pp. 1265–1275.

[67] Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend. “A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 459–473.

[68] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga. “Handling multiple objectives with particle swarm optimization”. In: *IEEE Transactions on Evolutionary Computation* 8.3 (June 2004), pp. 256–279.

[69] Jonathan E. Fieldsend. *Multi-Objective Particle Swarm Optimisation Methods*. Report. University of Exeter, 2004.

[70] J. Kennedy and R. C. Eberhart. “A discrete binary version of the particle swarm algorithm”. In: *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*. Vol. 5. Oct. 1997, 4104–4108 vol.5.

[71] Maurice Clerc. “Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem”. In: *New Optimization Techniques in Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 219–239.

[72] Quan-Ke Pan, M. Fatih Tasgetiren, and Yun-Chia Liang. “A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem”. In: *Computers & Operations Research* 35.9 (2008). Part Special Issue: Bio-inspired Methods in Combinatorial Optimization, pp. 2807–2839.

[73] Ali Husseinzadeh Kashan and Behrooz Karimi. “A discrete particle swarm optimization algorithm for scheduling parallel machines”. In: *Computers & Industrial Engineering* 56.1 (2009), pp. 216–223.

[74] Wei Pang et al. “Fuzzy discrete particle swarm optimization for solving traveling salesman problem”. In: *CIT ’04. The Fourth International Conference on Computer and Information Technology*. Sept. 2004, pp. 796–800.

[75] Elizabeth F. Gouvêa Goldbarg, Givanaldo R. de Souza, and Marco César Goldbarg. “Particle Swarm for the Traveling Salesman Problem”. In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Jens Gottlieb and Günther R. Raidl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 99–110.

[76] Elizabeth F. G. Goldbarg, Marco C. Goldbarg, and Givanaldo R. de Souza. “Particle Swarm Optimization Algorithm for the Traveling Salesman Problem”. In: *Traveling Salesman Problem*. Ed. by Federico Greco. InTech, Sept. 1, 2008, pp. 75–96.

[77] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. “Particle swarm optimization for integer programming”. In: *Evolutionary Computation, 2002. CEC ’02. Proceedings of the 2002 Congress on*. Vol. 2. 2002, pp. 1582–1587.

[78] Leandro dos Santos Coelho. “An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications”. In: *Reliability Engineering & System Safety* 94.4 (2009), pp. 830–837.

[79] R. E. Perez and K. Behdinan. “Particle swarm approach for structural design optimization”. In: *Computers & Structures* 85.19-20 (Oct. 2007), pp. 1579–1588.

[80] M. Sunar and A. D. Belegundu. “Trust region methods for structural optimization using exact second order sensitivity”. In: *International Journal for Numerical Methods in Engineering* 32.2 (Aug. 1991), pp. 275–293.

[81] Timothy Ganesan, Pandian Vasant, and Irraivan Elamvazuthy. “A hybrid PSO approach for solving non-convex optimization problems”. In: *Archives of Control Sciences* 22.1 (Jan. 2012), pp. 87–105.

[82] M. Fesanghary, E. Damangir, and I. Soleimani. “Design optimization of shell and tube heat exchangers using global sensitivity analysis and harmony search algorithm”. In: *Applied Thermal Engineering* 29.5-6 (Apr. 2009), pp. 1026–1031.

[83] Takao Yokota, Takeaki Taguchi, and Mitsuo Gen. “A solution method for optimal weight design problem of the gear using genetic algorithms”. In: *Computers & Industrial Engineering* 35.3-4 (Dec. 1998). Selected Papers from the 22nd ICC and IE Conference, pp. 523–526.

[84] Robert C. Juvinall and Kurt M. Marshek. *Fundamentals of machine component design*. English. Hoboken, NJ: John Wiley & Sons, 2012.

[85] Robert L. Norton. *Machine design: an integrated approach*. eng. 5th ed. Boston, Mass.: Prentice Hall, 2014.

[86] R. Venkata Rao and Vimal J. Savsani. *Mechanical Design Optimization Using Advanced Optimization Techniques*. Springer Series in Advanced Manufacturing. London: Springer London, 2012.

[87] V. Savsani, R. V. Rao, and D. P. Vakharia. “Optimal weight design of a gear train using particle swarm optimization and simulated annealing algorithms”. In: *Mechanism and Machine Theory* 45.3 (Mar. 2010), pp. 531–541.

[88] Shantanu Gupta, Rajiv Tiwari, and Shivashankar B. Nair. “Multi-objective design optimisation of rolling bearings using genetic algorithms”. In: *Mechanism and Machine Theory* 42.10 (Oct. 2007), pp. 1418–1443.

[89] B. Rajeswara Rao and Rajiv Tiwari. “Optimum design of rolling element bearings using genetic algorithms”. In: *Mechanism and Machine Theory* 42.2 (Feb. 2007), pp. 233–250.

[90] Jinhao Zhang et al. “Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems”. In: *Applied Mathematical Modelling* 63 (Nov. 2018), pp. 464–490.

[91] Andrzej Osyczka. *Evolutionary algorithms for single and multicriteria design optimization ; with 24 tables*. eng. Studies in fuzziness and soft computing Vol. 79. Heidelberg: Physica-Verl, 2002.

[92] Andrzej Osyczka, Stanislaw Krenich, and K. Karas. “Optimum design of robot grippers using genetic algorithms”. In: *Proceedings of the Third World Congress of Structural and Multidisciplinary Optimization (WCSMO), Buffalo, New York*. 1999, pp. 241–243.

[93] L. P. Pomrehn and P. Y. Papalambros. “Discrete Optimal Design Formulations With Application to Gear Train Design”. In: *Journal of Mechanical Design* 117.3 (1995), pp. 419–424.

[94] L. P. Pomrehn and P. Y. Papalambros. “Infeasibility and Non-Optimality Tests for Solution Space Reduction in Discrete Optimal Design”. In: *Journal of Mechanical Design* 117.3 (1995), pp. 425–432.

[95] Jinhua Wang and Zeyong Yin. “A ranking selection-based particle swarm optimizer for engineering design optimization problems”. In: *Structural and Multidisciplinary Optimization* 37.2 (Jan. 2008), pp. 131–147.

[96] E. Khorshid and A. Seireg. “Discrete nonlinear optimisation by constraint decomposition and designer interaction”. In: *International Journal of Computer Applications in Technology* 12.2/3/4/5 (1999), pp. 233–244.

[97] V. J. Savsani, R. V. Rao, and D. P. Vakharia. “Discrete optimisation of a gear train using biogeography based optimisation technique”. In: *International Journal of Design Engineering* 2.2 (2009), pp. 205–223.

[98] M. Dolen, H. Kaplan, and A. Seireg. “Discrete parameter-nonlinear constrained optimisation of a gear train using genetic algorithms”. In: *International Journal of Computer Applications in Technology* 24.2 (2005), pp. 110–121.

[99] B. K. Kannan and S. N. Kramer. “An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design”. In: *Journal of Mechanical Design* 116.2 (1994), pp. 405–411.

[100] Fu-zhuo Huang, Ling Wang, and Qie He. “An effective co-evolutionary differential evolution for constrained optimization”. In: *Applied Mathematics and Computation* 186.1 (Mar. 2007), pp. 340–356.

[101] Qie He and Ling Wang. “An effective co-evolutionary particle swarm optimization for constrained engineering design problems”. In: *Engineering Applications of Artificial Intelligence* 20.1 (Feb. 2007), pp. 89–99.

[102] Bahriye Akay and Dervis Karaboga. “Artificial bee colony algorithm for large-scale problems and engineering design optimization”. In: *Journal of Intelligent Manufacturing* 23.4 (Aug. 2012), pp. 1001–1014.

[103] E. Mezura-Montes and C. A. C. Coello. “A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems”. In: *IEEE Transactions on Evolutionary Computation* 9.1 (Feb. 2005), pp. 1–17.

[104] Hui Liu, Zixing Cai, and Yong Wang. “Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization”. In: *Applied Soft Computing* 10.2 (Mar. 2010), pp. 629–640.

[105] E. Sandgren. “Nonlinear Integer and Discrete Programming in Mechanical Design Optimization”. In: *Journal of Mechanical Design* 112.2 (1990), pp. 223–229.

[106] Tapabrata Ray and K. M. Liew. “Society and civilization: an optimization algorithm based on the simulation of social behavior”. In: *IEEE Transactions on Evolutionary Computation* 7.4 (Aug. 2003), pp. 386–396.

[107] Leandro dos Santos Coelho and Viviana Cocco Mariani. “Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization”. In: *Expert Systems with Applications* 34.3 (Apr. 2008), pp. 1905–1913.

[108] K. M. Ragsdell and D. T. Phillips. “Optimal Design of a Class of Welded Structures Using Geometric Programming”. In: *Journal of Engineering for Industry* 98.3 (1976), p. 1021.

[109] Ashok D. Belegundu and Jasbir S. Arora. “A study of mathematical programmingmethods for structural optimization. Part II: Numerical results”. In: *International Journal for Numerical Methods in Engineering* 21.9 (Sept. 1985), pp. 1601–1623.

[110] P. A. Simionescu, D. Beale, and G. V. Dozier. “Teeth-Number Synthesis of a Multispeed Planetary Transmission Using an Estimation of Distribution Algorithm”. In: *Journal of Mechanical Design* 128.1 (2006), pp. 108–115.

[111] Kalyanmoy Deb and Santosh Tiwari. “Multi-objective optimization of a leg mechanism using genetic algorithms”. In: *Engineering Optimization* 37.4 (June 2005), pp. 325–350.

[112] R. P. Williams, L.-W. Tsai, and S. Azarm. *Design of a Crank-and-rocker Driven Pantograph: a Leg Mechanism for the University of Maryland’s 1991 Walking Robot*. University of Maryland. Systems Research Center, 1991.

[113] C.A. Coello Coello and G.T. Pulido. “Multiobjective structural optimization using a microgenetic algorithm”. In: *Structural and Multidisciplinary Optimization* 30.5 (Nov. 2005), pp. 388–403.

[114] A. Kurpati, S. Azarm, and J. Wu. “Constraint handling improvements for multiobjective genetic algorithms”. In: *Structural and Multidisciplinary Optimization* 23.3 (Apr. 2002), pp. 204–213.

[115] Seyedali Mirjalili, Pradeep Jangir, and Shahrzad Saremi. “Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems”. In: *Applied Intelligence* 46.1 (July 2017), pp. 79–95.

[116] Tapabrata Ray and K.M. Liew. “A Swarm Metaphor for Multiobjective Design Optimization”. In: *Engineering Optimization* 34.2 (Jan. 2002), pp. 141–153.

[117] Xin-She Yang, Mehmet Karamanoglu, and Xingshi He. “Flower pollination algorithm: A novel approach for multiobjective optimization”. In: *Engineering Optimization* 46.9 (Oct. 2014), pp. 1222–1237.

[118] Adarsh Viji Elango et al. “Methods to Find Best Designs Among Infeasible Design Data Sets for Highly Constrained Design Optimization Problems”. In: *SAE International Journal of Materials and Manufacturing* 9.2 (Apr. 2016), pp. 378–386.

[119] H. Saruhan. “Differential evolution and simulated annealing algorithms for mechanical systems design”. In: *Engineering Science and Technology, an International Journal* 17.3 (Sept. 2014), pp. 131–136.

[120] Pascal Lafon. “Conception optimale de systèmes mécaniques : optimisation en variables mixtes”. PhD thesis. Institut national des sciences appliquées de Lyon, 1994.

[121] Tawatchai Kunakote and Sujin Bureerat. “Multi-objective topology optimization using evolutionary algorithms”. In: *Engineering Optimization* 43.5 (May 2011), pp. 541–557.

[122] C. A. Coello and A. D. Christiansen. “Multiobjective optimization of trusses using genetic algorithms”. In: *Computers & Structures* 75.6 (May 2000), pp. 647–660.

[123] Kazuhiro Saitou et al. “A Survey of Structural Optimization in Mechanical Product Development”. In: *Journal of Computing and Information Science in Engineering* 5.3 (2005), pp. 214–226.

[124] Joshua D. Deaton and Ramana V. Grandhi. “Stress-based design of thermal structures via topology optimization”. In: *Structural and Multidisciplinary Optimization* 53.2 (Feb. 2016), pp. 253–270.

[125] B. M. E. de Silva. “Minimum Weight Design of Disks Using a Frequency Constraint”. In: *Journal of Engineering for Industry* 91.4 (1969), p. 1091.

[126] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. “Genetic algorithms and highly constrained problems: The time-table case”. In: *Parallel Problem Solving from Nature*. Springer-Verlag, 1990, pp. 55–59.

[127] T. J. Berna, M. H. Locke, and A. W. Westerberg. “A new approach to optimization of chemical processes”. In: *AIChE Journal* 26.1 (Jan. 1980), pp. 37–43.

[128] F. Mistree, O. F. Hughes, and H. B. Phuoc. “An optimization method for the design of large, highly constrained complex systems”. In: *Engineering Optimization* 5.3 (Jan. 1981), pp. 179–197.

[129] L. Nardin et al. “modeFRONTIER©, a Framework for the Optimization of Military Aircraft Configurations”. In: *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*. Springer Berlin Heidelberg, 2009, pp. 191–205.

[130] T.W. Simpson et al. “Approximation methods in multidisciplinary analysis and optimization: a panel discussion”. In: *Structural and Multidisciplinary Optimization* 27.5 (June 2004).

[131] Evin J. Cramer et al. “Problem Formulation for Multidisciplinary Optimization”. In: *SIAM Journal on Optimization* 4.4 (Nov. 1994), pp. 754–776.

[132] Najeh Ben Guedria. “Improved accelerated PSO algorithm for mechanical engineering optimization problems”. In: *Applied Soft Computing* 40 (Mar. 2016), pp. 455–467.

[133] Zbigniew Michalewicz and Marc Schoenauer. “Evolutionary Algorithms for Constrained Parameter Optimization Problems”. In: *Evolutionary Computation* 4.1 (Mar. 1996), pp. 1–32.

[134] Xiaohui Hu and Russell Eberhart. “Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization”. In: *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002*. 2002, pp. 203–206.

[135] K. E. Parsopoulos and M. N. Vrahatis. “Particle Swarm Optimization Method for Constrained Optimization Problems”. In: *Intelligent Technologies - Theory and Applications: New Trends in Intelligent Technologies*. Ed. by P. Sincak et al. Ios Pr Inc, 2002.

[136] Xiaohui Hu, R. C. Eberhart, and Yuhui Shi. “Engineering optimization with particle swarm”. In: *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS’03 (Cat. No.03EX706)*. IEEE, 2003.

[137] G. Coath and S. K. Halgamuge. “A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems”. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC ’03.* IEEE, 2003.

[138] G. T. Pulido and C. A. C. Coello. “A constraint-handling mechanism for particle swarm optimization”. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753).* IEEE, 2004.

[139] R. V. Rao, V. J. Savsani, and D. P. Vakharia. “Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems”. In: *Computer-Aided Design* 43.3 (Mar. 2011), pp. 303–315.

[140] Wei-Liem Loh. “On Latin hypercube sampling”. In: *The Annals of Statistics* 24.5 (Oct. 1996), pp. 2058–2080.

[141] Qie He and Ling Wang. “A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization”. In: *Applied Mathematics and Computation* 186.2 (Mar. 2007), pp. 1407–1422.

[142] J. Robinson and Y. Rahmat-Samii. “Particle Swarm Optimization in Electromagnetics”. In: *IEEE Transactions on Antennas and Propagation* 52.2 (Feb. 2004), pp. 397–407.

[143] Ali Sadollah et al. “Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems”. In: *Applied Soft Computing* 13.5 (May 2013), pp. 2592–2612.

[144] Erwie Zahara and Yi-Tung Kao. “Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems”. In: *Expert Systems with Applications* 36.2 (Mar. 2009), pp. 3880–3886.

[145] Silvia Poles. *MOGA-II an improved multi-objective genetic algorithm.* Tech. rep. ESTECO, 2003.

[146] Hisao Ishibuchi and Yusuke Nojima. “Optimization of Scalarizing Functions Through Evolutionary Multiobjective Optimization”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 51–65.

[147] Hui Li and Qingfu Zhang. "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 13.2 (Apr. 2009), pp. 284–302.

[148] Bo Xu et al. "Dynamic deployment of virtual machines in cloud computing using multi-objective optimization". In: *Soft Computing* 19.8 (Aug. 2014), pp. 2265–2273.

[149] Gary G. Yen and Zhenan He. "Performance Metric Ensemble for Multiobjective Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 18.1 (Feb. 2014), pp. 131–144.

[150] Michael Farnsworth et al. "An efficient evolutionary multi-objective framework for MEMS design optimisation: validation, comparison and analysis". In: *Memetic Computing* 3.3 (Aug. 2011), pp. 175–197.

[151] Silvia Poles et al. "MOGA-II for an Automotive Cooling Duct Optimization on Distributed Resources". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 633–644.

[152] H.E. Radhi and S.M. Barrans. "Comparison between Multiobjective Population-Based Algorithms in Mechanical Problem". In: *Applied Mechanics and Materials* 110-116 (Oct. 2011), pp. 2383–2389.

[153] Enrico Rigoni and Silvia Poles. "NBI and MOGA-II, two complementary algorithms for Multi-Objective optimizations". In: *Practical Approaches to Multi-Objective Optimization*. Ed. by Jürgen Branke et al. Dagstuhl Seminar Proceedings 04461. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[154] Zai Wang et al. "A multi-objective approach to Redundancy Allocation Problem in parallel-series systems". In: *2009 IEEE Congress on Evolutionary Computation*. IEEE, May 2009.

[155] Vimal Savsani. "HBBABC: A Hybrid Optimization Algorithm Combining Biogeography Based Optimization (BBO) and Artificial Bee Colony (ABC) Optimization For Obtaining Global Solution Of Discrete Design Problems". In: *International Journal Of Computational Engineering Research* 2.7 (2012), pp. 85–97.

[156] Poonam Savsani and Vimal Savsani. “Passing vehicle search (PVS): A novel metaheuristic algorithm”. In: *Applied Mathematical Modelling* 40.5-6 (Mar. 2016), pp. 3951–3978.

[157] Kusum Deep et al. “A real coded genetic algorithm for solving integer and mixed integer optimization problems”. In: *Applied Mathematics and Computation* 212.2 (June 2009), pp. 505–518.

[158] Guillaume Corriveau, Raynald Guilbault, and Antoine Tahan. “Genetic algorithms and finite element coupling for mechanical optimization”. In: *Advances in Engineering Software* 41.3 (Mar. 2010), pp. 422–426.

[159] A. Muc and W. Gurba. “Genetic algorithms and finite element analysis in optimization of composite structures”. In: *Composite Structures* 54.2 (Nov. 2001), pp. 275–281.

[160] Francesco Parasiliti et al. “Finite-Element-Based Multiobjective Design Optimization Procedure of Interior Permanent Magnet Synchronous Motors for Wide Constant-Power Region Operation”. In: *IEEE Transactions on Industrial Electronics* 59.6 (June 2012), pp. 2503–2514.

[161] Miltiadis Kotinis. “A particle swarm optimizer for constrained multi-objective engineering design problems”. In: *Engineering Optimization* 42.10 (Oct. 2010), pp. 907–926.

[162] Ahmad Nourbakhsh, Hamed Safikhani, and Shahram Derakhshan. “The comparison of multi-objective particle swarm optimization and NSGA II algorithm: applications in centrifugal pumps”. In: *Engineering Optimization* 43.10 (Oct. 2011), pp. 1095–1113.

[163] M. Janga Reddy and D. Nagesh Kumar. “An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design”. In: *Engineering Optimization* 39.1 (Jan. 2007), pp. 49–68.

[164] Frank Wilcoxon. “Individual Comparisons by Ranking Methods”. In: *Biometrics Bulletin* 1.6 (Dec. 1945), pp. 80–83.

[165] J. J. Buksa and K. A. Williams. “SPRITE: a computer code for the optimization of space based heat pipe radiator systems”. In: *Proceedings of the 24th Intersociety Energy Conversion Engineering Conference*. IEEE, 1989.

[166] Xiao Ping Wu et al. “Analyzing and modeling on optimized L-ratio of evaporator section to condenser section for micro heat pipe heat sinks”. In: *Ninteenth Annual IEEE Semiconductor Thermal Measurement and Management Symposium, 2003.* IEEE, 2003.

[167] Tian Shen Liang and Yew Mun Hung. “Experimental investigation on the thermal performance and optimization of heat sink with U-shape heat pipes”. In: *Energy Conversion and Management* 51.11 (Nov. 2010), pp. 2109–2116.

[168] Sung Jin Kim, Joung Ki Seo, and Kyu Hyung Do. “Analytical and experimental investigation on the operational characteristics and the thermal optimization of a miniature heat pipe with a grooved wick structure”. In: *International Journal of Heat and Mass Transfer* 46.11 (May 2003), pp. 2051–2063.

[169] Valeri V. Vlassov, Fabiano L. de Sousa, and Walter K. Takahashi. “Comprehensive optimization of a heat pipe radiator assembly filled with ammonia or acetone”. In: *International Journal of Heat and Mass Transfer* 49.23-24 (Nov. 2006), pp. 4584–4595.

[170] Fabiano Luis de Sousa, Valeri Vlassov, and Fernando Manuel Ramos. “Generalized extremal optimization: An application in heat pipe design”. In: *Applied Mathematical Modelling* 28.10 (Oct. 2004), pp. 911–931.

[171] Fabiano Luis de Sousa, Valeri Vlassov, and Fernando Manuel Ramos. “Heat Pipe Design Through Generalized Extremal Optimization”. In: *Heat Transfer Engineering* 25.7 (Oct. 2004), pp. 34–45.

[172] Min-Joong Jeong, Takashi Kobayashi, and Shinobu Yoshimura. “Multidimensional visualization and clustering for multiobjective optimization of artificial satellite heat pipe design”. In: *Journal of Mechanical Science and Technology* 21.12 (Dec. 2007), pp. 1964–1972.

[173] Chengbin Zhang et al. “Optimization of heat pipe with axial Ω-shaped micro grooves based on a niched Pareto genetic algorithm (NPGA)”. In: *Applied Thermal Engineering* 29.16 (2009), pp. 3340–3345.

[174] Valery M. Kiseev, Valeri V. Vlassov, and Issamu Muraoka. “Experimental optimization of capillary structures for loop heat pipes and heat switches”. In: *Applied Thermal Engineering* 30.11-12 (Aug. 2010), pp. 1312–1319.

[175] Valery M. Kiseev, Valeri V. Vlassov, and Issamu Muraoka. "Optimization of capillary structures for inverted meniscus evaporators of loop heat pipes and heat switches". In: *International Journal of Heat and Mass Transfer* 53.9-10 (Apr. 2010), pp. 2143–2148.

[176] P. Maheshkumar and C. Muraleedharan. "Minimization of entropy generation in flat heat pipe". In: *International Journal of Heat and Mass Transfer* 54.1-3 (Jan. 2011), pp. 645–648.

[177] Christopher S. Roper. "Multiobjective optimization for design of multifunctional sandwich panel heat pipes with micro-architected truss cores". In: *International Journal of Heat and Fluid Flow* 32.1 (Feb. 2011), pp. 239–248.

[178] Zhen-ping Wan, Xiao-wu Wang, and Yong Tang. "Condenser design optimization and operation characteristics of a novel miniature loop heat pipe". In: *Energy Conversion and Management* 64 (Dec. 2012), pp. 35–42.

[179] R. V. Rao and K. C. More. "Optimal design of the heat pipe using TLBO (teaching–learning-based optimization) algorithm". In: *Energy* 80 (Feb. 2015), pp. 535–544.

[180] Ali Jokar et al. "Simulation and optimization of a pulsating heat pipe using artificial neural network and genetic algorithm". In: *Heat and Mass Transfer* 52.11 (Jan. 2016), pp. 2437–2445.

[181] Hong-jie Song et al. "Exergy analysis and parameter optimization of heat pipe receiver with integrated latent heat thermal energy storage for space station in charging process". In: *Applied Thermal Engineering* 119 (June 2017), pp. 304–311.

[182] Vivek K. Patel. "An efficient optimization and comparative analysis of ammonia and methanol heat pipe for satellite application". In: *Energy Conversion and Management* 165 (June 2018), pp. 382–395.

[183] S. A. Lurie, L. N. Rabinskiy, and Y. O. Solyaev. "Topology optimization of the wick geometry in a flat plate heat pipe". In: *International Journal of Heat and Mass Transfer* 128 (Jan. 2019), pp. 239–247.

[184] V. G. Rajesh and K. P. Ravindran. “Optimum heat pipe design: A nonlinear programming approach”. In: *International Communications in Heat and Mass Transfer* 24.3 (May 1997), pp. 371–380.

[185] Kwonho Kim, Kyun Ho Lee, and Seung Wook Baek. “A Study on Heat Pipe Optimization Using PSO”. In: *International Journal of Computer and Electrical Engineering* (2013), pp. 291–293.

[186] Oğuz Emrah Turgut and Mustafa Turhan Çoban. “Thermal design of spiral heat exchangers and heat pipes through global best algorithm”. In: *Heat and Mass Transfer* 53.3 (July 2016), pp. 899–916.

[187] Jay M. Ochterbeck. “Heat Pipes”. In: *Heat Transfer Handbook*. Ed. by Adrian Bejan and Allan D. Kraus. 2003.

[188] S. S. Shapiro and M. B. Wilk. “An Analysis of Variance Test for Normality (Complete Samples)”. In: *Biometrika* 52.3/4 (Dec. 1965), p. 591.

[189] Qiang Bai, Samuel Labi, and Kumares C. Sinha. “Trade-Off Analysis for Multiobjective Optimization in Transportation Asset Management by Generating Pareto Frontiers Using Extreme Points Nondominated Sorting Genetic Algorithm II”. In: *Journal of Transportation Engineering* 138.6 (June 2012), pp. 798–808.

[190] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. “Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization”. In: *Nanyang Technological University, Singapore* (2010).

[191] Eckart Zitzler and Lothar Thiele. “Multiobjective optimization using evolutionary algorithms — A comparative case study”. In: *Lecture Notes in Computer Science*. Ed. by Agoston E. Eiben et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 292–301.

[192] C.A.C. Coello. “An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends”. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. IEEE, 1999.

[193] H. Ishibuchi, Y. Nojima, and Tsutomu Doi. “Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures”. In: *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006, pp. 1143–1150.

[194] Ryan McGlen, Peter Kew, and David Reay. *Heat Pipes*. Elsevier Science, Oct. 12, 2006. 384 pp.

APPENDIX A

MATERIAL PROPERTIES USED FOR HEAT PIPE DESIGN PROBLEM

A.1 Container and wick material properties

Stainless steel - SS304 is taken as container and wick material.

$$\rho = 8000\,\mathrm{kg/m^3} \qquad k = 17.3\,\mathrm{W/mK} \qquad u_{ts} = 515 \times 10^6\,\mathrm{Pa}$$

A.2 Working fluid properties

Nomenclature- and units used for fluid properties are summarized below. Data is adapted from McGlen, Kew, and Reay [194].

Symbol	Name	Unit
T_v	vapor temperature	K
λ	latent heat of vaporization	$\mathrm{J/kg}$
ρ_l	liquid density	$\mathrm{kg/m^3}$
ρ_v	vapor density	$\mathrm{kg/m^3}$
k_l	liquid thermal conductivity	$\mathrm{W/mK}$
μ_l	liquid viscosity	$\mathrm{kg/ms}$
μ_v	vapor viscosity	$\mathrm{kg/ms}$
P_v	vapor pressure	$\mathrm{N^2/m}$
M	molar mass	$\mathrm{kg/mol}$
γ	Specific heat ratio	-

A.2.1 Methanol

$\gamma = 1.203$, $M = 0.032\,042$

T_v	λ	ρ_l	ρ_v	k_l	μ_l	μ_v	P_v	σ
223.15	1 194 000	843.5	0.01	0.21	0.0017	0.072	1000	0.0326
243.15	1 187 000	833.5	0.01	0.208	0.0013	0.078	2000	0.0295
263.15	1 182 000	818.7	0.04	0.206	0.000 945	0.085	4000	0.0263
283.15	1 175 000	800.5	0.12	0.204	0.000 701	0.091	10 000	0.0236
303.15	1 155 000	782	0.31	0.203	0.000 521	0.098	25 000	0.0218
323.15	1 125 000	764.1	0.77	0.202	0.000 399	0.104	55 000	0.0201
343.15	1 085 000	746.2	1.47	0.201	0.000 314	0.111	131 000	0.0185
363.15	1 035 000	724.4	3.01	0.199	0.000 259	0.119	269 000	0.0166
383.15	980 000	703.6	5.64	0.197	0.000 211	0.126	498 000	0.0146
403.15	920 000	685.2	9.81	0.195	0.000 166	0.131	786 000	0.0125
423.15	850 000	653.2	15.9	0.193	0.000 138	0.138	894 000	0.0104

A.2.2 Ethanol

$\gamma = 1.13$, $M = 0.046\,07$

T_v	λ	ρ_l	ρ_v	k_l	μ_l	μ_v	P_v	σ
243.15	939 400	825	0.02	0.177	0.0034	0.075	1000	0.0276
263.15	928 700	813	0.03	0.173	0.0022	0.08	2000	0.0266
283.15	904 800	798	0.05	0.17	0.0015	0.085	3000	0.0257
303.15	888 600	781	0.38	0.168	0.001 02	0.091	10 000	0.0244
323.15	872 300	762.2	0.72	0.166	0.000 72	0.097	29 000	0.0231
343.15	858 300	743.1	1.32	0.165	0.000 51	0.102	76 000	0.0217
363.15	832 100	725.3	2.59	0.163	0.000 37	0.107	143 000	0.0204
383.15	786 600	704.1	5.17	0.16	0.000 28	0.113	266 000	0.0189
403.15	734 400	678.7	9.25	0.159	0.000 21	0.118	430 000	0.0175

A.2.3 Ammonia

$\gamma = 1.32$, $M = 0.017\,031$

T_v	λ	ρ_l	ρ_v	k_l	μ_l	μ_v	P_v	σ
213.15	1 343 000	714.4	0.03	0.294	0.000 36	0.072	27 000	0.040 62
233.15	1 384 000	690.4	0.05	0.303	0.000 29	0.079	76 000	0.035 74
253.15	1 338 000	665.5	1.62	0.304	0.000 26	0.085	193 000	0.0309
273.15	1 263 000	638.6	3.48	0.298	0.000 25	0.092	424 000	0.0248
293.15	1 187 000	610.3	6.69	0.286	0.000 22	0.101	846 000	0.021 33
313.15	1 101 000	579.5	12	0.272	0.0002	0.116	1 534 000	0.018 33
333.15	1 026 000	545.2	20.49	0.255	0.000 17	0.127	2 980 000	0.013 67
353.15	891 000	505.7	34.13	0.235	0.000 15	0.14	4 090 000	0.007 67
373.15	699 000	455.1	54.92	0.212	0.000 11	0.16	6 312 000	0.005
393.15	428 000	374.4	113.16	0.184	0.000 07	0.189	9 044 000	0.0015

APPENDIX B

USAGE NOTES FOR MATLAB IMPLEMENTATION

The MATLAB implementation of MOFEPSO consists of two functions named `mofepso` and `mofepsooptions`[1]. While the former is the function that allows running the main algorithm, the second is a helper function to construct the necessary input to the main function.

B.1 The `mofepso` function

Runs the main algorithm of MOFEPSO.

B.1.1 Syntax

```
result = mofepso(options)
```

This function runs the MOFEPSO algorithm with the options given. The `options` structure defines all inputs required by the algorithm including constraint(s) function, objective(s) function, and algorithm parameters.

B.1.2 Input argument

The only input required by the `mofepso` function is a structure that includes all information required by the algorithm as well as some additional convenience features. The `options` structure has the fields described in Table B.1. The `mofepsooptions`

---

[1] The MATLAB implementation of MOFEPSO can be found at MATLAB central: https://www.mathworks.com/matlabcentral/fileexchange/68990-mofepso-multi-objective-feasibility-enhanced-particle-swarm.

function described in Section B.2 can be utilized to construct a valid options structure. The `mofepsooptions` function can create default values for all optional parameters. If the options structure is to be created manually (i.e. without using `mofepsooptions`), it must include fields for all options including the optional ones. Note that, mandatory options have no default values in Table B.1.

Table B.1: Fields of the options structure

Field	**Description**
`consFun`	Constraint(s) function. This function shall take the 1-by-N decoded decision variable array (if input decoding is used) and return a M-by-1 array of constraint values[2]. A constraint is assumed to be satisfied only when it is equal to or less than zero ($g \leq 0$). See Section B.1.2.1 for additional details. **Values:** Function handle **Default value:** None
`objFun`	Objective(s) function. This function shall take the 1-by-N decoded decision variable array (if input decoding is used) and return a K-by-1 array of objective values[3]. MOFEPSO minimizes the objectives. See Section B.1.2.1 for additional details. **Values:** Function handle **Default value:** None
`nCons`	Number of constraints. If the problem does not have constraints (`nCons=0`), the `consFun` parameter can also be set to zero (`consFun=0`). **Values:** 0 or positive integer **Default value:** None

---

[2] M is the number of constraints

[3] K is the number of objectives.

Table B.1: Fields of the options structure (continued)

Field	**Description**
`lBound`	Lower bounds of decision variables. **Values:** 1-by-N row vector of real numbers[4] **Default value:** None
`uBound`	Upper bounds of decision variables. **Values:** 1-by-N row vector of real numbers. **Default value:** None
`swarmSize`	Number of particles in swarm. **Values:** Positive integer **Default value:** 200
`maxIter`	Maximum number of iterations before algorithm stops. **Values:** Positive integer **Default value:** 200
`useInputDecoder`	Boolean flag to enable the usage of the *input decoder function*. **Values:** `true` `false` **Default value:** `false`

---

[4] N denotes the number of decision variables.

Table B.1: Fields of the options structure (continued)

Field	Description
`inputDecoder`	A function that decodes the real valued raw decision variables into decision variable values that meet the problem definition. The input decoder function shall accept a 1-by-N row vector of raw variable values and return a 1-by-N row vector of values abiding decision variable definitions of the problem. For example, if a decision variable can only take integer values, the input decoder function shall convert the real raw value in the corresponding cell of the decision variable array to an integer value using an appropriate mapping. This function is only used when `useInputDecoder == true`. **Values:** Function handle **Default value:** `@(x) x`
`transferVars`	Boolean flag to enable transfer of data from constraint function to objective function. In some problems, certain parameters calculated by the constraint function might be useful for the objective function. When transfer of data is enabled, a mechanism is provided for this purpose. See Section B.1.2.1 for details. **Values:** `true` `false` **Default value:** `false`

Table B.1: Fields of the options structure (continued)

Field	**Description**
`incrementFactor`	Increment factor determines the ratio of search space dimension the constraint sensitivity calculation algorithm will displace particles[5]. **Values:** Real number between 0 and 1 **Default value:** 0.3
`velocityInitializationFactor`	Velocity initialization factor determines the maximum magnitude of the velocity vectors of the initial swarm relative to search space size[6]. **Values:** Real number between 0 and 1 **Default value:** 0.3
`boundaryTolerance`	Virtual boundary search tolerance[7]. **Values:** Positive real number **Default value:** 0.01
`initialC0`	Value of the inertia factor at the beginning of iterations. Inertia factor is varied linearly through iterations. **Values:** Positive real number **Default value:** 0.9
`finalC0`	Value of the inertia factor at the end of iterations. **Values:** Positive real number **Default value:** 0.4
`C1`	Social acceleration factor. **Values:** Positive real number **Default value:** 2

---

[5] Constraint sensitivity calculation approach is explained in Section 3.2.

[6] Velocity initialization factor is denoted as a^v in Section 3.2.

[7] Virtual boundary search tolerance is denoted as x^{tol} in Section 3.5.2.

Table B.1: Fields of the options structure (continued)

Field	**Description**
`C2`	Cognitive acceleration factor. **Values:** Positive real number **Default value:** 2
`verbose`	Level of information printed to the standard output. When `verbose=0`, `mofepso` doesn't produce anything. Only iteration number and durations are printed if `verbose=1`. All information described in Section B.1.4 is printed if `verbose=2`. Although `verbose=3` is same as `verbose=2` in terms of what is printed to the standard output, position and velocity histories of particles are added to the returned result. **Values:** `012` **Default value:** 2
`fastVBS`	Boolean flag to enable or disable fast virtual boundary search. When enabled slowly converging VBS loops are halted. May decrease the number of constraint function evaluations but also prohibits convergence to the virtual boundary for some particles. **Values:** `true` `false` **Default value:** `true`

B.1.2.1 Constraint- and objective function definitions

Constraint- and objective functions are defined such that they take a 1-by-N array of decision variable values and return column vectors representing constraint function values or objective values. Following example shows function definitions for the benchmark problem called the pressure vessel design problem (see Section 3.8.1). The constraint function of the problem can be defined as follows.

```
function g = pv_constraint(x)
 g = [-x(1) + 0.0193*x(3);
      -x(2) + 0.00954*x(3);
      -pi*(x(3)^2)*x(4) - (4/3)*pi*x(3)^3 + 1296000;
      x(4) - 240];
end
```

Similarly, the objective function can be defined the following way.

```
function y = pv_objective(x)
 y = 0.6224*x(1)*x(3)*x(4) + 1.7781*x(2)*x(3)^2 ...
     + 3.1661*x(1)^2*x(4) + 19.84*x(1)^2*x(3);
end
```

Note that the pressure vessel problem is a single-objective problem. Therefore, the `pvobjective` function returns a scalar value instead of an array.

In some cases the problem might involve decision variables that are not real valued. MOFEPSO is able to handle these kind of problems as long as a mapping from a real valued range exists. For instance, the pressure vessel design problem involves two decision variables that can only take exact multiples of 0.0625. This can be achieved with an input decoder function that resembles the following:

```
function x_decoded = pv_inputDecoder(x)
  x_decoded = [0.0625*round(x(1)) 0.0625*round(x(2)) ...
               x(3) x(4)];
end
```

Note that, for input decoding to work properly, the `useInputDecoder` option must be set to `true` and a function handle must be assigned to the `inputDecoder` option.

MOFEPSO evaluates the objective function only in certain cases. Therefore, it does not make sense to combine calculation of constraints and objectives in a single process. However, some parameters computed within the context of the constraint function might be used in objective evaluation. If the `transferVars` option is enabled, MOFEPSO provides a mechanism to pass these parameters to the objective function. In this case, the constraint functions is defined as follows.

```
function [g, transfer] = constraint_function(x)
 % ...
 % calculate a and b which are required for both
 % constraint and objective functions
 % ...
 trasfer = {a b};
end
```

Similarly, the objective function is defined to take advantage of the transfered variables.

```
function y = objective_function(x, transfer)
 a = transfer{1};
 b = transfer{2};
 % ...
end
```

B.1.3 Output argument

The `mofepso` function returns a structure containing all non-dominated solutions found by the algorithm and other details related with the optimization run. Fields of the output structure is described in Table B.2.

Table B.2: Fields of the output structure

Field	**Description**
`particle`	A 1-by-I structure array describing current properties of particles in the swarm[8]. See Section B.1.3.1 for details of the particle structure.
`nonDom`	A cell array containing all non-dominated solutions as individual position structures.
`consSens`	The M-by-N logical matrix of constraint sensitivities. If `consSens(m,n)==true`, the mth constraint is sensitive to changes in the nth decision variable.
`violated`	The I-by-M logical matrix of constraint violations of current positions of particles. If `violated(i,m)==true`, the ith particle's current position violates mth constraint.
`options`	The options structure used.
`nObjEvals`	The integer value of total number of objective function evaluations performed by the algorithm.
`nConsEvals`	The integer value of total number of constraint function evaluations performed by the algorithm.

B.1.3.1 The particle structure

The particle structure defines the representation of information related with each particle. The fields of this structure is explained in Table B.3.

---

[8] I denotes the number of particles in the swarm (i.e. `swarmSize`).

Table B.3: Fields of the particle structure

Field	**Description**
`nonDom`	A cell array containing all personal non-dominated solutions as individual position structures. These positions are non-dominated among all positions attained by the particle of interest.
`pos`	The position structure that represents current position of the particle.
`v`	The 1-by-N array of real values that represent the current velocity of the particle.
`poshist`	The $(I + 1)$ sized array of position structure that represent the particle's position history. Only available when `verbose=3`
`vhist`	The $(I + 1)$-by-N array of velocity history matrix. Only available when `verbose=3`

B.1.3.2 The position structure

All positions output by `mofepso` are in the form described in this section. The position structure contains information related with the constraint function values and objective values alongside the spatial information related with the position (See Table B.4).

Table B.4: Fields of the position structure

Field	**Description**
`x`	The 1-by-N sized real valued array of un-decoded decision variable values (i.e. position in the decision space).
`cons`	The M-by-1 array of constraint function values at the position.

Table B.4: Fields of the position structure (continued)

Field	**Description**
`isFeasible`	The logical value (`true` `false`) representing the feasibility of the position
`obj`	The K-by-1 array of objective function values at the position.
`objIsEvaluated`	The logical value that becomes true if the objective function has been evaluated for the position.
`nConsEvals`	The integer value representing the total number of constraint function evaluations performed by the algorithm right after the constraint function was evaluated for the position.
`nObjEvals`	The integer value representing the total number of objective function evaluations performed by the algorithm right after the objective function was evaluated for the position.

B.1.4 Standard output

The `mofepso` function outputs some information related with its current status as the optimization is running. An example of the information printed to the standard output is shown below.

```
Initializing particles
Calculating constraint sensitivities
Iteration 1 started
  Iteration 1 finished in  0.0 seconds
  Number of times each constraint is violated
     1   2   3   4
     1   0   1   0
  Number of particles in number of violation bins
     0   1
    48   2
  Number of CFE: 423
  Number of OFE: 87
  Nondominated set (1 points):
   51559.3940 | 3.454039 0.798188 83.667524 100.257993 | 423 87
Iteration 2 started
  Iteration 2 finished in  0.0 seconds
  Number of times each constraint is violated
     1   2   3   4
     0   0   0   0
  Number of particles in number of violation bins


  Number of CFE: 517
  Number of OFE: 137
  Nondominated set (1 points):
   51559.3940 | 3.454039 0.798188 83.667524 100.257993 | 423 87
```

As seen above, `mofepso` prints information related with the progression of the run, duration of iterations, violation numbers of each constraint, constraint satisfaction level of the swarm, number of constraint/objective function evaluations, and solutions in the current non-dominated set. The array of number of particles in number of violation bins is the main metric for the swarm's feasibility. The meaning of having 48 particles in the 0 violation bin is that there are 48 particles that do not violate any constraints.

Note that although `mofepso` does not require to have any feasible particles in the initial swarm, it requires at least one satisfying particle for each constraint. Therefore, if a constraint is not satisfied by any of the particles, `mofepso` re-initializes the particles.

B.2 The `mofepsooptions` function

Creates an options structure to be used as the input argument of the `mofepso` function.

B.2.1 Syntax

```
options = mofepsooptions(consFun, objFun, nCons, ...
     lBound, ubound)
options = mofepsooptions(consFun, objFun, nCons, ...
     lBound, ubound, field1, value1,...,fieldN, valueN)
```

This function creates a valid options structure when all mandatory options listed in Table B.1 are provided as input arguments. Note that mandatory input argument names are same as the names of options they set. Therefore, Table B.1 can be referred for the properties of these arguments. Values for non-mandatory options can be given to `mofepsooptions` function as name-value pairs. If no value is provided for an option, `mofepsooptions` sets the value of the option to the default value given in Table B.1.

B.2.2 Output argument

The `mofepsooptions` function returns a structure that can directly be used as the input argument for the `mofepso` function.

B.3 Example

In this section, solution of the multi-objective four-stage gear train problem [98] with MOFEPSO in MATLAB is presented as an example. MATLAB source code files needed for the example problem have the structure shown in Figure B.1 relative to the main algorithm files. All code given assumes that the active directory is "/" in this figure.

The constraint function of the problem is given in MATLAB code 1. Note that this function is arranged to transfer some variables to the objective function.

Matlab code 1: `mog_constraint.m`

```
function [ constraintVec, transfer ] = mog_constraint( x )
%mog_constraint calculates the constraints of the four-stage gear train
```

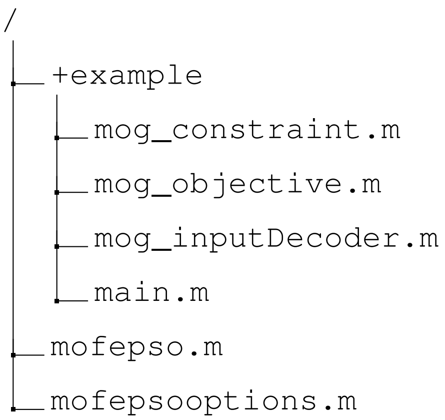


Figure B.1: Directory structure of the MATLAB source code files for the example problem

```
%problem

%Constants required are defined here:
%Gearbox scope:
omega_1 = 5000;     %Input speed [rpm]
omega_min = 245;    %Minimum output speed [rpm]
omega_max = 255;    %Maximum output speed [rpm]

%Gear stage scope:
C_p = 464.0;        %Elastic coefficient
CR_min = 1.4;       %Allowable contact ratio
d_min = 25.4;       %Minimum pinion or gear diameter [mm]
phi = 20;           %Pressure angle [deg]
W = 55.9;           %Input power[W]
J_R = 0.2;          %Geometry factor
K_M = 1.6;          %Mounting factor
K_O = 1.5;          %Overload factor
L_max = 127;        %Maximum housing dimension[mm]
sigma_H = 3290;     %Allowable fatigue stress [kgf/cm2]
sigma_N = 2090;     %Allowable bending stress [kgf/cm2]

%***************************************************************************
%Precalculated limits
gearToothBendingFatigueStrength = sigma_N*J_R / (0.0167*W*K_O*K_M);
gearToothContactStrength = ((sigma_H/C_p)^2) * ...
    ( (sin(2*degtorad(phi))) / (0.0668*W*K_O*K_M) );
```

```
%Minimum value for gearToothContactRatio
gearToothContactRatioMin = CR_min*pi*cos(degtorad(phi));
%***************************************************************************
%Inputs
firstPinionPos = [x(1) x(2)];

allGearPos = [ x(3) x(4);
               x(5) x(6);
               x(7) x(8);
               x(9) x(10)];

all_b = [x(11); x(12); x(13); x(14)];

all_N_g = [x(15); x(16); x(17); x(18)];
all_N_p = [x(19); x(20); x(21); x(22)];
%***************************************************************************
%Allowed values of b
pB = [3.175 5.715 8.255 12.7];

%Calculations
stageConstraints = zeros(15,4);
omega = omega_1;
pinionPos = firstPinionPos;

%Initialize volume for the objective function
V = 0;

for i=1:4
    gearPos = allGearPos(i,:);
    b = all_b(i);
    %Double versions of the int parameters for double precission calc.
    dN_p = double(all_N_p(i));
    dN_g = double(all_N_g(i));

    c = sqrt((gearPos(1)-pinionPos(1))^2+(gearPos(2)-pinionPos(2))^2);
    if ~(c > 0)
        c = 0.0001;
    end

    %stress and contact ratio calculations
```

```
gearToothBendingFatigue = ( ( (366000)/(pi*omega) ) + ...
    ( ( 2*c*dN_p ) / (dN_p+dN_g) ) ) * ...
    ( (dN_p+dN_g)^2 / (4*b*c^2*dN_p) );
gearToothContactStress = ( ( (366000)/(pi*omega) ) + ...
    ( ( 2*c*dN_p ) / (dN_p+dN_g) ) ) * ...
    ( (dN_p+dN_g)^3 / (4*b*c^2*dN_p^2*dN_g) );

A = dN_p*sqrt((sin(degtorad(phi))^2)/4 + 1/dN_p + 1/(dN_p^2));
B = dN_g*sqrt((sin(degtorad(phi))^2)/4 + 1/dN_g + 1/(dN_g^2));
C = (dN_p + dN_g)*sin(degtorad(phi))/2;
gearToothContactRatio = A + B - C;

%Geometric calculations
r_p = c * (dN_p/(dN_p+dN_g));    r_g = c * (dN_g/(dN_p+dN_g));
d_p = 2 * r_p;                   d_g = 2 * r_g;

addendum =  2*c/(dN_p+dN_g);
or_p = r_p + addendum;           or_g = r_g + addendum;

P = (dN_p+dN_g)/(2*c);

%calculation of the bounds
pinionBoundMin = pinionPos - [or_p or_p];
pinionBoundMax = pinionPos + [or_p or_p];
gearBoundMin = gearPos - [or_g or_g];
gearBoundMax = gearPos + [or_g or_g];

%speed
omega_out = omega*dN_p / dN_g;

%constraints
stageConstraints(1,i) = gearToothBendingFatigue - ...
    gearToothBendingFatigueStrength;
stageConstraints(2,i) = gearToothContactStress - ...
    gearToothContactStrength;
stageConstraints(3,i) = gearToothContactRatioMin - ...
    gearToothContactRatio;
stageConstraints(4,i) = d_min - d_g;
stageConstraints(5,i) = d_min - d_p;
stageConstraints(6,i) = -pinionBoundMin(1,1);
stageConstraints(7,i) = -pinionBoundMin(1,2);
stageConstraints(8,i) = pinionBoundMax(1,1) - L_max;
stageConstraints(9,i) = pinionBoundMax(1,2) - L_max;
stageConstraints(10,i) = -gearBoundMin(1,1);
stageConstraints(11,i) = -gearBoundMin(1,2);
stageConstraints(12,i) = gearBoundMax(1,1) - L_max;
stageConstraints(13,i) = gearBoundMax(1,2) - L_max;

%Min & Max Pitch calculations
P_min_array = [0.472 0.323 0.252 0];
P_max_array = [0.906 0.472 0.323 0.252];

stageConstraints(14,i) = (P_min_array(1) - P) * (b - pB(2)) * ...
    (b - pB(3)) * (b - pB(4)) * (-1);
stageConstraints(15,i) = (P_min_array(2) - P) * (b - pB(1)) * ...
    (b - pB(3)) * (b - pB(4)) * (1);
stageConstraints(16,i) = (P_min_array(3) - P) * (b - pB(1)) * ...
    (b - pB(2)) * (b - pB(4)) * (-1);
stageConstraints(17,i) = (P_min_array(4) - P) * (b - pB(1)) * ...
    (b - pB(2)) * (b - pB(3)) * (1);

stageConstraints(18,i) = (P - P_max_array(1)) * (b - pB(2)) * ...
    (b - pB(3)) * (b - pB(4)) * (-1);
stageConstraints(19,i) = (P - P_max_array(2)) * (b - pB(1)) * ...
    (b - pB(3)) * (b - pB(4)) * (1);
stageConstraints(20,i) = (P - P_max_array(3)) * (b - pB(1)) * ...
    (b - pB(2)) * (b - pB(4)) * (-1);
stageConstraints(21,i) = (P - P_max_array(4)) * (b - pB(1)) * ...
    (b - pB(2)) * (b - pB(3)) * (1);

% Calculate variables for the objective function
% Volume
volume = pi*b*c^2*(dN_p^2+dN_g^2)/(dN_p+dN_g)^2;
V = V + volume;
%Bound Calculations
boundMin = min( [pinionBoundMin; gearBoundMin] );
boundMax = max( [pinionBoundMax; gearBoundMax] );
if i==1
    boxBoundMin = boundMin;
    boxBoundMax = boundMax;
    else
        boxBoundMin = min([ boxBoundMin ; boundMin ]);
        boxBoundMax = max([ boxBoundMax ; boundMax ]);
    end

    %prepare for the next stage
    pinionPos = gearPos;
    omega = omega_out;
end

%constraint vector
constraintVec     = [stageConstraints(:,1);
                    stageConstraints(:,2);
                    stageConstraints(:,3);
                    stageConstraints(:,4);
                    omega_min - omega_out;
                    omega_out - omega_max];

%variables to be transferred to objective function
transfer = {V, boxBoundMin, boxBoundMax};
end
```

The objective function receives several parameters transferred from the constraint function and calculates the objectives. Objectives are then returned as a column vector. The source code for the objective function is given in MATLAB code 2

Matlab code 2: `mog_objective.m`

```
function objectiveVec = mog_objective( x, transfer )
%mog_constraint calculates the objectives of the four-stage gear train
%problem
%***************************************************************************
%Inputs
all_b = [x(11); x(12); x(13); x(14)];
%***************************************************************************
%Get transferred variables
V = transfer{1};
boxBoundMin = transfer{2};
boxBoundMax = transfer{3};

%output vector
```

```
boundDif = boxBoundMax - boxBoundMin;
V_box = sum(all_b) * boundDif(1,1) * boundDif (1,2);

objectiveVec     = [V; V_box];
end
```

All decision variables of the four-stage gear train problem have discrete allowed values. The algorithm's un-decoded real valued positions need to be converted to decision vectors valid in the problem's discrete decision space. For this purpose, an input decoder (see MATLAB code 3) is required. Note that, for this input decoder to work, the `useInputDecoder` option must also be enabled.

Matlab code 3: `mog_inputDecoder.m`

```
function decoded_x = mog_inputDecoder(x)
%mog_inputDecoder decodes the real valued decision vector and decodes to
%discrete values

%Possible values
pPositions = 12.7 * (1:9);
pB = [3.175 5.715 8.255 12.7];

%round the variables
for i=1:10
    x(i) = rounder(x(i),pPositions);
end

for i=11:14
    x(i) = rounder(x(i),pB);
end

for i=15:22
    x(i) = round(x(i));
end

decoded_x = x;
end

function val =  rounder(inVal,pValues)
    [~,ind] = min(abs(inVal - pValues));
    val = pValues(ind);
```

```
end
```

Finally, `main.m` sets the necessary options of the algorithm and runs the optimization. Results are returned by the function `mofepso` as a structure of the form specified in B.1.3.

Matlab code 4: `main.m`

```
lBound = [12.7 12.7 12.7 12.7 12.7 12.7 12.7 12.7 12.7 12.7 ...
    3.175 3.175 3.175 3.175 7 7 7 7 7 7 7 7];
uBound = [114.3 114.3 114.3 114.3 114.3 114.3 114.3 114.3 114.3 114.3 ...
    12.7 12.7 12.7 12.7 60 60 60 60 60 60 60 60];

options = mofepsooptions(@example.mog_constraint, ...
    @example.mog_objective, 86, lBound, uBound, ...
    'useInputDecoder', true, 'inputDecoder', @example.mog_inputDecoder, ...
    'transferVariables', true, 'maxIter', 500, 'swarmSize', 200);

result = mofepso(options);
```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Hasanoğlu, Mehmet Sinan
Nationality: Turkish
Date and Place of Birth: 1984, Ankara
Marital Status: Married
Phone: +90 312 590 9304
E-mail: sinan.hasanoglu@tubitak.gov.tr

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.S.	METU Mechanical Engineering	2008
B.S.	METU Mechanical Engineering	2005
High School	Ankara Fen Lisesi	2001

PROFESSIONAL EXPERIENCE

Year	**Place**	**Enrollment**
Jan 2018 - Present	TÜBİTAK SAGE, Air Defense Missiles Program Management Division	Project Manager
Jun 2016 - Jun 2018	TÜBİTAK SAGE, Air Defense Missiles Program Management Division	Division Head

Dec 2014 - May 2016	TÜBİTAK SAGE, Systems Engineering Group	Group Coordinator
Aug 2014 - Dec 2014	TÜBİTAK SAGE, Systems Engineering Division	Division Head, Chief Research Engineer
Apr 2011 - Aug 2015	TÜBİTAK SAGE, Systems Engineering Division	Senior Research Engineer
Mar 2009 - Aug 2010	TÜBİTAK SAGE, Systems and Specialty Engineering Division	Senior Research Engineer
Dec 2004 - Apr 2009	TÜBİTAK SAGE, Systems and Specialty Engineering Division	Research Engineer

PUBLICATIONS

International Journal Publications

1. Mehmet Sinan Hasanoglu and Melik Dolen. “Multi-objective feasibility enhanced particle swarm optimization”. In: *Engineering Optimization* 50.12 (Feb. 2018), pp. 2013–2037

2. Mehmet Sinan Hasanoglu and Melik Dolen. “Feasibility enhanced particle swarm optimization for constrained mechanical design problems”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 232.2 (2018), pp. 381–400

International Conference Publications

1. Umit Kutluay, Ibrahim Karbancioglu, and Mehmet Hasanoglu. “External Geometry Optimization with Flight Mechanics Constraints Using Response Surfaces”. In: *AIAA Atmospheric Flight Mechanics Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2008

2. Bayindir Kuran, Mehmet Hasanoglu, and Kenan Bozkaya. “Robust Design Optimization for Multiple Responses Using Response Surface Methodology

and Taguchi Approach: Solid Rocket Motor Application”. In: *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, Apr. 2007

3. Kenan Bozkaya, Bayindir Kuran, Mehmet Hasanoglu, Mehmet Yildirim, and Mehmet Ak. “Effects of Production Variations on the Reliability of a Solid Rocket Motor”. In: *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*. American Institute of Aeronautics and Astronautics, July 2006