

TESE DE DOUTORADO Nº 174

**THE NN-DM METHOD - AN ARTIFICIAL NEURAL NETWORK MODEL FOR  
DECISION-MAKER'S PREFERENCE**

**Luciana Rocha Pedro**

DATA DA DEFESA: 20/12/2013

**Universidade Federal de Minas Gerais**

**Escola de Engenharia**

**Programa de Pós-Graduação em Engenharia Elétrica**

THE NN-DM METHOD - AN ARTIFICIAL NEURAL NETWORK  
MODEL FOR DECISION-MAKER'S PREFERENCES

Luciana Rocha Pedro

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Ricardo Hiroshi Caldeira Takahashi

Belo Horizonte - MG

Dezembro de 2013

P372n Pedro, Luciana Rocha.  
The NN-DM method [manuscrito]: an artificial neural network model for decision-maker's preferences / Luciana Rocha Pedro. - 2013.  
xviii, 164 f., enc.: il.

Orientador: Ricardo Hiroshi Caldeira Takahashi.

Tese (doutorado) Universidade Federal de Minas Gerais,  
Escola de Engenharia.

Anexos: f. 145-162.

Bibliografia: f. 136-144.

1. Engenharia elétrica - Teses. 2. Algoritmos genéticos - Teses.  
3. Processo decisório por critério múltiplo - Teses. I. Takahashi, Ricardo Hiroshi Caldeira. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)


**"The NN-DM Method – An Artificial Neural Network Model for  
Decision-Maker's Preferences"**

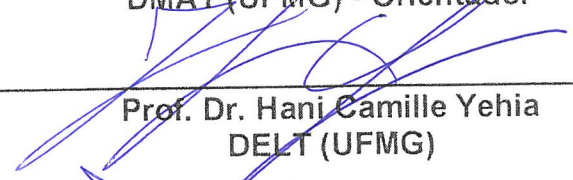
**Luciana Rocha Pedro**


Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

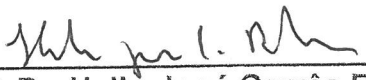
Aprovada em 20 de dezembro de 2013.

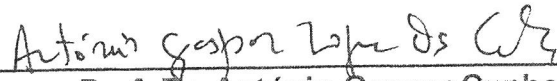
Por:

  
\_\_\_\_\_  
Prof. Dr. Ricardo Hiroshi Caldeira Takahashi  
DMAT (UFMG) - Orientador

  
\_\_\_\_\_  
Prof. Dr. Hani Camille Yehia  
DELT (UFMG)

  
\_\_\_\_\_  
Prof. Dr. Frederico Gadelha Guimarães  
DEE (UFMG)

  
\_\_\_\_\_  
Prof. Dr. Helio José Corrêa Barbosa  
(Laboratório Nacional de Computação Científica (LNCC))

  
\_\_\_\_\_  
Prof. Dr. António Gaspar Cunha  
Dep. de Engenharia de Polímeros (Univ. do Minho - Portugal)

“Your act was unwise,” I exclaimed “as you see by the outcome.”  
He solemnly eyed me. “When choosing the course of my action,”  
said he, “I had not the outcome to guide me.”

*Ambrose Bierce*

# Acknowledgements

To my faithful companion, Paloquinho, for his support at all times.

To my advisor, Ricardo Takahashi, for freedom of thought and his unconditional support.

To my family that, even distant, always cheer for me.

To the doctors Alexandre Celestino and Rodrigo Cardoso and to my friends Cristine Almeida, Camila Albino, Fernanda Alvarenga, and Adriano Silva for the friendship over the past decades.

To Chrystian, Fernando, and Leonardo for the support in those final moments.

To all my friends for the moments of relaxation and fun.

To Capes for the encouraging scientific and financial support.

# Abstract

This work presents a methodology based on the multi-attribute utility theory to approximate the decision-maker's utility function: the Neural Network Decision-Maker method (NN-DM method). The preference information extracted from the Decision-Maker (DM) involves ordinal description only and is structured by a partial ranking procedure. An artificial neural network is then constructed to approximate the partial ranking reproducing the DM's preferences in a specific domain. The NN-DM method is suitable in situations in which a recurrent decision process must be performed considering different sets of alternatives and the same DM.

A hybridization between the NN-DM method and the Interactive Territory Defining Evolutionary Algorithm (iTDEA) is also developed in this work. Considering the same amount of preference information, the NN-DM method is able to construct a model for the DM's preferences to guide iTDEA. Henceforth, no further queries are required from the DM related to similar decision-making problems.

Additionally, an Interactive Non-dominated Sorting algorithm with Preference Model (INSPM) based on NSGA-II is proposed. A slight modification in the diversity maintenance strategy inside NSGA-II enables INSPM to distinguish preferable regions within an estimate of the Pareto-optimal front. A parameter allows the control of the preferable regions density and provides from fronts in which there is no interference from the DM until fronts in which the preferred solution is apparent. In all situations the Pareto-front extent is guaranteed.

Finally, the NN-DM method is adapted to find a model for the DM's preferences in a polymer extrusion process. The DM's requirement is filling a matrix expressing the preferences considering ordinal comparisons. The NN-DM method is able to provide a model which sorts the alternatives from the best to the worst one according to the DM's preferences in a real scenario.

**Keywords:** Multi-criteria decision analysis, preference model, artificial neural network, multi-objective optimization, genetic algorithm.

# Resumo

Este trabalho apresenta uma metodologia baseada na teoria da utilidade multi-atributo para aproximar a função de utilidade de um tomador de decisão: o método NN-DM. A informação de preferência extraída do tomador de decisão (DM) envolve apenas descrição ordinal e é estruturada por um procedimento de ordenação parcial. Uma rede neural artificial é então construída para aproximar a ordenação parcial reproduzindo as preferências do DM em um domínio específico. O método NN-DM é apropriado em situações em que um processo de decisão recorrente deve ser realizado considerando diferentes conjuntos de alternativas e um mesmo DM.

Uma hibridização entre os métodos NN-DM e iTDEA também é desenvolvida neste trabalho. Considerando-se a mesma quantidade de informação de preferência, o método NN-DM é capaz de construir um modelo para as preferências do DM para guiar o iTDEA. Deste ponto em diante não são mais necessárias perguntas ao DM relacionadas a tomadas de decisão semelhantes.

Adicionalmente, o algoritmo INSPM, inspirado no NSGA-II, é proposto. Uma ligeira modificação na estratégia de manutenção da diversidade do NSGA-II possibilita ao INSPM distinguir regiões preferíveis em uma estimativa da fronteira Pareto-ótimo. Um parâmetro permite o controle da densidade nas regiões preferíveis e fornece desde fronteiras em que não há nenhuma interferência do DM até fronteiras em que a solução preferida é aparente. Em todas as situações a extensão da fronteira Pareto é garantida.

Finalmente, o método de NN-DM é adaptado para encontrar um modelo para as preferências do DM em um processo de extrusão de polímeros. O requisito ao DM é preencher uma matriz que expressa suas preferências considerando comparações ordinais. O método NN-DM é capaz de fornecer um modelo que classifica as alternativas da melhor para a pior de acordo com as preferências do DM em um cenário real.

**Palavras-chave:** Análise de decisão multi-critério, modelo de preferência, rede neural artificial, otimização multi-objetivo, algoritmo genético.



# Contents

<b>Acknowledgements</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Resumo</b>	<b>vi</b>
<b>Table of Contents</b>	<b>x</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xiv</b>
<b>List of Symbols</b>	<b>xv</b>
<b>List of Abbreviations</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Organization . . . . .	6
<b>2 Decision Models</b>	<b>10</b>
2.1 Introduction . . . . .	10
2.2 Classical Decision-making Methods . . . . .	11
2.2.1 Introduction . . . . .	11
2.2.2 ELECTRE Methods . . . . .	11
2.2.3 AHP Methods . . . . .	12
2.2.4 ROR Methods . . . . .	13
2.3 Modeling the DM's Preferences . . . . .	15
2.3.1 Introduction . . . . .	15
2.3.2 Artificial Neural Networks . . . . .	15
2.3.3 Other Techniques . . . . .	19
2.4 Requirements to the DM . . . . .	21

<b>3</b>	<b>Notation and Problem Statement</b>	<b>23</b>
3.1	Multi-Criteria Decision-Making Analysis . . . . .	23
3.2	Multi-Objective Optimization . . . . .	26
3.3	INSPM . . . . .	30
<b>4</b>	<b>The NN-DM Method</b>	<b>32</b>
4.1	Introduction . . . . .	32
4.2	The NN-DM Methodology . . . . .	33
4.2.1	Step 1: Domain Establishment . . . . .	34
4.2.2	Step 2: Ranking Construction . . . . .	37
4.2.3	Step 3: Artificial Neural Network Approximation . . . . .	39
4.2.4	Step 4: Performance Assessment . . . . .	46
4.2.5	DM calls . . . . .	48
4.3	The Algorithm . . . . .	49
4.4	Illustrative Examples . . . . .	50
4.5	Discussion . . . . .	52
<b>5</b>	<b>The Improved NN-DM Method</b>	<b>55</b>
5.1	Introduction . . . . .	55
5.2	Step 1 - Domain Establishment . . . . .	56
5.3	Step 2 - Ranking Construction . . . . .	57
5.3.1	Dominance . . . . .	57
5.3.2	The Improved Partial Ranking . . . . .	58
5.4	DM Calls . . . . .	59
5.5	The Algorithm . . . . .	60
5.6	Illustrative Examples . . . . .	61
5.6.1	Example A . . . . .	61
5.6.2	Example B . . . . .	64
5.7	Discussion . . . . .	67
<b>6</b>	<b>The NN-DM Method And iTDEA</b>	<b>69</b>
6.1	Introduction . . . . .	69
6.1.1	Interactive Algorithms . . . . .	69
6.2	TDEA, prTDEA, and iTDEA . . . . .	73
6.3	Computational Experiments . . . . .	77
6.4	Discussion . . . . .	82
<b>7</b>	<b>The NN-DM Method And NSGA-II</b>	<b>84</b>
7.1	Introduction . . . . .	84
7.2	The Adapted NN-DM Methodology . . . . .	86
7.2.1	Step 1 - Domain Establishment . . . . .	87

7.2.2	Step 2 - Ranking Construction . . . . .	87
7.2.3	Step 4 - Performance Assessment . . . . .	91
7.3	NN-DM Method and NSGA-II . . . . .	91
7.3.1	Dynamic Crowding Distance . . . . .	92
7.3.2	Neural Network Dynamic Crowding Distance . . . . .	93
7.4	The Algorithm . . . . .	94
7.4.1	NN-DM Model . . . . .	95
7.4.2	INSPM Main Program . . . . .	97
7.5	Computational Experiments . . . . .	100
7.5.1	INSPM and Utility Function . . . . .	102
7.5.2	INSPM and NN-DM Method . . . . .	104
7.5.3	Comparison with iTDEA . . . . .	106
7.6	Discussion . . . . .	109
<b>8</b>	<b>Polymer Extrusion Process</b>	<b>112</b>
8.1	Introduction . . . . .	112
8.2	Available Data . . . . .	113
8.3	Interaction with the DM . . . . .	115
8.4	The Adapted NN-DM Methodology . . . . .	119
8.4.1	Step 1: Domain Establishment . . . . .	119
8.4.2	Step 2: Ranking Construction . . . . .	120
8.4.3	Step 4: Performance Assessment . . . . .	120
8.4.4	Algorithm . . . . .	121
8.5	Computational Experiments . . . . .	122
8.6	Case Study . . . . .	124
8.7	Discussion . . . . .	130
<b>9</b>	<b>Conclusions and Ideas for Future Work</b>	<b>132</b>
	<b>Bibliography</b>	<b>136</b>
<b>A</b>	<b>The NEWRB Function</b>	<b>145</b>
A.1	Definition . . . . .	145
A.2	Description . . . . .	145
A.3	Algorithm . . . . .	146
A.4	Simulating the Network . . . . .	147
<b>B</b>	<b>A Comparison Between Mergesort and Quicksort</b>	<b>149</b>
B.1	Algorithms . . . . .	150
B.1.1	Quicksort . . . . .	150
B.1.2	Mergesort . . . . .	151

B.2 Results . . . . .	152
<b>C Decision-making Matrices - Polymer Extrusion Process</b>	<b>155</b>
<b>Index</b>	<b>163</b>

# List of Figures

1.1	Similar Pareto-optimal fronts. . . . .	5
4.1	Refinement of a regular two-dimensional domain $\mathcal{D}$ . . . . .	35
4.2	Refinement of a regular three-dimensional domain $\mathcal{D}$ . . . . .	36
4.3	Refinement of a generic domain. . . . .	36
4.4	Refinement of a Pareto-optimal front. . . . .	37
4.5	RBF network architecture. . . . .	40
4.6	Gaussian functions with $\sigma = 1, 2, 3$ . . . . .	41
4.7	Surface and level sets of the functions $\mathcal{U}$ and $\hat{\mathcal{U}}$ . . . . .	45
4.8	Example of a function $\hat{\mathcal{U}}$ being employed. . . . .	45
4.9	DM's underlying utility functions. . . . .	51
4.10	Partial ranking. . . . .	51
4.11	Models $\hat{\mathcal{U}}$ obtained by the NN-DM method. . . . .	52
4.12	Partial ranking examples. . . . .	53
5.1	Domain establishment. . . . .	57
5.2	DM's underlying utility function $\mathcal{U}$ . . . . .	62
5.3	Partial ranking with $n = 50$ alternatives. . . . .	63
5.4	Model $\hat{\mathcal{U}}$ for the DM's preferences. . . . .	63
5.5	DM's underlying utility function $\mathcal{U}$ . . . . .	65
5.6	Partial ranking with $n = 50$ alternatives. . . . .	65
5.7	Model $\hat{\mathcal{U}}$ for the DM's preferences. . . . .	66
5.8	Two-dimensional instance: number of queries and KTD. . . . .	66
5.9	Three-dimensional instance: number of queries and KTD. . . . .	67
5.10	Partial ranking examples. . . . .	68
6.1	Different territory sizes in two dimensions. . . . .	75
6.2	Estimates of the Pareto-optimal front from iTDEA and NN-DM methods. . . . .	79
6.3	Statistical values. . . . .	80
6.4	Estimates of the Pareto-optimal front from iTDEA and NN-DM methods. . . . .	81

6.5	Statistical values. . . . .	82
7.1	Ranking examples. . . . .	89
7.2	DM's underlying utility functions. . . . .	101
7.3	Utility function $\mathcal{U}_1$ : results for $w = -1$ , $w = 0$ , and $w = 0.5$ . . . . .	103
7.4	Utility function $\mathcal{U}_1$ : results for $w = 1$ , $w = 2$ , and $w = 5$ . . . . .	103
7.5	Utility function $\mathcal{U}_1$ : INSPM results. . . . .	105
7.6	Level sets of the functions $\mathcal{U}$ and $\hat{\mathcal{U}}$ . . . . .	107
7.7	Comparison between iTDEA and INSPM methods. . . . .	108
8.1	Available estimates of the Pareto-optimal fronts. . . . .	115
8.2	General NN-DM model: $\mathbf{Q} \times \mathbf{P}$ . . . . .	122
8.3	General NN-DM model: $\mathbf{Q} \times \mathbf{W}$ . . . . .	123
8.4	NN-DM model sorting in the problem $\mathbf{Q} \times \mathbf{P}$ . . . . .	123
8.5	NN-DM model sorting in the problem $\mathbf{Q} \times \mathbf{W}$ . . . . .	124
8.6	NN-DM models applied to the estimates of the Pareto-optimal front. . . . .	126
8.7	Comparison among models $\hat{\mathcal{U}}_1$ , $\hat{\mathcal{U}}_2$ , $\hat{\mathcal{U}}_3$ , and $\hat{\mathcal{U}}_4$ in EPF $\mathbf{QW}_1$ . . . . .	127
8.8	EPF $\mathbf{QW}_1$ embedded in two different domains. . . . .	129
8.9	Level sets of the resulting NN-DM model constructed based on the domain of EPF $\mathbf{QW}_1$ . The colorbar indicates the modeled DM's preferences. . . . .	130
B.1	Average number of comparisons considering Quicksort and Mergesort. . . . .	152
B.2	Average number of comparisons considering four sorting algorithms. . . . .	153

# List of Tables

2.1	The fundamental scale of absolute numbers (AHP).	13
2.2	The set of the utility function linguistic variable's values.	19
4.1	MATLAB <sup>®</sup> parameters of the NEWRB function.	44
4.2	Example of the KTD metric.	47
4.3	Concordant and discordant pairs.	47
4.4	DM's underlying utility functions.	50
5.1	MATLAB <sup>®</sup> parameters: the NEWRB function.	64
6.1	iTDEA parameters.	77
6.2	NN-DM parameters.	78
7.1	Parameters of the INSPM algorithm.	100
7.2	KTD and number of queries in INSPM.	106
7.3	iTDEA parameters.	109
8.1	Multi-objective optimization problems in a single screw extrusion process.	114
8.2	Objectives, aim of optimization, and range of variation.	114
8.3	Example of a decision-making matrix $\mathcal{M}$ .	116
8.4	Example of a filled decision-making matrix $\mathcal{M}$ .	117
8.5	Partitions of each objective.	118
8.6	Sub-matrices with different preferences.	125
A.1	Parameters of the NEWRB function.	146
A.2	Inputs of the SIM function.	147
A.3	Outputs of the SIM function.	148
B.1	Average number of comparisons spent for sorting a list.	153
C.1	Unfilled decision-making matrix: $\mathbf{Q} \times \mathbf{P}$ .	156
C.2	Filled decision-making matrix: $\mathbf{Q} \times \mathbf{P}$ .	157
C.3	Unfilled decision-making matrix: $\mathbf{Q} \times \mathbf{W}$ .	158

C.4	Filled decision-making matrix: $\mathbf{Q} \times \mathbf{W}$ – Matrix $M_1$ .	159
C.5	Filled decision-making matrix: $\mathbf{Q} \times \mathbf{W}$ – Matrix $M_2$ .	160
C.6	Filled decision-making matrix: $\mathbf{Q} \times \mathbf{W}$ – Matrix $M_3$ .	161
C.7	Filled decision-making matrix: $\mathbf{Q} \times \mathbf{W}$ – Matrix $M_4$ .	162



# List of Symbols

$a$	An available alternative of the MCDM problem	23
$\mathcal{A}$	Set of alternatives of the MCDM problem	23
$\mathcal{C}$	Set of criteria of the MCDM problem	24
$\mathcal{C}_i$	A criterion of the MCDM problem	24
$d$	Dimension of the decision-making problem	35
$\mathcal{D}$	Domain of the approximation	33
$\delta$	Function which represents the grid's position	116
$\mathbf{f}$	Objective function in a general MOOP	26
$f_i$	Component of the objective function in a general MOOP	26
$F_i$	Preferred alternatives related to the pivot $v_i$	37
$F$	Feasible set	27
$\mathcal{F}$	Set of simulated alternatives	29
$g_i$	Inequality constraint in a general MOOP	26
$G$	Grid of simulated alternatives	35
$h_i$	Equality constraint in a general MOOP	26
$k$	Number of pivot alternatives	58
$L_{\text{melt}}$	The length of screw required to melt the polymer	112
$m$	Number of objective functions in a general MOOP	27
$M$	Matrix of the underlying utility function	64
$\mathcal{M}$	Decision-making matrix	115
$m_{ij}$	Element of the decision-making matrix $\mathcal{M}$	115
$n_i$	Number of alternatives linearly spaced in each sub-dimension of the domain $\mathcal{D}$	35
$n_{in}$	Number of random simulated alternatives consid- ered in constructing the initial NN-DM model	87
$n_{step}$	Number of random simulated alternatives added in each NN-DM model update	87
$nvp$	Number of points in each validation set	48
$nvs$	Number of validation sets	48
$N$	Number of radial basis functions	40
$N_{pop}$	Number of individuals in the genetic population	97

$p$	Number of inequality constraints in a general MOOP	27
$\mathbf{p}$	Image of an available alternative of the MCDM problem	24
$P$	DM's preference function	24
$\mathcal{P}$	Set of alternatives	37
$\mathbf{P}$	The power consumption required to rotate the screw	112
$\mathbf{P}_{\max}$	The capacity of pressure generation	112
$\mathcal{P}$	Estimate of the Pareto-optimal front	27
$\mathcal{P}_{\text{NN}}$	Estimate of the Pareto-optimal front and the NN-DM model	31
$\mathcal{P}_{\text{DM}}$	Estimate of the Pareto-optimal front and the utility function $\mathcal{U}$	31
$\phi$	Radial basis function	40
$q$	Number of equality constraints in a general MOOP	27
$\mathbf{Q}$	The mass output of the machine	112
$\mathcal{R}$	Function which provides the ranking of an alternative	38
$\sigma$	Parameter of the Gaussian function	41
$\mathbf{T}_{\text{melt}}$	The average melt temperature of the polymer at die exit	112
$T(n)$	Approximated number of queries to the DM	48
$T_i$	Non-preferred alternatives related to the pivot $v_i$	37
$\tau$	Average Kendall-tau distance value	47
$tol_{up}$	Tolerance for the KTD value regarding the NN-DM model updating	98
$tol_{st}$	Tolerance for the KTD value regarding the NN-DM model stability	48
$\mathcal{U}$	Utility function	24
$\hat{\mathcal{U}}$	Approximation of the utility function $\mathcal{U}$	33
$\hat{\mathcal{U}}_c$	Current ANN obtained by the NN-DM method	31
$\hat{\mathcal{U}}_f$	Former ANN obtained by the NN-DM method	31
$v_i$	Pivot alternative chosen randomly in the stage $i$	37
$w$	Parameter to control the preferable regions density	93
$\mathbf{W}$	The mixing capacity measure by the average of deformation	112
$w_i$	Weight of the RBF network	40
$\mathbf{x}$	Input of the neural network	40
$\mathbf{x}_i$	Center of each radial basis function	40
$X$	Vector of decision variables in a general MOOP	27
$y$	Neural network approximating function	40

# List of Abbreviations

ALENA	Artificial Life Evolving from Natural Affinities	4
AHP	Analytic Hierarchy Process	12
ANFIS	Adaptive Neuro-Fuzzy Inference System	19
ANN	Artificial Neural Network	15
AWTP	Augmented Weighted Tchebycheff Programs	16
BC-EMO	Brain-Computer Evolutionary Multi-objective Optimization	21
CD	Crowding Distance	91
DCD	Dynamic Crowding Distance	91
DM	Decision-Maker	1
DMS	Diversity Maintenance Strategy	92
DNN	Decision Neural Network	17
DSM	Downhill Simplex Method	19
EPF	Estimate of the Pareto-optimal Front	29
ELECTRE	ELimination Et Choix Traduisant la REalité (ELimination and Choice Translating REality)	11
EMO	Evolutionary Multi-objective Optimization	84
FFANN	Feed-Forward Artificial Neural Network	15
FIS	Fuzzy Inference System	19
INSPM	Interactive Non-dominated Sorting algorithm with Preference Model	85
IPOA	Interactive Polyhedral Outer Approximation	21
iTDEA	Interactive Territory Defining Evolutionary Algorithm	70
IWTP	Interactive Weighted Tchebycheff Procedure	16
KTD	Kendall-Tau Distance	46
LMS	Least Mean Squares	43
MATLAB <sup>®</sup>	MATrix LABoratory	6
MAUT	Multi-Attribute Utility Theory	2
MCDA	Multi-Criteria Decision Analysis	2
MCDM	Multi-Criteria Decision-Making	2
MCDS	Multi-Criteria Decision Support	18

MLP	Multi-Layer Perceptron .....	17
MOOP	Multi-Objective Optimization Problem .....	4
NN-DCD	Neural Network Dynamic Crowding Distance .....	91
NN-DM	Neural Network Decision-Maker .....	33
NSGA-II	Non-dominated Sorting Genetic Algorithm-II .....	85
PF	Pareto-optimal front .....	27
PI-EMO-VF	Progressively Interactive EMO approach using Value Functions .....	20
prTDEA	Preference-based TDEA .....	73
R-NSGA-II	Reference-point-based NSGA-II .....	70
RBF	Radial Basis Function .....	39
ROR	Robust Ordinal Regression .....	13
RPSGA <sub>e</sub>	Reduced Pareto Set Genetic Algorithm with Elitism .	113
RSO	Reactive Search Optimization .....	20
SBX	Simulated Binary Crossover .....	97
SVD	Singular Value Decomposition .....	43
TDEA	Territory Defining Evolutionary Algorithm .....	73
UTA	UTilités Additives .....	14
UTA <sup>GMS</sup>	UTilités Additives revisited by Greco, Mousseau, and Słowiński .....	14
ZDT4	Acronym inspired in Zitzler, Deb, and Thiele .....	101

# Chapter 1

## Introduction

One of the most common actions to human beings is decision-making. Each person is constantly making decisions on a variety of different subjects. There are all kinds of decisions: easy and difficult, important and irrelevant, personal and professional. It is well-known that there are good and bad decisions. So, a natural question is asked: is there a procedure for making a decision to ensure that the final result reflects a good decision?

Several efforts have been undertaken to explore this issue. Psychologists studied how decision-makers work under different circumstances and philosophers have questioned whether there is really a good decision. Logic contributed to the understanding of the process of decision-making and mathematics, including statistics, provided a formal structure for the process, defining criteria for optimality.

A decision-making problem can be visualized as a situation in which a person, called decision-maker and denoted by DM, has to select the best alternative (or action) belonging to a set of alternatives. The DM should then

express her/his preferences toward the elements of this set and the solution of the problem is the DM's preferred alternative. As each alternative is often associated with several attributes the problem becomes a Multi-Criteria Decision-Making (MCDM) problem.

The Multi-Criteria Decision Analysis (MCDA) is a research area composed of methods and techniques for assisting or supporting people and organizations in decision-making. At present there are several methods for decision-making and this number grows every day. All methods claim to solve decision-making problems, but in several situations different methods produce different results for the exactly same problem; even simple problems with few alternatives and criteria. Among the studies comparing decision-making methods one stands out to make clear an important question: what decision-making method should be employed in choosing the best decision-making method? [Triantaphyllou and Mann, 1989]

One main theoretical tendency in mathematical modeling of decision-making problems is the decision based on the Multi-Attribute Utility Theory (MAUT). MAUT assumes that there exists a function  $\mathcal{U}$ , denoted utility function, which reproduces the DM's preferences. This function assigns a scalar value to the alternatives which can then be sorted by the simple comparison of the values [Keeney and Raiffa, 1976]. The MAUT-based methods are appropriate in situations in which there is a previous complete knowledge of all necessary information about the problem leading to well-structured preferences for the DM.

In current MAUT methodologies, the problem setting is usually stated as: given a decision problem, with its set of possible solutions (the alter-

natives), establish a rational route to find a acceptable solution, under the DM's viewpoint. It is not always possible to assume that the DM is able to inform the cardinal value of her/his preferences on any alternative; instead, the DM is usually able to supply only ordinal information, stating that an alternative  $a_i$  is better than an alternative  $a_j$  or that  $a_j$  is better than  $a_i$ , or still that those alternatives are equivalent. Also, the DM is usually able to perform comparisons within a set with few alternatives only, being unable to process large sets properly.

The context of the present work is a problem that should be solved several times in instances that differ from one to another in certain decision parameters that affect the preferences and the set of available alternatives. Nevertheless, it is expected that the decision-making over similar sets of decision parameters and available alternatives leads to similar decisions. Indeed, there should be a structure for the DM's preferences that can be assumed to be valid in all such problem instances. This structure can be arbitrary and possibly presenting non-linear dependencies among several decision criteria. The aim of this thesis is to present a methodology for the extraction of such preference structure in the form of a neuron's network function<sup>1</sup> that reproduces the preference relations obtained from the DM. This function performs a kind of regression on the DM answers about her/his preferences delivering new answers to alternatives that were not evaluated.

The specific structure of interaction with the DM assumed here requires only holistic judgments, considering situations in which the DM should evaluate a solution as a whole, instead of weighting the criteria employed in

---

<sup>1</sup>Mathematically, a neuron's network function is defined as a composition of other functions, which can further be defined as a composition of other functions.

constructing this solution. For instance, in generative art image, it would be meaningless to ask a DM for the relative importance of features such as brightness or contrast. A more meaningful query is formulated as what is the preferred image, considering certain given alternatives.

Due to the holistic judgments the methodology introduced in this work can be applied to a promising field called computational creativity which embraces the idea of a machine that makes art. In his book [Steiner \[2012\]](#) tells the story of how the rise of computerized decision-making affects every aspect of business and daily life. Interesting examples are Emily Howell [[Cope, 2005](#)], a computer program with an interactive interface that allows both musical and language communication, and ALENA (Artificial Life Evolving from Natural Affinities) [[Cope, 2011](#)], a program which produces art from mathematical formulas derived from calculations created by nonlinear functions.

Another example in which this methodology can be applied is a Multi-Objective Optimization Problem (MOOP) in which the DM has to choose the preferred solution in the Pareto-optimal front several times in different instances. In this kind of recurrent situation a preference model, which can be employed repeatedly either in avoiding further queries to the DM or in reducing the number of necessary queries, constitutes a relevant enhancement in relation to single-time use models. A recurrent decision situation can happen, for instance, when a product is to be manufactured several times in batches, in each case with operational conditions that are different from the other cases (for instance: different availability or cost of raw materials, different loading conditions of the required logistic systems, etc.). In each case there exists different optimization problems, with slightly different con-



straints and objective functions, but in all those cases the DM's preferences remain the same.

Figure 1.1 exemplifies an analytical situation with four Pareto-optimal fronts: each front represents a solution of a MOOP instance. In this situation the similar Pareto-optimal fronts belong to the same domain and represent different sets of alternatives evaluated by the same DM.

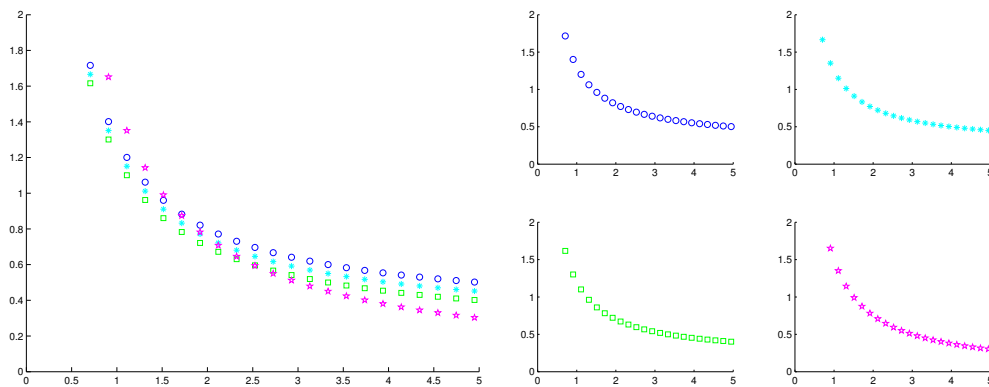


Figure 1.1: Similar Pareto-optimal fronts.

The proposed methodology also takes into account certain aspects of the geometric structure in finding an approximation of the DM's preferences. The assignment of space coordinates to the alternatives provides a geometric structure to the utility function making possible a regression process. This means that alternatives with similar coordinates in the feature space (the space in which the available alternatives with the corresponding decision parameters are embedded) should have similar preference values, that is, the utility function should be modeled as a continuous function. The resulting function may guide the search for the preferred alternative from any set of alternatives, even when none of such alternatives has already been considered,

relying only on the information about points that belong to the same region of the space.

This thesis presents a methodology for the construction of a function which models the DM's preferences: the Neural Network Decision-Maker method (NN-DM method). In this methodology, compatible with the MAUT assumptions, a function is built from a partial ranking process based on the ordinal information provided by the DM. This function is then employed in quantifying the preferences within a specific domain. An artificial neural network is the technique chosen to construct the approximating function that should have level sets which coincide with the ones of the DM's utility function. The resulting function is able to model the DM's preferences and can be employed in avoiding the formulation of new queries to the DM in new instances of the same decision-making problem.

For executing this work all data processing was performed off-line by the commercial software package MATLAB<sup>®</sup> [MathWorks, 2009] on a microcomputer with the following configuration: CPU Intel Core i3-3227U 1.90GHz, RAM Memory 6GB, and operating system Windows 8 64-bit.

## 1.1 Organization

This thesis is organized as follows:

**Chapter 2** presents an extensive review of the literature, including decision-making methods, interactive algorithms, different models for the DM's preferences, and real-world applications.

**Chapter 3** provides the problem statement and the notation. The problem statement is presented for the main areas considered in this thesis: multi-objective optimization problem and multi-criteria decision-making problem. The notation employed along the thesis is presented in this chapter and can also be checked in the List of Symbols.

**Chapter 4** introduces the NN-DM method which is an original methodology developed in this work. This method is based on the construction of a partial ranking from a grid of alternatives considering ordinal information only from the DM. An artificial neural network is employed in approximating this ranking creating a model for the DM's preferences: the NN-DM model. This work produced a paper in the Congresso Brasileiro de Redes Neurais with the following details:

[[Pedro and Takahashi, 2009](#)]

L. R. Pedro and R. H. C. Takahashi. Modeling the decision-maker utility function through artificial neural networks. In *Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)*, volume 1, 2009.

**Chapter 5** introduces improvements in the NN-DM method. The basic modifications are: the domain is now composed of random alternatives and the ranking is constructed based on the total sorting of a subset of alternatives. These changes are developed to produce a stable partial ranking which provides extra information to the artificial neural network. Additionally, the dominance among the alternatives is considered likely reducing the number of queries to the DM. This work

produced a paper in the 6th International Conference on Evolutionary Multi-criterion with the following details:

[[Pedro and Takahashi, 2011](#)]

L. R. Pedro and R. H. C. Takahashi. Modeling decision-maker preferences through utility function level sets. In *6th International Conference on Evolutionary Multi-criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 550–563. Springer Berlin Heidelberg, Ouro Preto, Brasil, 2011.

**Chapter 6** describes the iTDEA method and employs the improved NN-DM method for constructing a model for the DM’s preferences inside iTDEA. The iTDEA methodology is preserved and the NN-DM model replaces the original DM in the interactive process. This work produced a paper in the 7th International Conference on Evolutionary Multi-criterion with the following details:

[[Pedro and Takahashi, 2013](#)]

L. R. Pedro and R. H. C. Takahashi. Decision-maker preference modeling in interactive multiobjective optimization. In *7th International Conference on Evolutionary Multi-criterion Optimization*, volume 7811 of *Lecture Notes in Computer Science*, pages 811–824. Springer Berlin Heidelberg, Sheffield, UK, 2013.

**Chapter 7** describes an evolutionary algorithm which progressively interacts with the DM called INSPM. In INSPM the NSGA-II methodology

is almost entirely preserved, except for the original diversity mechanism (crowding distance) which is replaced with the NN-DCD, a dynamic crowding distance weighted by the NN-DM model. This work produced a paper in the Information Sciences Journal with the following details:

[[Pedro and Takahashi, 2014](#)]

L. R. Pedro and R. H. C. Takahashi. Inspm: An interactive evolutionary multi-objective algorithm with preference model. *Information Sciences*, 268(0):202–219, 2014.

**Chapter 9** presents the final conclusions and ideas for future work.

# Chapter 2

## Decision Models

### 2.1 Introduction

The purpose of this chapter is to examine the current methodologies employed in helping the DM to make her/his decision. Classical methods such as ELECTRE (Section 2.2.2) and AHP (Section 2.2.3) are considered, as long as new approaches like ROR (Section 2.2.4). Alternative methods in which a direct approximation of the DM's preferences is constructed are also examined. Specifically, a special attention is given to the artificial neural networks (Section 2.3.2), focus of the proposed work, but other techniques are also covered in this review (Section 2.3.3). The requirements to the DM are presented throughout each method's description and a comparison with the demands of the proposed method is discussed in Section 2.4.

## 2.2 Classical Decision-making Methods

### 2.2.1 Introduction

This section briefly reviews two popular MCDM methods, ELECTRE and AHP, and a promising methodology called Robust Ordinal Regression (ROR). ELECTRE is a decision making method based on outranking relationships, AHP uses pairwise comparisons to compare the alternatives and estimate criteria weights and ROR implements an interactive preference construction paradigm recognized as a mutual learning of the model and the DM's preferences.

### 2.2.2 ELECTRE Methods

The ELECTRE methods comprise a family of MCDM methods that originated in France during the middle of the 1960s. The acronym ELECTRE stands for ELimination Et Choix Traduisant la REalité (ELimination and Choice Translating REality). The method was first proposed by Roy [1968] and his colleagues at Société d'Economie et de Mathématiques Appliquées (SEMA). There are two main parts to an ELECTRE application: first, the construction of one or several outranking relationships<sup>1</sup>, which aim at comparing in a comprehensive way each pair of actions; second, an exploitation procedure that elaborates on the recommendations obtained in the first phase. The research on ELECTRE methods is still evolving and gaining acceptance thanks to new application areas, new methodological and theoretic-

---

<sup>1</sup>Formally, an outranking relationship states that even though two alternatives  $a_i$  and  $a_j$  do not dominate each other, it is realistic to accept the risk of regarding  $a_i$  as almost surely better than  $a_j$ .

cal developments, as well as user-friendly software implementations. Recent applications of ELECTRE methods can be found in: assisted reproductive technology [Matias, 2008], promotion of social and economic development [Rangel *et al.*, 2009], sustainable demolition waste management strategy [Roussat *et al.*, 2009], assessing the risk of nanomaterials [Tervonen *et al.*, 2009], and unequal area facility layout problems [Aiello *et al.*, 2013].

### 2.2.3 AHP Methods

The Analytic Hierarchy Process (AHP) is a structured technique for organizing and analyzing complex decisions based on mathematics and psychology. It was developed by Thomas L. Saaty in the 1970s [Saaty, 1977] and it has been extensively studied and refined since then. The AHP involves a theory of measurement through pairwise comparisons and relies on the DM's judgments to derive priority scales. The comparisons are made considering a scale of absolute judgments (Table 2.1) that reproduces how much more one element dominates another with respect to a given attribute. In an attempt to improve the judgments, which may be inconsistent, the derived priority scales are synthesized by multiplying them by the priority of their parent nodes and adding for all such nodes. Therefore, the DM not only needs to create priorities for the alternatives with respect to the criteria or sub-criteria, but also for the criteria themselves. The AHP has been employed in making decisions in several scenarios and Saaty [2008] includes an extensive list of applications.



Equal Importance	1
Weak or slight	2
Moderate importance	3
Moderate plus	4
Strong importance	5
Strong plus	6
Very strong or demonstrated importance	7
Very, very strong	8
Extreme importance	9

Table 2.1: The fundamental scale of absolute numbers (AHP).

## 2.2.4 ROR Methods

The Robust Ordinal Regression (ROR) has been proposed with the purpose of taking into account the sets of parameters compatible with the DM's preference information.

[Angilella \*et al.\* \[2004, 2010\]](#) proposed a non-additive ROR on a set of alternatives whose utility is evaluated considering the Choquet integral<sup>2</sup>. The interaction among the criteria can then be modeled by fuzzy measures parameterizing the approach. The DM is requested to answer holistic pairwise preference comparisons on the alternatives and on the importance of criteria and to express the intensity of the preference on specific pairs of alternatives and pairs of criteria. The output is a set of fuzzy measures (capacities) such that the corresponding Choquet integral is compatible with the DM's preference information. Recently, [Corrente \*et al.\* \[2013\]](#) drew attention upon recent advances in ROR clarifying the specific interpretation of the concept

---

<sup>2</sup>The Choquet integral is the discrete form of the generalization of the Lebesgue integral with respect to fuzzy measures

of preference learning adopted in ROR and MCDA.

Greco *et al.* [2008] presented a method called  $UTA^{GMS}$  which generalizes the UTA method [Jacquet-Lagreze and Siskos, 1982]. The  $UTA^{GMS}$  method considers a set of additive value functions resulting from an ordinal regression for multiple criteria ranking of a set of alternatives. The following preference information is required from the DM:

- pairwise preference comparison on the alternatives from a reference set;
- the intensity of preference of a pair of alternatives, say  $a$  over  $b$ , in comparison to the intensity of preference of another pair of alternatives, say  $c$  over  $d$ ;
- pairwise comparison of importance of criteria;
- pairwise comparison of the differences between importance of criteria;
- negative and positive interaction expressing redundancy or synergy between couples of criteria;
- pairwise comparison of interaction intensity among couples of criteria;
- pairwise comparison of the differences of interaction intensity among couples of criteria.

The resulting preference model is the set of all additive value functions compatible with the preference information.

## 2.3 Modeling the DM's Preferences

### 2.3.1 Introduction

Several algorithms have been developed to model the DM's preferences directly, considering that those preferences are already well defined by the DM and can be reproduced by a utility function. Two scenarios are usually proposed:

1. a direct model of the DM's preferences, employed in making the decisions inside the method, and
2. a model employed in defining the DM's preferences used as entrance to classic methods.

Next sections presents algorithms intended to directly model the DM's preferences based on an underlying utility function employing artificial neural networks (Section 2.3.2), which is the tool employed in the NN-DM method, as well as other techniques (Section 2.3.3).

### 2.3.2 Artificial Neural Networks

Several previous works have already exploited the idea of representing the DM's preferences employing Artificial Neural Networks (ANNs). The main difference between the following methods and the methodology proposed in this thesis is the way the information is required from the DM.

The first work found in this category was proposed by [Sun \*et al.\* \[1996\]](#). The Interactive FFANN Procedure is an interactive procedure for solving

multiple objective programming problems based on Feed-Forward Artificial Neural Networks (FFANNs). In this method, the DM articulates preference information over representative samples from the non-dominated set. The preference is extracted from the DM either by assigning preference values to the sample solutions or by making pairwise comparisons answering questions similar to those posed in the AHP [Saaty, 1977]. The revealed preference information is considered training a FFANN which solves an optimization problem to search for improved solutions. In the computational experiments four different value functions of  $L_p$ -metric form with  $p = 1$ ,  $p = 2$ ,  $p = 4$ , and  $p = \infty$  are chosen to simulate the DM. The efficiency is measured by the quality of the worst, best, and average non-dominated point.

Sun *et al.* [2000] proposed a new interactive multiple objective programming procedure that combines the Interactive Weighted Tchebycheff Procedure (IWTP) [Steuer and Choo, 1983] and the interactive FFANN procedure [Sun *et al.*, 1996]. In this procedure, non-dominated solutions are built by solving Augmented Weighted Tchebycheff Programs (AWTP) [Steuer, 1986]. The DM indicates preference information by directly assigning values to criterion vectors (which are later rescaled) or by making pairwise comparisons among non-dominated solutions answering questions similar to those presented in the AHP [Saaty, 1977]. The revealed preference information is considered by training a FFANN which selects new solutions for presentation to the DM on the next iteration. In the computational experiments, linear, quadratic,  $L_4$  metric and Tchebycheff metric<sup>3</sup> value functions are chosen to simulate the DM. The efficiency is measured by the quality of the final solution, the nadir point, and the worst non-dominated point (evaluated as

---

<sup>3</sup>Tchebycheff metric is defined by a set of Tchebycheff weight ranges.

the non-dominated extreme point that has the lowest preference value).

A method focusing an EMO search on specific areas of the Pareto-optimal front is developed by [Todd and Sen \[1999\]](#). The method performs interactions with the DM to model her/his general preferences employing a Multi-Layer Perceptron (MLP) network. The proposed EMO requires a second special population called Pareto population which stores all non-dominated solutions as they evolve over the generations. The preference process takes place at regular intervals of the EMO procedure and a preference set with ten individuals from the normal population is displayed to the DM. The system then gathers preference information by asking for a score between 0 and 1 for each member of the preference set and the adjusted training set is employed in training the MLP with back propagation. The preference surface is employed in scoring the Pareto individuals and then in selecting a set of individuals from the Pareto population which are re-inserted into the normal population promoting the search in the preferable regions. The method concentrates search effort on the regions of the Pareto surface of greatest interest to the DM which reflects in a variety in the density of the resulting Pareto solutions. However it is not clear how to control this density and the resources demanded from the DM are not intuitive.

[Chen and Lin \[2003\]](#) proposed a new approach for solving MCDM problems based on a Decision Neural Network (DNN) employed in capturing and representing the DM's preferences. The interactive DNN approach consists of four phases: identification, modeling, solving MCDM, and implementation. With the DNN an optimization problem is solved to search for the most desirable solution. The architecture involves two ANNs which process the criterion vectors leading to results whose ratio is calculated and delivered

as the final result. The DM is asked to indicate pairwise comparison results including approximate ratios or intervals.

[Golmohammadi \[2011\]](#) presented a fuzzy multi-criteria decision-making model based on a FFANN employed in capturing and representing the DM's preferences. The proposed model can consider historical data and update the database information for alternatives over time for future decisions. The DM's preferences are captured from pairwise comparisons with a scale similar to the AHP procedure. The regular procedure of pairwise comparison is improved by adding a scale in which an objective is compared with an ideal objective. The mean square error was employed in comparing the network and the desired outputs validating the obtained results.

Finally, a direct adaptive method of multi-objective optimization based on neural network approximation of the DM's preferences is introduced by [Karpenko \*et al.\* \[2010\]](#). The method considers a linguistic function assuming the values presented in [Table 2.2](#) and interactions between the DM and a Multi-Criteria Decision Support (MCDS) system. Each iteration consists of two phases: analysis phase, in which the DM evaluates the solution proposed by the MCDS system, and computation phase, in which the MCDS system produces an optimal solution. The DM's utility function is approximated by both MLP and RBF networks considering the components of a weighting coefficient vector as input and its linguistic function value as output.

Extremely bad	1
Very bad	2
Bad	3
Not very bad	4
Satisfactory	5
Quite good	6
Good	7
Very good	8
Excellent	9

Table 2.2: The set of the utility function linguistic variable's values.

After two years [Karpenko \*et al.\* \[2012\]](#) presented a continuation of the exploration described in [Karpenko \*et al.\* \[2010\]](#) in which an investigation of the MCDM problems was carried out with: MLP and RBF networks, Mamdani-type Fuzzy Inference System (FIS), Adaptive Neuro-Fuzzy Inference System (ANFIS), and a method based on Downhill Simplex Method (DSM). The research on the method effectiveness is tested in two two-dimensional two-objective problems and in one three-dimensional three-objective problem. Although all the techniques allow the achievement of the optimal solution, ANFIS and the MLP and RBF networks provided the best solution for the smallest number of iterations.

### 2.3.3 Other Techniques

Models for the DM's preferences constructed by different techniques are also available in the literature. These methods often assume a specific type of approximating function, but there are also methods in which the DM's utility function is modeled as a general function, as the artificial neural networks are able to deliver.

Yang and Sen [1996] designed linear goal programming models built to estimate piecewise linear local utility functions based on pairwise comparisons of efficient solutions as well as objectives. The models capture the DM's preference information and support the search for the best compromise solutions in multi-objective optimization.

Tangian [2002] considered a model for constructing quadratic utility functions from interviewing the DM. This interview estimate both cardinal and ordinal utility and it is designed to guarantee a unique non-trivial output of the model. The constructing of the quasi-concave utility function is then reduced to a problem of non-linear programming.

The Progressively Interactive EMO approach using Value Functions (PI-EMO-VF) [Deb *et al.*, 2010] is a preference-based methodology which is embedded in an EMO algorithm and leads the DM to the most preferred solution of her/his choice. For this purpose periodically the DM is supplied with a handful of currently non-dominated points and s/he is asked to rank the points from the best to the worst one. This preference information is considered in modeling a strictly monotone value function which drives the EMO algorithm in major ways: 1) in determining termination of the overall procedure, and 2) in modifying the domination principle, which directly affects EMO algorithm's convergence and diversity-preserving operators. It should be noticed that the polynomial value function captures the preference information related only to the points that had been considered in constructing it. A new model is required every time the DM is interrogated while the optimization process is running.

The methodology of Reactive Search Optimization (RSO) is adopted by



Battiti and Passerini [2010] for evolutionary interactive multi-objective optimization. The machine learning technique and the DM’s judgments are taken into account to build robust incremental models for the DM’s utility function. The Brain-Computer Evolutionary Multi-objective Optimization (BC-EMO) employs the technique of support vector ranking together with a k-fold cross-validation procedure in selecting the best kernel during the utility function training procedure. The DM’s interactions are made through pairwise comparisons considering only holistic judgments.

Finally, Lazimy [2013] proposed an Interactive Polyhedral Outer Approximation (IPOA) method which progressively constructs a polyhedral approximation of the DM’s preference structure and a polyhedral outer-approximation of the feasible set of the multi-objective optimization problems. The piecewise linear approximation of the DM’s preferences is constructed on the basis of two forms of preference assessments: an estimate of the local trade-off vector and the ranking of the new objective vector relative to the existing vectors.

## 2.4 Requirements to the DM

The classical approaches (Section 2.2) require from the DM information about her/his preferences that may not be intuitive, such as the outranking relations (ELECTRE), the scale of absolute judgments (AHP), and the holistic pairwise preference comparisons. The idea behind those methods is the construction of relations that help the DM to express her/his preferences considering that those preferences are not yet well defined by the DM. The techniques that model the preferences directly (Section 2.3) usually require

from the DM pairwise comparisons similar to the information demanded by the classic approaches or some sort of score or ranking of the alternatives.

The model proposed in this work demands from the DM only ordinal information about two alternatives. In this context, the assumption that the DM corresponds to a utility function is reasonable because the comparisons are made between instances that are familiar to her/him. The DM has only to choose her/his preferred alternative without making more elaborated comparisons or giving weights to alternatives or criteria. Based only on this ordinal information a complete model for the DM's preferences, the NN-DM model, is obtained. In this model, alternatives belonging to the same domain can be evaluated according to the DM's preferences without any further information from the DM. The resulting NN-DM model might be thought, then, as the function providing information about the DM's preferences to the methods presented in Sections 2.2 and 2.3. Therefore, with the NN-DM model the DM is indirectly supplying complex information such as weighted preferences through only ordinal information.

# Chapter 3

## Notation and Problem Statement

### 3.1 Multi-Criteria Decision-Making Analysis

The multi-criteria decision-making analysis consists of a set of methods and techniques for assisting or supporting people and organizations to make decisions, considering multiple criteria. The subject of this thesis is the class of decision-making problems in which the alternatives to the problem are directly presented to the DM. The DM needs to answer queries concerning the preferences which lead to the discovery of a model for her/his preferences. The problem considered here involves the following basic elements.

**A set  $\mathcal{A}$  of alternatives (possible actions or choices)**

This set can be discrete or continuous and it is considered the domain of the decision-making problem. Each element  $a \in \mathcal{A}$  corresponds to an available alternative and each feature of  $a$  provides a problem dimension.

### **A set $\mathcal{C}$ of criteria (possible consequences or attributes)**

Each alternative  $a \in \mathcal{A}$  has criteria which reflect the consequences of its execution. Each criterion represents a point of view modeled by a function  $\mathcal{C}_i : \mathcal{A} \rightarrow \mathbb{R}$ . Therefore, the values  $\mathbf{p} = \mathcal{C}(a)$ ,  $a \in \mathcal{A}$ , are the image of the available alternatives and represent the information on which the DM has to take her/his decision. Since the DM deals only with the image of the available alternatives, from now on each value  $\mathbf{p}$  will be called an alternative for a short notation.

### **A decision-maker**

The merit of each alternative  $\mathbf{p}$  is assigned by a person, called here decision-maker (DM). In the context assumed in this work, the DM formally corresponds to a utility function  $\mathcal{U}$  for which it is not possible to directly measure the values of  $\mathcal{U}(\mathbf{p})$ , for any  $\mathbf{p}$ . Only the ordinal information extracted from yes/no queries to the DM may be provided by a preference function  $P$  which encodes the preference relations among all pairs of alternatives. The best alternative  $\mathbf{p}^*$  is the one that maximizes the function  $\mathcal{U}$  in the set  $\mathcal{C}(\mathcal{A})$ .

The DM is responsible for presenting a solution for the decision-making problem, which can be stated as:

- provide the best alternative or a limited set of efficient alternatives;
- rank the alternatives from the best to the worst one;
- classify the alternatives in predefined homogeneous groups.

The problem presented in this work is to find an approximation of the utility function  $\mathcal{U}$  which expresses the DM's preferences. For this purpose, the preference function  $P$ , which provides ordinal information from the DM, is employed in extracting information from the DM about her/his utility function  $\mathcal{U}$ . For each pair of alternatives  $(p_i, p_j)$  the function  $P$  is given by Equation 3.1.

$$\begin{cases} P(p_i, p_j) = -1, & \text{if } p_i \text{ is preferable than } p_j, \\ P(p_i, p_j) = 0, & \text{if } p_i \text{ and } p_j \text{ are equivalent,} \\ P(p_i, p_j) = 1, & \text{if } p_i \text{ is less preferable than } p_j. \end{cases} \quad (3.1)$$

Although the function  $P$  is able to provide only ordinal relation about the DM's preferences, it has a direct connection with the utility function  $\mathcal{U}$ , as shown in Equation 3.2.

$$\begin{cases} P(p_i, p_j) = -1, & \text{if and only if } \mathcal{U}(p_i) > \mathcal{U}(p_j), \\ P(p_i, p_j) = 0, & \text{if and only if } \mathcal{U}(p_i) = \mathcal{U}(p_j), \\ P(p_i, p_j) = 1, & \text{if and only if } \mathcal{U}(p_i) < \mathcal{U}(p_j). \end{cases} \quad (3.2)$$

In this work it is assumed that the function  $P$  is defined for any pair of alternatives  $(p_i, p_j)$  and if two alternatives  $p_i$  and  $p_j$  are equally preferable a coin flip<sup>1</sup> decides each one is the preferred alternative. However, there is a major constraint on the information availability by considering the function  $P$ . As the DM is a human being, the answers to the comparisons between all pairs of alternatives may not be available, because there are limitations on time and patience. Therefore, the goal is to minimize the amount of queries required from the DM. This aim is achieved by selecting certain pairs of

---

<sup>1</sup>Coin flipping is the practice of throwing a coin in the air to choose between two alternatives, sometimes to resolve a dispute between two parties. It is a form of sorting which inherently has only two possible and equally likely outcomes.

alternatives for comparison and then exploring the acquired information to construct a suitable model for the DM's preferences.

As the proposed methodology assumes that the DM is a person, the inherit inconsistency of human-beings, which may lead to ranking reversals, is taking into the account. Luckily the regression approach proposed here regulates the final surface in relation to those ranking reversals so that the corresponding values do not play a significant role in building an adequate approximation of the DM's preferences.

Often, the cost or benefit of an alternative  $\mathbf{p}$  can be expressed through a function  $f$  dependent on decision variables. Therefore, the achievement of the best arrangement of the variables that maximizes this function leads to an optimization process, described next.

## 3.2 Multi-Objective Optimization

A Multi-Objective Optimization Problem (MOOP) is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Formally, a MOOP can be defined by Equation 3.3:

$$\min \mathbf{f}(X) = (f_1(X), f_2(X), \dots, f_m(X))$$

subject to

$$\begin{cases} g_i(X) \leq 0, & i = 1, 2, \dots, p \\ h_i(X) = 0, & i = 1, 2, \dots, q \end{cases}$$

(3.3)

in which  $f_i$  are the objective functions,  $g_i$  are the inequality constraints,  $h_i$  are

the equality constraints, and  $X = (x_1, x_2, \dots, x_N)$  is the vector of decision variables.

In a MOOP a set of different optimal solutions may exist where no single solution can be considered better than the others with respect to all the criteria. The feasible set, denoted  $F$ , is composed of the vectors  $X$  that satisfy all constraints. The solution set is then defined by the property of **dominance**. A vector  $X \in F$  is said to be **dominated** by another vector  $\bar{X} \in F$  if  $f_i(\bar{X}) \leq f_i(X)$  for all  $i = 1, \dots, M$  and there exists  $j \in \{1, \dots, M\}$  such that  $f_j(\bar{X}) < f_j(X)$ . The notation  $\bar{X} \prec X$  indicates that  $\bar{X}$  dominates  $X$ . The Pareto-optimal set  $\mathcal{P}$ , defined by Equation 3.4, is the MOOP's solution set.

$$\mathcal{P} = \{ \mathcal{X} \in \mathcal{F} \mid \nexists \bar{\mathcal{X}} \in \mathcal{F} \text{ such that } \bar{\mathcal{X}} \prec \mathcal{X} \} \quad (3.4)$$

The image of the Pareto-optimal set in the feature space is called Pareto-optimal front, or just Pareto-front (PF). In the absence of any additional subjective preference information, none of the PF solutions can be said to be inferior when compared to any other solution, as they are superior in at least one criterion.

Researchers study multi-objective optimization problems from different viewpoints and, thus, there exist solutions with different philosophies and goals when setting and solving them. The goal may be to find a representative set of solutions, and/or quantify the trade-offs in satisfying the different objectives, and/or find a single solution. In real-world MOOPs usually only one solution is chosen to be implemented. In order to obtain this single solution a decision-maker (DM) can make a choice regarding the importance of

the objectives in the optimization process or certain external criteria. Therefore, the final solution of a MOOP results from the combined optimization and decision processes which motivates the development of methods called preference-based methods. Preference-based methods are multi-objective optimization methods in which the relative importance attributed to the criteria is considered and the solution that best satisfies the DM's preferences is selected [Miettinen, 1999]. The preference-based methods can be divided into three different categories, as follows.

**A priori** The decision-maker must specify their preferences, expectations, and/or choices before the optimization process. The method consists of calculating a single criterion value by considering the individual criteria. The MOOP then becomes a single-objective optimization generating a single solution. The preferences can be expressed, for example, in terms of an aggregate function combining the individual objective values into a single utility value.

**A posteriori** All the criteria are optimized simultaneously and the Pareto-optimal set is obtained. The best solution can be chosen directly by the DM or selected based on the DM's preferences. The preferences can be expressed, for instance, in terms of an approximation of the utility function.

**Interactive** Decision-making and optimization are interleaved, that is, the DM must provide preference information about the current set of available solutions while the optimization process is running, leading the optimization algorithm during the search. Usually the outcome is one or a set of preferred solutions.



The selection of a single solution from the PF resultant of an optimization process requires information that may be not present in the objective functions. This information, expressing subjective preferences, must be introduced by the DM. The integration of the DM's preferences in the optimization procedure allows the distinction among the solutions in the estimate of the Pareto-optimal front (EPF) and, as a consequence, the selection of a single solution from the EPF. In the decision-making problem resultant from the MOOP the following elements are considered here.

#### **A set $\mathcal{A}$ of alternatives**

This set is composed by the decision variable space. Each alternative  $a \in \mathcal{A}$  represents a vector of decision variables.

#### **A set $\mathcal{C}$ of criteria**

This set is composed by the feature space. Each alternative  $\mathbf{p} = \mathcal{C}(a)$  represents a feasible solution of the MOOP.

#### **A decision-maker**

The utility function  $\mathcal{U}$  is assumed to be defined in the feature space, which means the criteria are the objectives of the MOOP. Since the DM evaluates the solutions in the feature space, from now on these solutions represent the available alternatives.

#### **A simulated decision-making problem**

A set  $\mathcal{F}$  with simulated alternatives is constructed in the feature space to obtain information from the DM about the entire domain in

which the utility function  $\mathcal{U}$  is being approximated. The set  $\mathcal{F}$  offers a kind of information which usually is not provided directly by the alternatives in the PF.

The idea behind the construction of a simulated decision-making problem is to find an appropriate model for the DM's preferences in the entire specified domain. When it comes to find the best alternative in a PF the majority of the algorithms in the literature considers only the information about the available alternatives which usually is enough to find the preferred alternative belonging to that specific set. However, as the dimension of the PF is smaller than the dimension of the feature space, this lack of information could compromise the method's performance. Therefore, the complete information about the feature space, available from the simulated alternatives, becomes crucial to constructing a precise model for the DM's preferences in the whole domain.

### 3.3 INSPM

Chapter 7 proposes an EMO methodology, called INSPM, developed from the NSGA-II algorithm interacting with the NN-DM method. In this scenario, the set  $\mathcal{A}$  is an estimate of the Pareto-optimal set and the corresponding PF works as a problem instance of the multi-criteria decision problem. Since NSGA-II is an evolutionary algorithm the set  $\mathcal{A}$  is discrete and each element  $\mathbf{p} = \mathbf{f}(a)$ ,  $a \in \mathcal{A}$ , corresponds to the image of an available alternative located on the current estimate Pareto-optimal set. Additionally, the interaction between the NN-DM method and the NSGA-II algorithm demands the definition of the following elements.

- A set  $\mathcal{P}_{\text{NN}}$  of all individuals in the current EPF obtained by NSGA-II guided by the NN-DM model.
- A set  $\mathcal{P}_{\text{DM}}$  of all individuals in the current EPF obtained by NSGA-II guided by the utility function  $\mathcal{U}$ . The set  $\mathcal{P}_{\text{DM}}$  is a reference to assess the performance of the proposed methodology.
- A function  $\hat{\mathcal{U}}_f$  which is the former artificial neural network obtained by the NN-DM method.
- A function  $\hat{\mathcal{U}}_c$  which is the current artificial neural network obtained by the NN-DM method.

# Chapter 4

## The NN-DM Method

### 4.1 Introduction

This work assumes that the DM is aware of her/his preferences at the beginning of the decision-making process and those preferences are defined regarding all the alternatives. The DM's answers are not quantitative, that is, given two alternatives  $p_i$  and  $p_j$ , with  $i \neq j$ , the alternative  $p_i$  is preferable than the alternative  $p_j$  or vice versa, but it is not possible to determine how much preferable is this solution.

Considering a single decision-making problem one simple way of finding the best solution is to perform the following procedure among the alternatives:

- an alternative is chosen and compared with the remaining ones; this alternative is called **pivot**;
- the alternatives which are preferable than the pivot pass to the next step;

- the process is repeated until only one alternative has left; this alternative is the preferred alternative on the DM's viewpoint.

According to MAUT, there exists a utility function  $\mathcal{U}$  which reproduces the DM's preferences assigning a scalar value to each alternative. The problem of finding an approximation  $\hat{\mathcal{U}}$  of the utility function  $\mathcal{U}$  can then be stated as a regression problem that should be performed over sampled points coming from  $\mathcal{U}$ . However, as only ordinal information can be obtained from  $\mathcal{U}$  by the preference function  $P$ , a partial ranking inspired by the described procedure is considered in constructing the regression. Once the function  $\hat{\mathcal{U}}$  is estimated it can be employed in quantifying any alternative within its domain and the DM's preferred alternative is the one with greatest value of  $\hat{\mathcal{U}}$ .

## 4.2 The NN-DM Methodology

The approach of modeling the DM's preferences in a general domain usually creates a more expensive process because the DM's preferences have to be modeled into an entire domain and not only for the available alternatives. The NN-DM method presents a way of finding not only the preferred solution, but a model, called here NN-DM model, that simulates the DM's preferences in a specific domain  $\mathcal{D}$  with similar demand from the DM when s/he is consulted to find only the preferred solution. The NN-DM model reproduces the DM's preferences in the domain  $\mathcal{D}$  considering information about the feature space. After the training, the model is able to find the preferred alternative in any problem instance within the considered domain. Once the NN-DM model is adjusted the preferred solution can be obtained by

employing the resulting model in attributing scalar values to each available alternative and then by choosing the solution which has the higher value. The resulting NN-DM model can also be employed in taking decisions when similar decision-making problems are presented.

The original NN-DM method is divided into four steps.

**Step 1** Domain Establishment

Select the domain for the utility function approximation and construct a grid of simulated alternatives.

**Step 2** Ranking Construction

Build a partial ranking for the alternatives assigning a scalar value to each alternative.

**Step 3** Artificial Neural Network Approximation

Construct an artificial neural network which interpolates the results and represents the DM's preferences.

**Step 4** Performance Assessment

Evaluate the resulting model according to the DM's preferences.

The next subsections present the details of those steps.

### **4.2.1 Step 1: Domain Establishment**

The domain  $\mathcal{D}$  of the approximation  $\hat{U}$  is defined as the smallest hyper-box with edges parallel to the coordinate axes that contains the set of available

alternatives  $\mathcal{A}$ . In this domain a simulated decision-making problem in which the alternatives are located as a grid is built. The queries to the DM are presented over the simulated decision-making problem, that is, the available alternatives are not directly considered in the model's construction. The grid of alternatives is built to find a uniform representation of the utility function in the domain  $\mathcal{D}$ . The number of alternatives in each dimension of the grid is related to the quality of  $\hat{\mathcal{U}}$ : the bigger the refinement, the better the approximating function, but also a higher number of queries is required from the DM.

Consider a decision-making problem in a space with dimension  $d$  and let  $n_i$  be the number of alternatives linearly spaced in each sub-dimension of the domain  $\mathcal{D}$ , with  $i = 1, 2, \dots, d$ . The grid of simulated alternatives  $G$  is constructed by the intersection of the refinements in each sub-dimension of  $\mathcal{D}$ . As each sub-dimension has  $n_i$  alternatives, the set  $G$  has  $\prod_{i=1}^n n_i$  alternatives. Figures 4.1 and 4.2 present a visualization of the described procedure in two and three-dimensional domains, respectively.

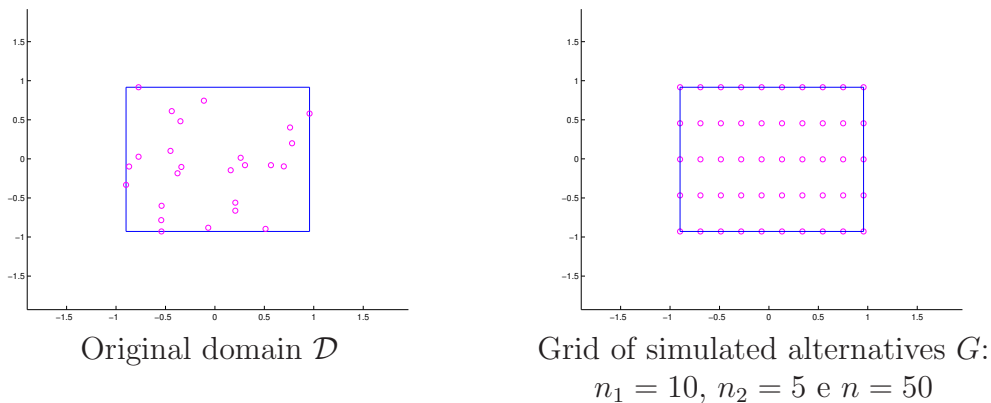


Figure 4.1: Refinement of a regular two-dimensional domain  $\mathcal{D}$ .

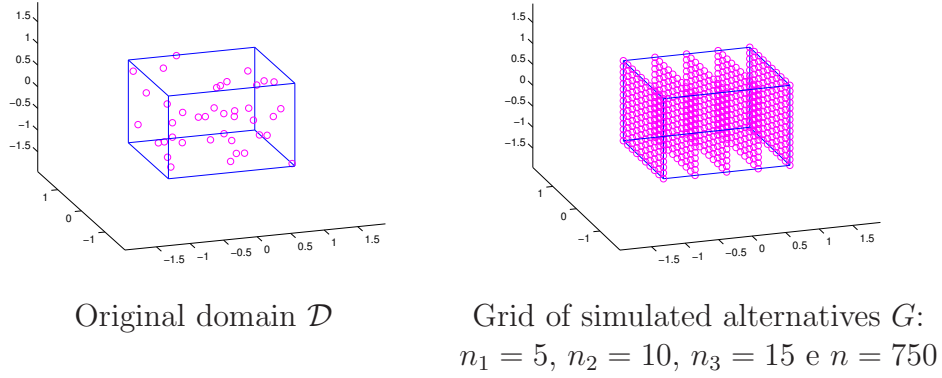


Figure 4.2: Refinement of a regular three-dimensional domain  $\mathcal{D}$ .

Considering a generic domain, the internal and external refinements can be defined, as shown in Figure 4.3.

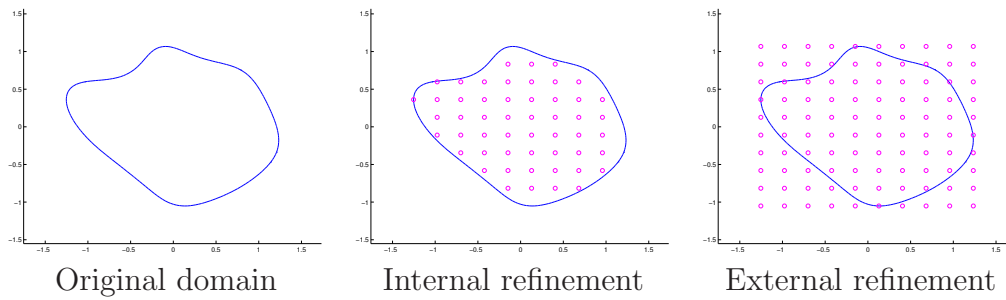


Figure 4.3: Refinement of a generic domain.

Considering now alternatives on a curve, for instance a PF derived from a multi-objective optimization, an external refinement is constructed around the curve. The external refinement generates the information to find a suitable model for DM's preferences in the whole domain. Figure 4.4 illustrates an external refinement of a PF obtained in a two-objective optimization problem. Note that the solutions on the PF are not directly part of the



refinement.

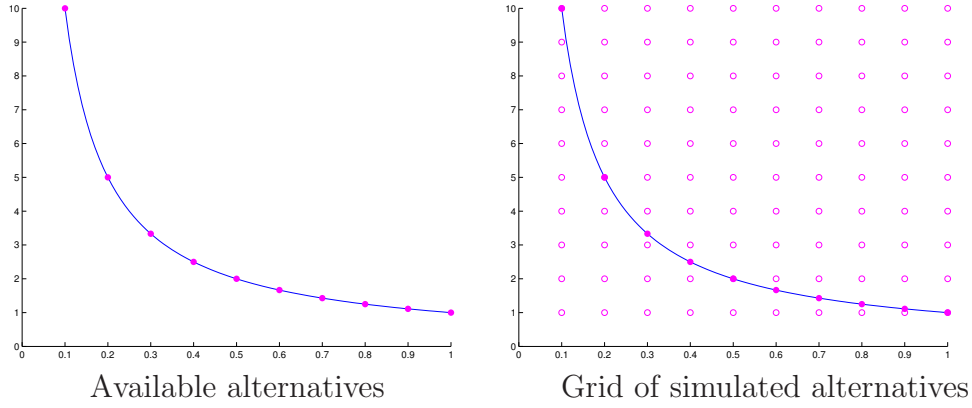


Figure 4.4: Refinement of a Pareto-optimal front.

## 4.2.2 Step 2: Ranking Construction

The partial ranking is the developed technique employed in building a partial sorting for the alternatives. The ranking assigns a scalar value to each alternative and provides a way of quantifying the DM's preferences. Let  $P$  be a set with  $n$  alternatives and let  $v_1 = p \in P$  be an alternative chosen randomly<sup>1</sup> in  $\mathcal{A}$ , called alternative **pivot**. Considering the DM's preferences between the pivot  $v_1$  and each other alternative  $p \in P$  two new sets are constructed: the preferable alternatives, denoted  $F_1$ , and the non-preferable alternatives, denoted  $T_1$ . Choosing a pivot  $v_2$  in the set  $F_1$  and repeating the process the sets  $F_2$  and  $T_2$  are constructed. This process is repeated until the set  $F_k$  has only one alternative which corresponds to the preferred alternative by the DM<sup>2</sup>.

<sup>1</sup>All the random procedures in this work generate values according to a uniform distribution.

<sup>2</sup>As stated in Section 3.1, in situations in which the alternatives are equally preferable each alternative has 50% chance to be the preferred one.

During the construction of the sets  $F_i$  and  $T_i$  the level of each alternative  $p \in P$  is defined. Let  $k$  be the final stage of the technique, that is, the stage in which the set  $F_k$  contains only one alternative. For each  $p \in P$ , if  $p \in T_j$  then the level of  $p$  is defined as  $\mathcal{R}(p) = j$ . If  $p \notin T_j$ , for all  $j = 1, \dots, k - 1$ , then  $p \in F_k$  and the level of  $p$  is defined as  $\mathcal{R}(p) = k$ . Thus the level of each alternative  $p \in P$  is defined by Equation 4.1.

$$\mathcal{R}(p) = \begin{cases} j, & \text{if } a \in T_j, \text{ for some } j, \\ k, & \text{if } a \notin T_j, \text{ for all } j. \end{cases} \quad (4.1)$$

The ranking technique enables to quantify the DM's preferences in any set of alternatives within the domain by assigning a scalar value to each alternative  $p \in P$  and constructing the function  $\mathcal{R} : P \rightarrow \mathbb{R}$ . The function  $\mathcal{R}$  provides the data for training a regression technique extending the function  $\mathcal{R}$  to a function  $\hat{\mathcal{U}} : \mathcal{D} \rightarrow \mathbb{R}$  which represents the DM's preferences in the entire domain  $\mathcal{D}$ .

The ranking-based classification offers a quantitative (cardinal) way of comparing the alternatives which is a kind of information which is not provided directly by the DM. In the partial ranking an alternative which is assigned a level  $i + 1$  is necessarily better than an alternative with a level  $i$  although two alternatives with the same level  $i$  may be not equivalent under the utility function  $\mathcal{U}$ . The information acquired by the ranking is obtained through a reasonable amount of information from the DM and it is enough to allow the regression technique to construct an appropriate model for the DM's utility function  $\mathcal{U}$ .

### 4.2.3 Step 3: Artificial Neural Network Approximation

The regression technique chosen to approximate the underlying utility function is an Artificial Neural Network (ANN). An ANN is an information processing paradigm which is inspired by the way the information processing mechanisms of biological nervous systems, such as the brain, process the information. The key element of this paradigm is the structure of the information processing system which is composed of a large number of highly interconnected processing elements (neurons) working together to solve specific problems. ANNs often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape.

An ANN learns by examples and the aim of this learning is the attainment of models with good generalization capacity associated to the capacity to learn from a reduced set of examples and to supply coherent answers to unknown data. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons; in an ANN the synaptic connections are represented by its weights.

The Radial Basis Function (RBF) network (Figure 4.5) is the type of ANN employed in this work. The main features of RBF networks are:

- they are two-layer feed-forward networks;
- they are very good at interpolation;
- the hidden layer implements a set of radial basis functions (e.g. Gaussian functions);

- the output layer implements linear summation functions;
- the network training is divided into two stages: first the parameters of the hidden layer are determined, and then the weights from the hidden to output layer are obtained;
- both training and learning are very fast.

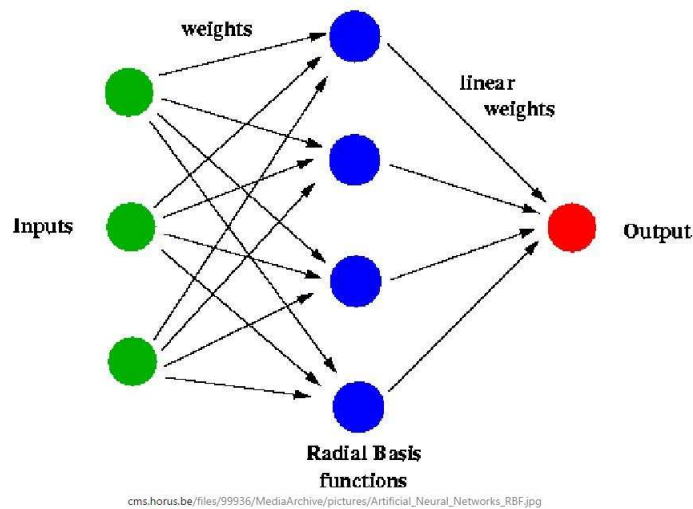


Figure 4.5: RBF network architecture.

Formally, a RBF network is a real-valued function whose value depends only on the distance from  $\mathbf{x}$  to a point  $\mathbf{x}_i$ , called center, so that  $\phi(\mathbf{x}, \mathbf{x}_i) = \phi(\|\mathbf{x} - \mathbf{x}_i\|)$ <sup>3</sup>. Sums of radial basis functions are typically considered in building up function approximations of the form given by Equation 4.2, in which the approximating function  $y(\mathbf{x})$  is represented as a sum of  $N$  radial basis functions, each one associated with a different center  $\mathbf{x}_i$ , and weighted by an appropriate coefficient  $w_i$ . It can be shown that any continuous function on a

<sup>3</sup>The adopted norm is the Euclidean norm.

compact interval can in principle be interpolated with arbitrary accuracy by a sum of this form if a sufficiently large number  $N$  of radial basis functions is considered.

$$y(\mathbf{x}) = \sum_{i=1}^N w_i \cdot \phi(\|\mathbf{x} - \mathbf{x}_i\|) \quad (4.2)$$

In this work a common type of radial basis function is chosen: a Gaussian given by Equation 4.3,

$$\phi(r) = \exp\left(-\frac{r^2}{\sigma^2}\right), \quad (4.3)$$

in which  $r = \|\mathbf{x} - \mathbf{x}_i\|$  and  $\sigma$  is a parameter related to the spread of the function. Figure 4.6 presents an example of unidimensional Gaussian functions with  $\mathbf{x}_i = 0$  and different  $\sigma$  values.

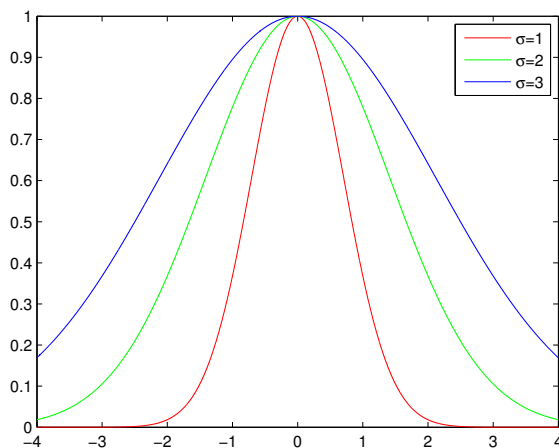


Figure 4.6: Gaussian functions with  $\sigma = 1, 2, 3$ .

The RBF network possess three parameters: (i) the centers of the RBF functions ( $\mathbf{x}_i$ ); (ii) the spread of the Gaussian RBF functions ( $\sigma$ ), and (iii) the weights from the hidden to the output layer ( $w_i$ ). Several methods have

been proposed for training RBF networks. Generally, the training is divided into two stages. In the first stage (Steps 1 and 2) the number of radial basis functions and their parameters are determined based on unsupervised methods. In the second stage (Step 3) the adjustment of the weights from the hidden to the output layer is performed. Essentially this stage consists in finding the weights that optimize a single layer linear network.

The following techniques have been applied to train the RBF networks in each specified step.

**Step 1** Finding the centers of the radial basis functions

- Initial methods, in which each data sample is assigned to a basis function [Specht, 1990].
- Fixed centers selected at random [Broomhead and Lowe, 1988].
- K-means algorithm [Macqueen, 1967; Moody and Darken, 1989].
- Adaptive k-means algorithm (self-organizing map) [Kohonen, 1989].
- Subset selection [Berk, 1978; Chen *et al.*, 1991]:
  - forward selection: starts with an empty subset; added one basis function at a time (the one that most reduces the sum-squared-error); until some chosen criterion stops;
  - backward elimination: starts with the full subset; removed one basis function at a time (the one that least increases the sum-squared-error); until some chosen criterion stops.

**Step 2** Finding the spread of the radial basis function

- Each value of  $\sigma$  is defined as the average of the Euclidean distances between the center of each sample and the center of the nearest sample [Moody and Darken, 1989].
- $P$ -nearest neighbor algorithm (Equation 4.4): a number  $P$  is chosen; for each center, the  $P$  nearest centers are found; the root-mean squared distance between the current cluster center and its  $P$  nearest neighbors is calculated [Knuth, 1998].

$$\sigma_j = \sqrt{\frac{1}{P} \sum_{i=1}^P (\mathbf{x}_j - \mathbf{x}_i)^2} \quad (4.4)$$

**Step 3** Finding the weights from the hidden to the output layer

- Singular value decomposition (SVD) [Lay, 2002].
- Least Mean Squares (LMS) algorithm [Widrow and Hoff, 1988].

For the construction of the RBF network the MATLAB<sup>®</sup> NEWRB function is employed with the parameters given by Table 4.1. The NEWRB function iteratively creates a radial basis network one neuron at a time. Neurons are added to the network until the sum-squared error falls beneath an error goal or a maximum number of neurons is reached. This function is flexible enough to provide a suitable approximation of the DM's preferences in each scenario considered in this work. For additional information about the NEWRB function, check the Appendix A.

Name	Value	Name	Value
P	Alternatives in the domain	SPREAD	500
T	Ranking of the alternatives	MN	200
GOAL	0.1	DF	25

Table 4.1: MATLAB<sup>®</sup> parameters of the NEWRB function.

The RBF network  $\hat{\mathcal{U}}$  which approximates the utility function  $\mathcal{U}$  has the set of simulated alternatives  $\mathcal{F}$  as its input and the ranking level  $\mathcal{R}$  of each alternative as its output. The major advantage of choosing an artificial neural network is that the proposed methodology does not require a specific form from the underlying utility function  $\mathcal{U}$ . Figure 4.7 presents an example of a Gaussian underlying utility function and a model obtained by the NN-DM method. Once the function  $\hat{\mathcal{U}}$  is estimated, it can be employed in quantifying any alternative within its domain, as shown in Figure 4.8.



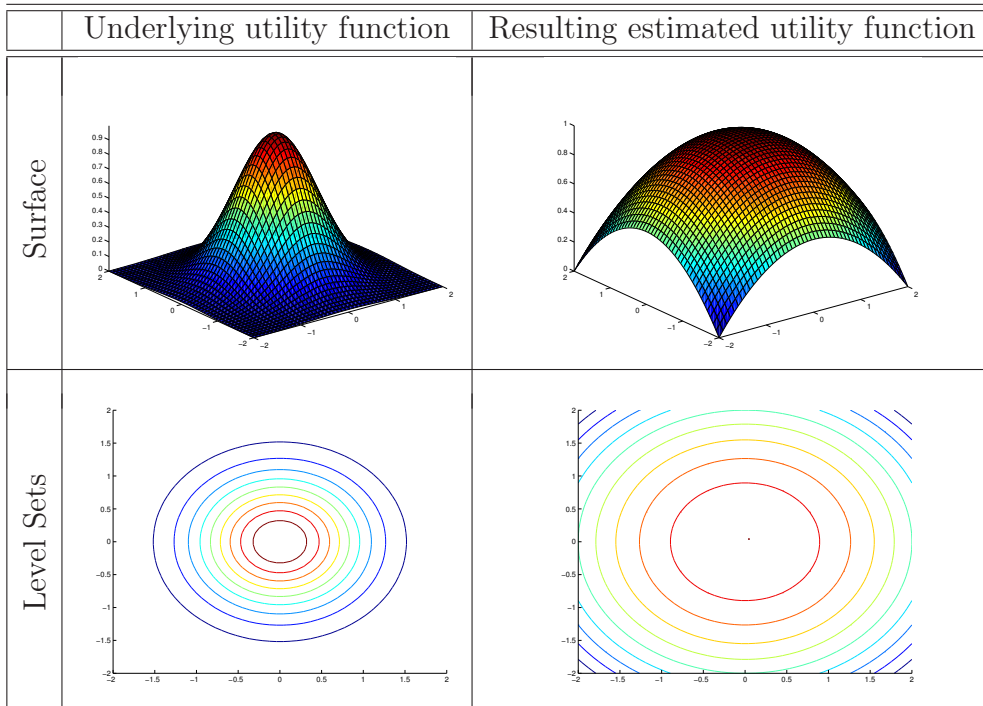


Figure 4.7: Surface and level sets of the functions  $\mathcal{U}$  and  $\hat{\mathcal{U}}$ .

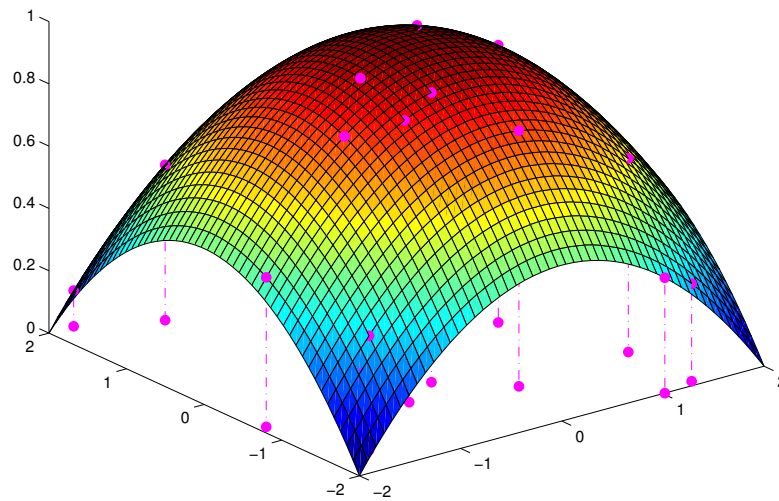


Figure 4.8: Example of a function  $\hat{\mathcal{U}}$  being employed.

It is not necessary to model  $\mathcal{U}$  exactly because when the ranking is employed in building the approximation the resulting function  $\hat{\mathcal{U}}$  has level sets which are similar to the ones of  $\mathcal{U}$  and possesses information enough to codify the DM's preferences. Therefore, the final surface is normalized by scaling the RBF output between 0 and 1. It is worth mentioning that any other interpolation method could have been chosen and the choice of the RBF network is due to its easy implementation and reduced computational load since the weights are updated linearly.

#### 4.2.4 Step 4: Performance Assessment

Now that a model  $\hat{\mathcal{U}}$  for the DM's utility function  $\mathcal{U}$  is available a value for each alternative can be inferred and a sorting of all alternatives can be constructed. This section presents a metric to assess the performance of the model  $\hat{\mathcal{U}}$  related to the DM's preferences.

The Kendall-tau Distance (KTD) [Kendall, 1938] is a metric that counts the number of pairwise disagreements between two ranking lists. The KTD for a set  $A$  sorted according the rankings  $\tau_1$  and  $\tau_2$  is given by Equation 4.5.

$$K(\tau_1, \tau_2) = |\{(i, j) : i < j, (\tau_1(i) < \tau_1(j) \wedge \tau_2(i) > \tau_2(j)) \vee (\tau_1(i) > \tau_1(j) \wedge \tau_2(i) < \tau_2(j))\}|. \quad (4.5)$$

Considering  $n$  the list size, the normalized Kendall-tau distance, obtained by dividing the KTD value by  $n(n-1)/2$  (total number of pairs), lies in the interval  $[0, 1]$ . The normalized Kendall-tau distance, here simply represented by KTD, is employed in this work as the merit function.

As an example, consider the sorting by height and weight of a group of five people is required and consider the sorting provided by Table 4.2.

Person	A	B	C	D	E
Sorting by height	1	2	3	4	5
Sorting by weight	3	4	1	2	5

Table 4.2: Example of the KTD metric.

The calculus of the Kendall-tau distance is made by comparing each pair of people and counting the number of discordant pairs, that is, the number of times where the values in the list  $L_1$  (height) are in the opposite order in the list  $L_2$  (weight).

Pair	Height	Weight	Discordant Pair
$(A, B)$	$1 < 2$	$3 < 4$	
$(A, C)$	$1 < 3$	$3 > 1$	$X$
$(A, D)$	$1 < 4$	$3 > 2$	$X$
$(A, E)$	$1 < 5$	$3 < 5$	
$(B, C)$	$2 < 3$	$4 > 1$	$X$
$(B, D)$	$2 < 4$	$4 > 2$	$X$
$(B, E)$	$2 < 5$	$4 < 5$	
$(C, D)$	$3 < 4$	$1 < 2$	
$(C, E)$	$3 < 5$	$1 < 5$	
$(D, E)$	$4 < 5$	$2 < 5$	

Table 4.3: Concordant and discordant pairs.

As there are four pairs whose values are in the opposite order, the KTD value is 4. Normalizing this value, the resulting KTD, denoted  $\tau$ , is  $\tau = 0.4$ . The value  $\tau = 0.4$  indicates that there is a small similarity between the lists considered.

The KTD evaluates the proximity between the sorting for the alternatives provided by the approximation  $\hat{\mathcal{U}}$  and the ideal sorting provided by the utility function  $\mathcal{U}$ . In the study conducted here the KTD is a suitable metric because

an absolute reference (the utility function  $\mathcal{U}$ ) is assumed to be available in the tests that have been performed. The KTD is a measure of the closeness between the sorting given by the resulting model and the optimal sorting provided by the DM.

In an attempt to assess the model's performance, in each algorithm  $nvs = 30$  validation sets with  $nvp = 50$  randomly distributed alternatives are created in the domain  $\mathcal{D}$ . The validation sets are constructed to evaluate the performance of the  $\hat{\mathcal{U}}$  model within the entire domain  $\mathcal{D}$ . A sorting for each validation set is obtained by the functions  $\mathcal{U}$  and  $\hat{\mathcal{U}}$  and the resulting KTD, also denoted  $\tau$ , is the average of the obtained values of each validation set. A model is then said stable if  $\tau$  satisfies a predefined tolerance  $tol_{st}$ .

#### 4.2.5 DM calls

This section presents an estimate of the number of queries required from the DM in the ranking procedure developed in Section 4.2.2. Let  $n$  be the number of alternatives in the set  $G$  and assume  $n = 2^k$ , for some  $k \in \mathbb{N}$ . In each step the pivot has to be compared with each alternative and, on average, the pivot splits the set in two sets with same size. That means in the first step the pivot is compared with  $n - 1$  alternatives, in the second step with  $n/2 - 1$  alternatives, and the process goes on until only one alternative has left. As the expected number of levels is  $k = \log n$ ,<sup>4</sup> the total number of queries, denoted  $T(n)$ , is given by Equation 4.6.

---

<sup>4</sup>The notation  $\log x$  denotes the same as  $\log_2 x$  in the whole text.

$$\begin{aligned}
T(n) &= (n-1) + \left(\frac{n}{2} - 1\right) + \left(\frac{n}{4} - 1\right) + \dots + \left(\frac{n}{n} - 1\right) \\
&= 2^k + \frac{2^k}{2} + \frac{2^k}{4} + \dots + \frac{2^k}{2^k} - k \\
&= 2^k \cdot \left(1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{2^k}\right) - k \\
&= 2^k \cdot \left(\frac{2 \cdot 2^k - 1}{2^k}\right) - k \\
&= 2 \cdot 2^k - 1 - k \\
&= 2n - \log n - 1.
\end{aligned} \tag{4.6}$$

Taking now an arbitrary number of alternatives the number of queries to the DM is approximately given by  $T(n) = 2n$ . This result represents a linear behavior between those variables. However, it is important to point out that the number of queries in the domain is connected to the number of criteria of the decision-making problem.

### 4.3 The Algorithm

Consider a decision-making problem with dimension  $d$ , domain  $\mathcal{D}$ , and underlying utility function  $\mathcal{U}$ . Let  $\mathcal{A} \subset \mathcal{D}$  be a set of available alternatives and  $G \subset \mathcal{D}$  be a grid of simulated alternatives. Define the function  $\mathcal{R} : G \rightarrow \mathbb{R}$  which attributes a real number to each alternative  $\mathbf{p} \in G$  acquired by the stages of a ranking procedure. By employing a RBF network the function  $\mathcal{R}$  can be extended to a function  $\hat{\mathcal{U}} : \mathcal{D} \rightarrow \mathbb{R}$ , which reproduces the DM's preferences in the entire domain  $\mathcal{D}$ . Algorithm 1 presents the NN-DM method.

---

**Algorithm 1** NN-DM Method

---

- 1: Obtain the domain  $\mathcal{D}$
  - 2: Construct the grid of simulated alternatives  $G$
  - 3: **while**  $\#F_i > 1$  **do**
  - 4:     Select a pivot  $v_i$  belonging to the set  $F_i$
  - 5:     Obtain the sets  $F_{i+1}$  and  $T_{i+1}$
  - 6: **end while**
  - 7: Assign a rank  $\mathcal{R}(\mathbf{p})$  to each alternative  $\mathbf{p} \in G$
  - 8: Adjust the RBF network  $\hat{\mathcal{U}}$
  - 9: Assess the performance under a tolerance  $tol_{st}$
- 

## 4.4 Illustrative Examples

This section presents illustrative examples of the NN-DM method. Figure 4.9 shows three DM's underlying utility functions simulated by the functions given by Table 4.4. The functions  $\mathcal{U}_1$  and  $\mathcal{U}_3$  possess only one preferred alternative and the function  $\mathcal{U}_2$  is an example of a DM with several preferred solutions. In each instance  $\mathcal{D} = [0, 1] \times [0, 1]$  and  $\mathbf{p} = (p_1, p_2)$  represents an alternative.

Utility Function	Expression
$\mathcal{U}_1(\mathbf{p})$	$\exp(-p(1) \cdot p(1)) \cdot \exp(-p(2) \cdot p(2))$
$\mathcal{U}_2(\mathbf{p})$	$\sin(p(1) + p(2))$
$\mathcal{U}_3(\mathbf{p})$	$p(1) \cdot p(2)$

Table 4.4: DM's underlying utility functions.

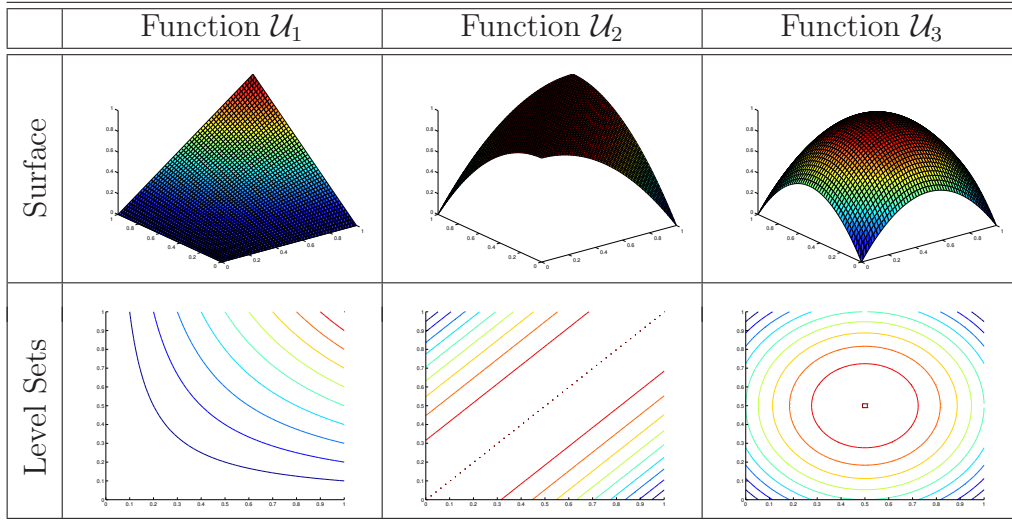


Figure 4.9: DM's underlying utility functions.

Figure 4.10 shows the partial ranking obtained by Step 2 of the NN-DM method considering a grid with  $n_1 = n_2 = 20$  alternatives. The average number of queries to the DM is  $2 \cdot 20^2$  according to Equation 4.6.

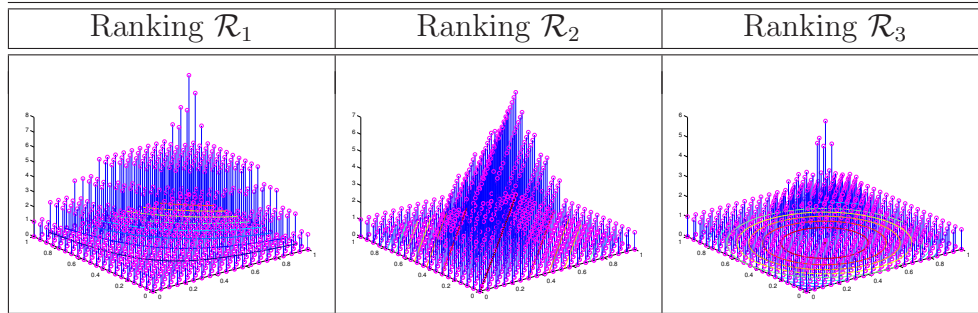


Figure 4.10: Partial ranking.

Figure 4.11 presents the RBF network  $\hat{\mathcal{U}}$  which represents the DM's preferences.

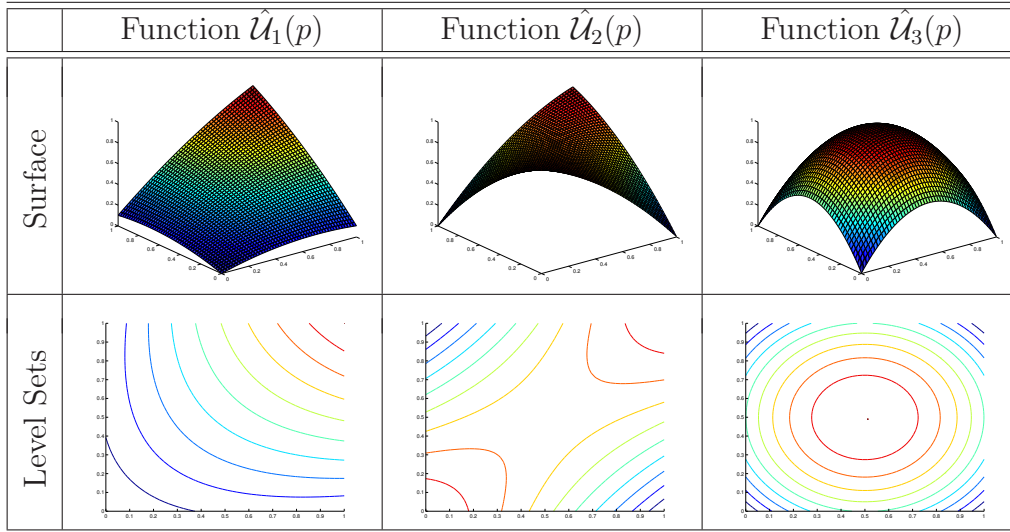


Figure 4.11: Models  $\hat{\mathcal{U}}$  obtained by the NN-DM method.

The tests show that the NN-DM method is able to construct models that represents the DM's preferences. Assuming the same utility function  $\mathcal{U}$  in problem instances of the same decision-making problem the resulting model  $\hat{\mathcal{U}}$  can be employed in avoiding the formulation of new queries to the DM.

## 4.5 Discussion

The NN-DM method is efficient at constructing an approximation  $\hat{\mathcal{U}}$  for the DM's utility function  $\mathcal{U}$  but the partial ranking is not stable, since the whole process depends on the choice of the pivot alternative. In a poor scenario, the pivot can be the worst alternative in all stages and the partial ranking becomes a total ranking, demanding excessive information from the DM. In an even worse scenario, if the first pivot is the best alternative according to the DM's preferences only two levels are constructed and the



information would not be enough to find an appropriate model for the DM's utility function.

Figure 4.12 introduces the same underlying utility functions presented in Figure 4.9 and examples of the partial ranking obtained by the NN-DM method considering the same grid with  $n_1 = n_2 = 20$  alternatives.

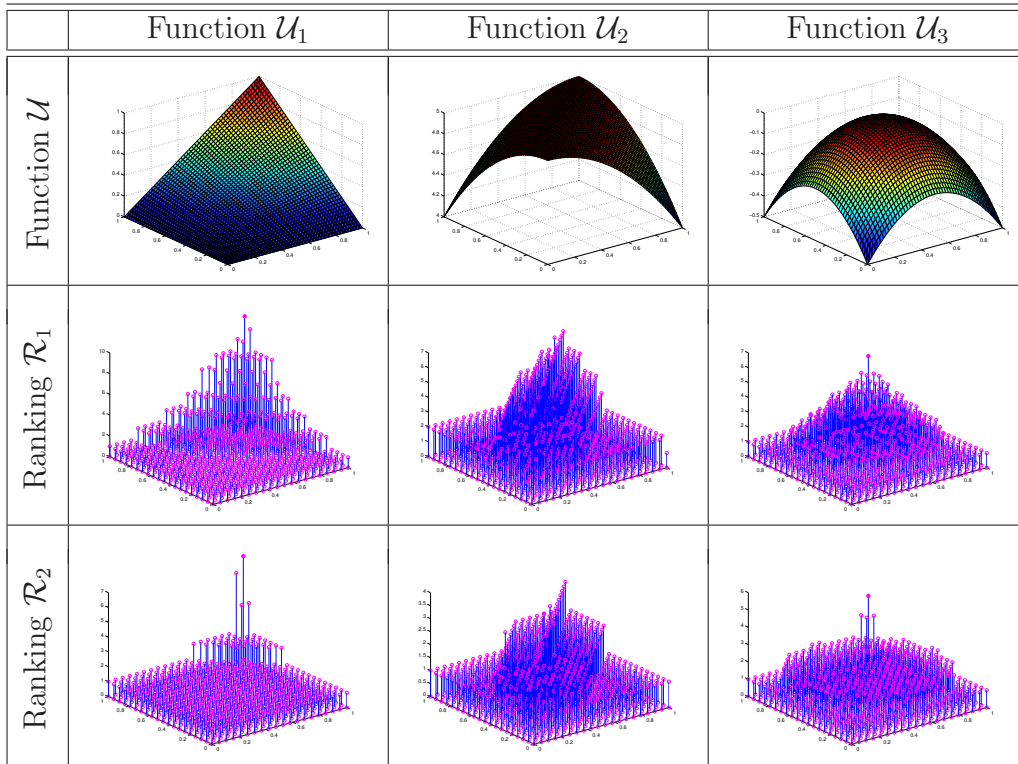


Figure 4.12: Partial ranking examples.

Figure 4.12 shows that, due to the elimination process of alternatives in the ranking construction, in certain runs the main characteristic of the utility function is lost in early stages of the technique. The main reason for this procedure not represent properly the underlying utility functions is the high number of alternatives with small values. The number of alternatives in each

level is approximately half of the number of alternatives in the previous level which clusters the available information in lower levels quickly.

Another issue present in the NN-DM method is the grid of alternatives. In situations in which the problem dimension is high the amount of information required from the DM to construct an appropriate model may not be available. Furthermore, if the decision-making process is interactive, as in successive approximations of the PF, it is not possible to insert new alternatives in the domain without resizing the whole domain.

Finally, no kind of dominance is taken into account here. In the class of problems considered in this work the dominance among the alternatives can be considered in replacing certain queries and reducing the requirements from the DM.

# Chapter 5

## The Improved NN-DM Method

### 5.1 Introduction

The purpose of the NN-DM method is to find an approximation  $\hat{\mathcal{U}}$  of the utility function  $\mathcal{U}$ . Once the domain  $\mathcal{D}$  is established the DM should answer queries about her/his preferences over  $\mathcal{D}$ , conducting to a ranking model for the preferences and, therefore, the construction of the function  $\hat{\mathcal{U}}$  which approximates the utility function  $\mathcal{U}$ .

This chapter presents improvements introduced in the original NN-DM method. The improved methodology can be described in the same four main steps as the original NN-DM method, but optimizations are developed in the Steps 1 and 2: the domain  $\mathcal{D}$  is now composed by random simulated alternatives, the dominance is considered in taking the decision replacing the DM in certain queries, and the partial ranking is built over a totally sorted subset of the simulated alternatives in an attempt to create a stable ranking.

## 5.2 Step 1 - Domain Establishment

In its original description the NN-DM method constructs a grid of simulated alternatives in the domain  $\mathcal{A}$  which provides a uniform representation of the DM's utility function  $\mathcal{U}$ . Unfortunately, by considering higher dimensions the grid becomes ineffective, demanding several queries to the DM. In a real scenario the DM is a person which has restrictions related to her/his time leading to restrictions in the number of queries to be answered.

In the improved NN-DM method a simulated decision-making problem in which the alternatives are randomly located is built in the domain  $\mathcal{D}$ . The queries to the DM are now presented over these simulated alternatives. By considering random alternatives instead of the grid of alternatives the model's accuracy still varies according to the number of alternatives but now new alternatives can be added gradually until the method is able to obtain an appropriate representation of the DM's preferences.

Figure 5.1 presents an example of the domain establishment considering non-dominated alternatives from a multi-objective optimization problem. The available alternatives are not sufficient to represent the entire domain of the DM's preferences, demanding the insertion of simulated alternatives. Note that the available alternatives are not considered in constructing the model.

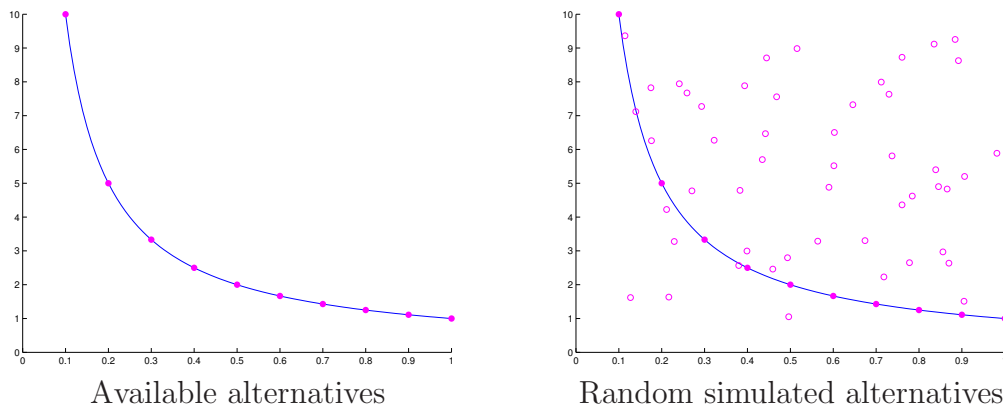


Figure 5.1: Domain establishment.

## 5.3 Step 2 - Ranking Construction

### 5.3.1 Dominance

In the class of problems considered here the dominance between the alternatives can be considered in replacing certain queries to the DM reducing the demand in the process of preference extraction. Given two alternatives  $a_i$  and  $a_j$ , three situations are possible:

- if  $a_i$  dominates  $a_j$ , then  $a_i$  must be the preferred alternative;
- if  $a_j$  dominates  $a_i$ , then  $a_j$  must be the preferred alternative;
- if neither  $a_i$  nor  $a_j$  dominates each other the DM is then consulted to find the preferred alternative.

In situations in which the decision-making problem comes from a multi-objective optimization problem usually the available alternatives belong to

a set with non-dominated solutions. The extra cost of considering simulated alternatives within the entire domain the DM's requirements can now be reduced by taking into account the dominance. Henceforth, only the information about non-dominated alternatives is required from the DM making the whole process cost-effective.

### 5.3.2 The Improved Partial Ranking

The partial ranking is a technique employed in finding a partial sorting for the alternatives leading to the assignment of a scalar value to each alternative. In an attempt to produce a stable partial ranking a modification in the way the pivots are built is introduced in the improved NN-DM method. Considering a set  $\mathcal{A}$  with  $n$  alternatives the improved partial ranking is performed through the following steps.

1. Choose randomly  $k = \log n$  alternatives from the set  $\mathcal{F}$ ; these alternatives are called **pivots**. The  $k$  value is inspired on the expected number of levels of the original procedure.
2. Sort the pivots in ascending order of the DM's preferences considering the ordinal information obtained from yes/no queries. A rank is assigned to each pivot corresponding to its position in this sorted list. In this step the number of queries that the DM has to answer is equal to the number of comparisons that a sorting algorithm performs. Therefore, considering an algorithm as Mergesort, the average number of queries to obtain a total sorting is  $k \cdot \log k$ .
3. For each one of the  $n - k$  remaining alternatives, assign a rank that

is the same one of the pivot immediately better than the alternative, in the DM's preferences. If the current alternative is better than the pivot with rank  $k$  it receives rank  $k + 1$  and the number of pivots is increased. Each remaining alternative is compared with the middle pivot and, based on the result, compared with the middle pivot of the higher or lower sub-partition. This process continues until a rank is assigned for each remaining alternative.

The procedure developed here creates a partition of the set  $\mathcal{F}$  into at least  $k = \log n$  disjunct subsets representing the ranking levels which is the average number of levels of the original methodology. The main difference between these procedures is that the improved partial ranking always has at least  $k$  levels which provides a more balanced number of alternatives in each level and generates suitable information to construct the model for the DM's preferences.

## 5.4 DM Calls

In the improved NN-DM method the interaction with the DM is necessary in two occasions: the pivot total sorting and the position selection of each remaining alternative.

### Pivot Total Sorting

The number of queries the DM has to answer is equal to the number of comparisons that a sorting algorithm must execute. A method such as Mergesort is known to perform, on average,  $k \cdot \log k$  comparisons

between the alternatives to sort them, therefore this value represents a good estimate of the number of queries presented to the DM for the total sorting of the pivots.

### Position Selection Of Each Remaining Alternative

For selecting the position of each remaining alternative this alternative must be compared with the pivot alternatives. A binary search procedure requires on average  $\log k$  queries by each alternative. So, as there are  $n - k$  remaining alternatives, then  $(n - k) \cdot \log k$  queries are made, on average, during this procedure.

Therefore, the average total of queries  $T(n)$  to the DM is given by Equation 5.1.

$$\begin{aligned}
 T(n) &= k \cdot \log k + (n - k) \cdot \log k \\
 &= k \cdot \log k + n \cdot \log k - k \cdot \log k && (5.1) \\
 &= n \cdot \log k \\
 &= n \cdot \log(\log n).
 \end{aligned}$$

## 5.5 The Algorithm

Algorithm 2 presents the improved NN-DM method.



---

**Algorithm 2** Improved NN-DM Method

---

- 1: Obtain the domain  $\mathcal{D}$
  - 2: Construct the set  $\mathcal{F}$  with  $n$  randomly simulated alternatives
  - 3: Select the  $k = \log n$  pivots
  - 4: Sort the pivots in ascending order
  - 5: Assign a rank  $\mathcal{R}(v_k)$  to each pivot  $v_k$
  - 6: Assign a rank  $\mathcal{R}(a_i)$  to each  $n - k$  remaining alternative  $a_i$
  - 7: Adjust the RBF network  $\hat{\mathcal{U}}$
  - 8: Assess the performance
- 

## 5.6 Illustrative Examples

This section presents two illustrative examples of the improved NN-DM method. In the first example the DM's underlying utility function is simulated by a bimodal Gaussian, indicating that the proposed methodology is effective in relation to multi-modal functions, although with certain restrictions. In the second example the DM's underlying utility function allows the dominance to replace the DM and a comparison between the queries effectively answered by the DM and by the dominance is realized.

### 5.6.1 Example A

The first example considers the DM's underlying utility function  $\mathcal{U}$  simulated by a bimodal Gaussian given by Equation 5.2, in which  $\mathbf{p} = (p_1, p_2)$  is an alternative. The function  $\mathcal{U}$  represents a DM with two distinct preferable regions (two local maximums), although one region is most preferable than the other one (global maximum). Figure 5.2 presents the surface and level sets of the function  $\mathcal{U}$ .

$$\mathcal{U}(p) = 25 \cdot \exp(-(p_1 - 3)^2 - p_2^2) + \exp(-p_1^2 - (p_2 - 3)^2) \quad (5.2)$$

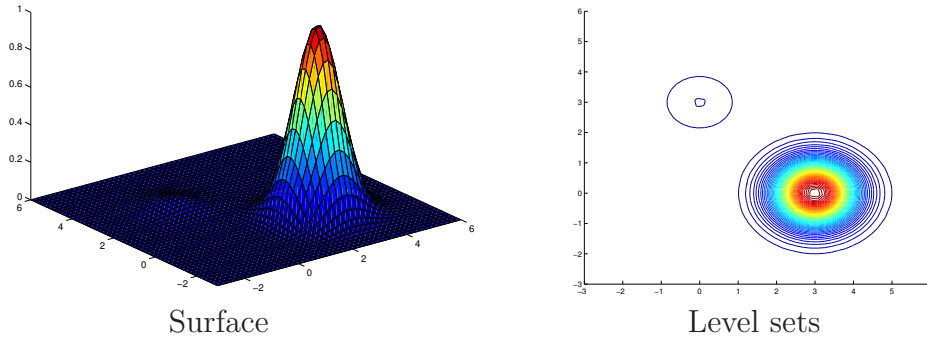


Figure 5.2: DM's underlying utility function  $\mathcal{U}$ .

In the domain  $\mathcal{D} = [-2, 5] \times [-2, 5]$  a simulated decision-making problem with  $n = 50$  alternatives is constructed. Figure 5.3 shows the partial ranking obtained by Step 2. Note that the difference between the local maximums are smoothed by the ranking procedure. This behavior is expected since the pivots take place in an integer scale and, by considering this specific function  $\mathcal{U}$ , the majority of the pivots would have values similar to zero leading the local maximums with similar ranking.

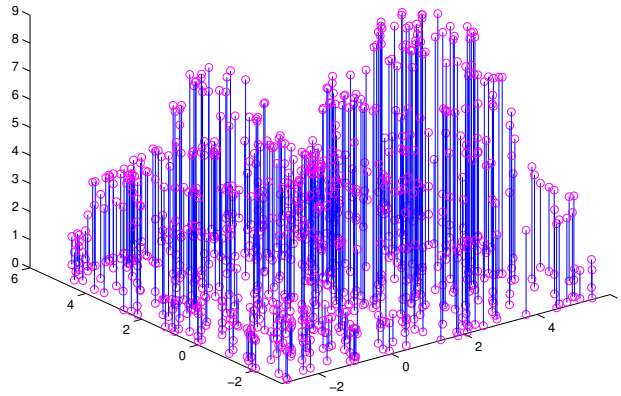


Figure 5.3: Partial ranking with  $n = 50$  alternatives.

Figure 5.4 presents the RBF network  $\hat{\mathcal{U}}$  which represents the DM's preferences for the given function  $\mathcal{U}$ . In this example the MATLAB<sup>®</sup> parameters are giving by Table 5.1.

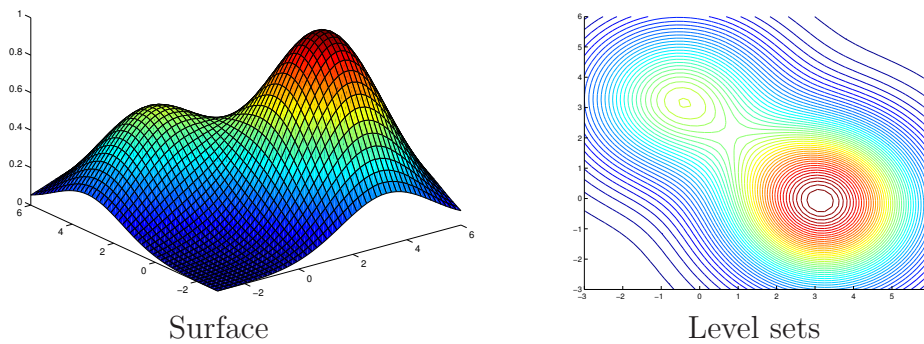


Figure 5.4: Model  $\hat{\mathcal{U}}$  for the DM's preferences.

Name	Value	Name	Value
P	Alternatives in the domain	SPREAD	5
T	Ranking of the alternatives	MN	20
GOAL	0.5	DF	2

Table 5.1: MATLAB<sup>®</sup> parameters: the NEWRB function.

### 5.6.2 Example B

Now the DM's underlying utility function  $\mathcal{U}$  is simulated considering the function given by Equation 5.3, with matrices  $M$  given by  $I_2$  and  $I_3$  (identity matrices of size 2 and 3, respectively). In this example the dominance can be considered in replacing the DM in queries in which one alternative dominates the other one.

$$\mathcal{U}(p) = \exp(-\mathbf{p} \cdot M \cdot \mathbf{p}^t) \quad (5.3)$$

In the two-dimensional instance a simulated decision-making problem with  $n = 50$  alternatives is constructed in the domain  $\mathcal{D} = [0, 1] \times [0, 1]$ . Figure 5.5 presents the underlying utility function  $\mathcal{U}$ .

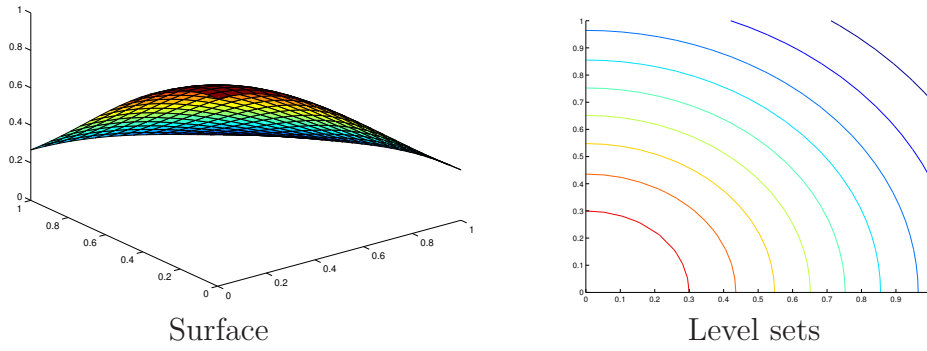


Figure 5.5: DM's underlying utility function  $\mathcal{U}$ .

Figure 5.6 shows the partial ranking obtained by Step 2 and Figure 5.7 presents the RBF network  $\hat{\mathcal{U}}$  which represents the DM's preferences.

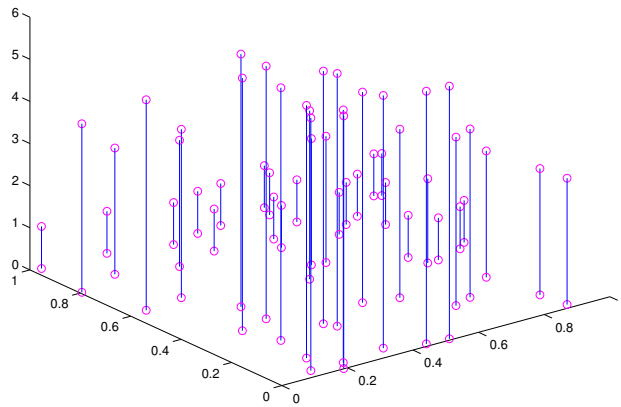


Figure 5.6: Partial ranking with  $n = 50$  alternatives.

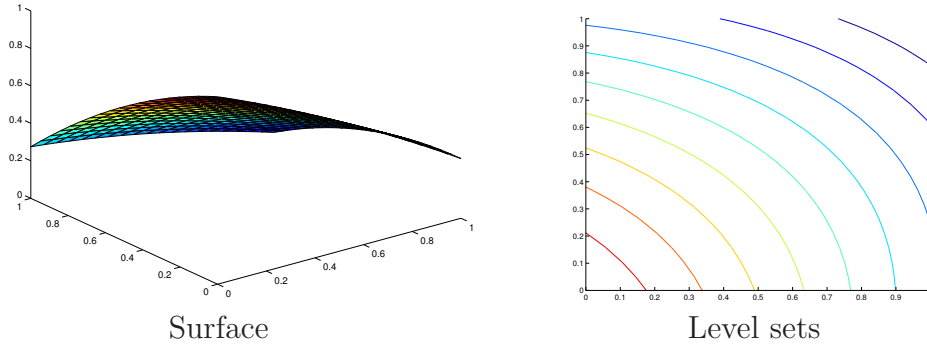


Figure 5.7: Model  $\hat{U}$  for the DM's preferences.

Figure 5.8 presents a comparison among the queries involved in finding an approximation of the DM's preferences and the resulting KTD of each model considering the number of alternatives between 50 and 500, with increments of 50.

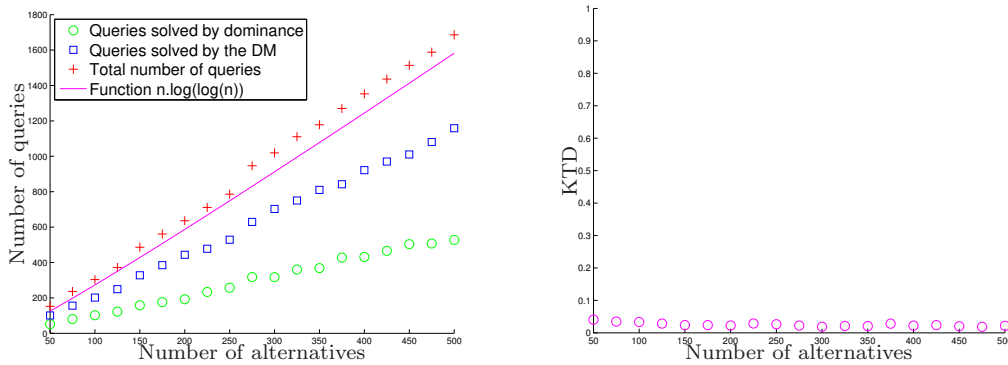


Figure 5.8: Two-dimensional instance: number of queries and KTD.

Figure 5.9 reproduces the same graphics in Figure 5.8 for the three-dimensional instance. The domain now is  $\mathcal{D} = [0, 1] \times [0, 1] \times [0, 1]$ .

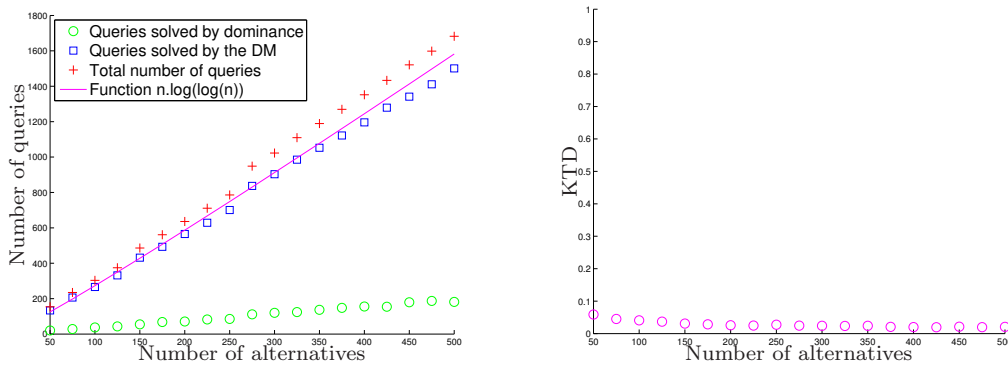


Figure 5.9: Three-dimensional instance: number of queries and KTD.

The tests show that the improved NN-DM method is still able to construct models that represents the DM’s preferences with a reasonable amount of information required from the DM. Figures 5.8 and 5.9 suggest that in higher dimensions the impact of the queries solved by dominance is reduced leaving all the hard work to the DM.

## 5.7 Discussion

Figure 5.10 presents the results of the same underlying utility functions defined by Table 4.4 in Chapter 4. Now the ranking is stable and the RBF network is able to produce an approximation  $\hat{\mathcal{U}}$  of the utility function  $\mathcal{U}$  which reflects the DM’s preferences. The RBF parameters are the same provided by Table 4.1.

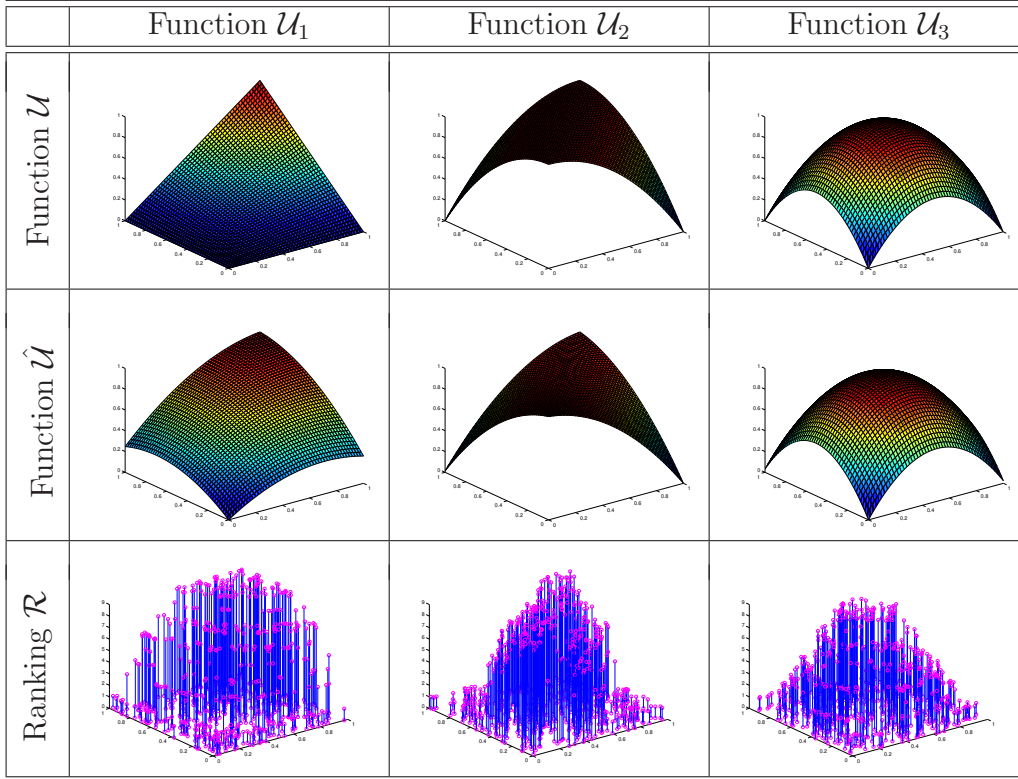


Figure 5.10: Partial ranking examples.

The improved NN-DM method is an effective way to construct a model which represents the DM's preferences. Assuming  $n > 16$ , the demand required from the DM ( $T(n) = n \cdot \log(\log n)$ ) is higher than the original NN-DM method ( $T(n) = 2n$ ). However, the stability acquired by the improvements justify the extra costs. Furthermore, the introduction of the dominance reduces the number of queries required from the DM. From now on only the improved NN-DM method is considered in this work and the references will be made suppressing the term "improved".



# Chapter 6

## The NN-DM Method And iTDEA

### 6.1 Introduction

#### 6.1.1 Interactive Algorithms

The development of multi-objective approaches for the design of an increasing number of real-world systems is a current trend. Although there are available, at this moment, several Evolutionary Multi-objective Optimization (EMO) techniques that aim to provide representative samplings of the Pareto-sets in multi-objective optimization problems [[Fonseca and Fleming, 1995](#); [Knowles and Corne, 2000](#); [Deb \*et al.\*, 2002](#); [Zitzler \*et al.\*, 2002](#)], their application to the actual design of real systems still requires a further step in which, given a set of possible solution alternatives, a specific alternative should be chosen to be implemented. This step is usually recognized as a task that is attributed to the DM. Although the ultimate target in real-world applications is to come up with a single solution, the interactive procedures

can be applied with a decision-making strategy to find the best solution or a set of preferred solutions in regions of interest to the DM.

Classical interactive multi-objective optimization methods usually demand the DM to suggest a reference direction or reference points or other clues which result in a preferred set of solutions on the PF. The following publications present examples of this type of demand.

The reference-point-based NSGA-II (R-NSGA-II) [Deb and Sundar, 2006] put together one preference-based strategy with an EMO methodology in a procedure in which the DM supplies one or more reference points. An iteration of the algorithm demonstrates how a preferred set of solutions near the reference points can be found. The appointed argument is that with a number of trade-off solutions in the region of interest the DM would be able to make a better and more reliable decision than by receiving just a single solution. The obtained solutions range is controlled by a parameter  $\epsilon$  that also controls the PF extent. Since the complete PF is not the target of the approach, some non-PF solutions can be found by the current procedure.

In another attempt considering reference points Köksalan and Karahan [2010] developed the Interactive Territory Defining Evolutionary Algorithm (iTDEA). The iTDEA method creates a territory around each current solution where no other solutions are allowed and defines smaller territories around the preferred solutions producing denser coverage of these regions. At each interaction, the algorithm asks the DM to choose her/his best solution among a set of representative solutions to guide the search toward the selected solution neighborhood. The territory idea has been shown to work well in converging to the PF as well as focusing on the desired parts of the fron-

tier. The iTDEA method is better explored in Chapter 6 and its results are reference to the INSPM algorithm in Chapter 7.

It is well-known that the process of optimizing two or more conflicting objectives usually leads to a set of solutions, the PF solutions, which cannot be ordered by simple comparison of their objective function values. These incomparable solutions, called non-dominated solutions, are outcomes of multi-objective optimization algorithms. The first canonical algorithms for multi-objective optimization problems intend to deliver a detailed uniform sampling of the PF [Fonseca and Fleming, 1995; Zitzler and Thiele, 1999; Deb *et al.*, 2002]. Once this sampling is available it is assumed, for instance, that a DM would compare those solutions, indicating the preferred one as the final solution of the problem.

In recent years, a new approach with particular emphasis on the problems with more than three objectives started to receive a growing attention. Due to the high cardinality of a detailed sampling of the entire PF, some works have proposed procedures that concentrate the sampling in some regions of the PF. These regions are defined on the basis of information obtained from interactions between the optimization algorithm and the DM.

Among the algorithms which consider the DM interacting with the optimization process the work by Köksalan and Karahan [2010] receives a special mention here. That work proposed the Interactive Territory Defining Evolutionary Algorithm (iTDEA). iTDEA is a preference-based multi-objective evolutionary algorithm which identifies the preferable region interacting with the DM on predetermined generations. In each interaction with the DM, a new best individual is chosen and a new preferable region is stipulated, with a

smaller territory for each individual in that region. Individuals falling in that region are assigned smaller territories than those located elsewhere leading to a higher sampling density of the preferable regions.

It should be noticed that the information extracted from the DM by iTDEA is useful only inside the scope of the optimization process in which such information is obtained. Whenever the same (or a similar) problem needs to be solved the DM has to answer the queries about the same region again. However, frequently a multi-objective optimization problem might be solved for slightly different conditions, which produces a different PF from one run to the other, with the DM's preferences kept unchanged. For instance, a product may be produced in different instances with different constraints in the resources availability, or with different parameters in the objective functions.

This chapter presents the results of the hybridization of the NN-DM method with iTDEA. Considering the same amount of preference information required by iTDEA the NN-DM method is able to construct a model for the DM's preferences. From this point forward no further queries are required from the DM related to that specific region of the feature space. The NN-DM model can now solve similar decision-making problems coming from optimization problems leading to EPFs in the same region of the space. Once this preference model is adjusted it also can be employed inside the optimization process in guiding the search without demanding additional information from the DM.

## 6.2 TDEA, prTDEA, and iTDEA

The Territory Defining Evolutionary Algorithm (TDEA) was proposed by [Karahan and Köksalan \[2010\]](#). The TDEA is a steady-state elitist evolutionary algorithm developed to approximate the PF in multi-objective optimization problems based on a territory around each individual. Introducing the DM's preferences a priori in TDEA the preference-based TDEA (prTDEA) is an algorithm constructed to obtain a detailed approximation of the desired regions within the entire PF. Improving this idea the authors proposed the Interactive Territory Defining Evolutionary Algorithm (iTDEA) [[Köksalan and Karahan, 2010](#)]. iTDEA is an algorithm that interacts with the DM during the course of the optimization at predetermined generations, finding the best current solution, and guiding the search toward the neighborhood of that solution. The next paragraphs present a brief explanation about these three algorithms. For further information about these methods, including a detailed overview of the algorithms, check the reference [[Karahan, 2008](#)].

The Territory Defining Evolutionary Algorithm (TDEA) is an algorithm which maintains two populations: a regular population, which has a fixed size, and an archive population, which has flexible size and contains the non-dominated individuals copied from the regular population. In each generation, a single offspring is created and tested considering the dominance for the acceptance in the regular population. If the offspring is accepted in the regular population the individuals in the archive population dominated by the offspring are removed from the archive. If the offspring is dominated by one individual in the archive population, it is rejected, otherwise a territory is defined around the individual closest to the offspring. The offspring is

accepted in the archive population only if it does not violate this territory.

Let  $y = (f_1, f_2, \dots, f_m)$  be an individual in the archive population. The territory of the individual  $y$  is defined as the region within a distance  $\delta$  of  $y$  in each objective among the regions that neither dominate nor are dominated by  $y$ . Mathematically, the territory of  $y$  contains all points in  $V$  defined by Equation 6.1

$$V = \{y' : |f_j - f'_j| < \delta, \text{ for } j = 1, 2, \dots, m \wedge y \text{ and } y' \text{ do not dominate each other}\} \quad (6.1)$$

where  $f_j$  and  $f'_j$  are the  $j$ -th objective values of  $y$  and  $y'$ , respectively, and  $\delta$  determines the territory size<sup>1</sup>. The objectives are previously scaled between 0 and 1 so that the territory sizes can be meaningfully selected in proportion to the scaled ranges of the objectives.

The territory defining property is responsible for the archive population diversity since each individual in this population controls a territory and disallows other individuals in its territory. The idea of favorable weights is employed in identifying the location of an individual. The favorable weights of an individual are a set of weights that minimize its weighted Tchebycheff distance from the ideal point.

In TDEA, the parameter  $\delta$  defines the territory size, which bounds the maximum number of individuals in the archive population. By allowing the territory size parameter  $\delta$  to have two different values the authors introduce a new version of TDEA: the preference-based TDEA (prTDEA).

The prTDEA possesses a mechanism to incorporate the DM's preferences and to modify the territory size of an individual depending on its location on

---

<sup>1</sup>The original paper considers the symbol  $\tau$  which is replaced here with the symbol  $\delta$  in an attempt to avoid confusion with the Kendall-tau distance already represented by  $\tau$ .

the PF. Before the optimization, the algorithm requires the DM to specify her/his preferable region  $R_P$ , defined by a set of Tchebycheff weight ranges, and defines the remaining space as  $R_U$ . Therefore, two values for the parameter  $\delta$  are stipulated, respectively:  $\delta_P$  and  $\delta_U$ . A small  $\delta_P$  maintains more individuals from the preferable region in the archive population while individuals located elsewhere have the eventual neighbors eliminated by a larger  $\delta_U$ . The prTDEA also requires a change in the acceptance procedure for the archive population: the  $\delta$  value is now determined by the region that contains the offspring. An illustration of different territory sizes is given by Figure 6.1, obtained from [Karahan and Köksalan \[2010\]](#).

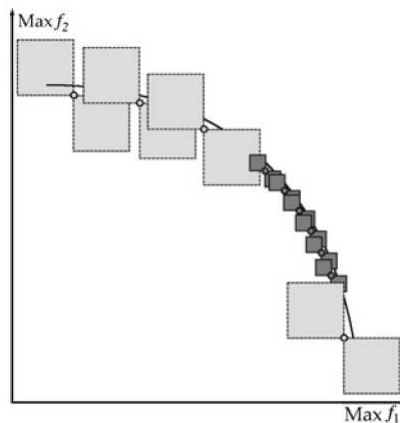


Figure 6.1: Different territory sizes in two dimensions.

Improving the prTDEA, the authors developed the Interactive Territory Defining Evolutionary Algorithm (iTDEA), an interactive approach that converges to the preferred solutions by progressively obtaining preference information from the DM. iTDEA identifies the preferable region interacting with the DM at predetermined generations. Interaction stages  $h = 1, 2, \dots, H$  are scheduled at the generations  $G_1, G_2, \dots, G_H$ , respectively. At the interaction stage  $h$ , the DM chooses the preferred individual among the filtered

sample of individuals obtained so far by the algorithm. The preferred individual determines the preferred weight region  $R^h$ , which is defined by a set of Tchebycheff weight ranges and it has a specific  $\delta$  value,  $\delta_h$ . Individuals falling in these regions are assigned smaller territories than those located elsewhere so that the density of the preferable regions is higher. For the acceptance in the archive population, the algorithm determines all  $R^h$  regions to which the offspring belongs and selects the last created region  $k$ , which has the smallest  $\delta_k$ .

The starting territory size is  $\delta_0$ , the final is  $\delta_H$ , and the intermediate  $\delta$  values calculated by an exponential decrease. The filtering procedure considers a modified dominance scheme similar to  $\epsilon$ -dominance to select individuals to be presented to the DM. Considering  $m$  objectives, the number of solutions in each interaction stage is  $P = 2m$ , except for the first stage in which  $P = 4m$ . The DM is required to find the best solution among those  $P$  filtered solutions in 4 or 6 interactions with the algorithm. Considering that only the ordinal information is available with binary comparisons, for each set of  $n$  elements, at least  $n - 1$  queries are made to the DM [Knuth, 1997]. Thus, a lower bound for the number of queries presented to the DM is  $10m - 4$  in 4 interactions and  $14m - 6$  in 6 interactions. In this work these estimates are references to the number of queries that are considered in constructing the NN-DM model for the DM's preferences.

The DM's preferences are simulated considering Tchebycheff, linear, and quadratic underlying utility functions. The algorithm is tested in three problems with two and three objectives. The runs are made with and without filtering and tests are performed with the incorporation of a Gaussian noise in the utility function calculations. iTDEA converges to the DM's preferable



region interactively in all selected test problems.

This chapter proposes the construction of a model for the DM's preferences to replace the DM in iTDEA by considering the NN-DM method. As the decision-making problem is related to a multi-objective optimization problem the domain for the approximating function  $\hat{\mathcal{U}}$  is induced from the domain of the EPF during the optimization process.

### 6.3 Computational Experiments

Computational experiments in multi-objective optimization problems with two and three objectives are reported in this section. iTDEA is tested considering 4 interactions with the DM in each problem. The iTDEA and NN-DM parameters chosen in each scenario are displayed in Tables 6.1 and 6.2, respectively.

	$2D$	$3D$
Ideal vector, $f^*$	(0, 0)	(0, 0, 0)
Population size	200	200
$\tau_0$	0.1	0.1
$\tau_H$	0.001	0.001
Number of iterations $T$	10 000	10 000
Number of replications $T$	50	50

Table 6.1: iTDEA parameters.

	$2D$	$3D$
Ideal vector, $f^*$	$(0, 0)$	$(0, 0, 0)$
Number of interactions $H$	4	4
Number of training points $T$	12	18
Estimate number of queries	16	26
Real number of queries	17	43

Table 6.2: NN-DM parameters.

In each instance the DM's underlying utility function is simulated considering the function given by Equation 5.3, reproduced here in the Equation 6.2

$$\mathcal{U}(\mathbf{p}) = \exp(-\mathbf{p} \cdot M \cdot \mathbf{p}^t), \quad (6.2)$$

in which the matrix  $M$  is instantiated in the two-dimensional problems with

$$M_{10} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad M_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6.3)$$

and in the three-dimensional problems with

$$M_{100} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad M_{111} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.4)$$

As a first example, a two-objective optimization problem with two decision variables proposed in Equation 6.5 is considered. The resulting EPFs are presented in Figure 6.2.

$$\begin{aligned} \mathbf{p} &= \{p_1, p_2\}, \mathbf{f} = (f_1, f_2), \\ f_i(\mathbf{p}) &= (\mathbf{p} - \mu_i) \cdot S \cdot (\mathbf{p} - \mu_i)^t, i = 1, 2 \\ S &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \mu_1 &= [1 \ 0] \\ & & \mu_2 &= [0 \ 1] \end{aligned} \quad (6.5)$$

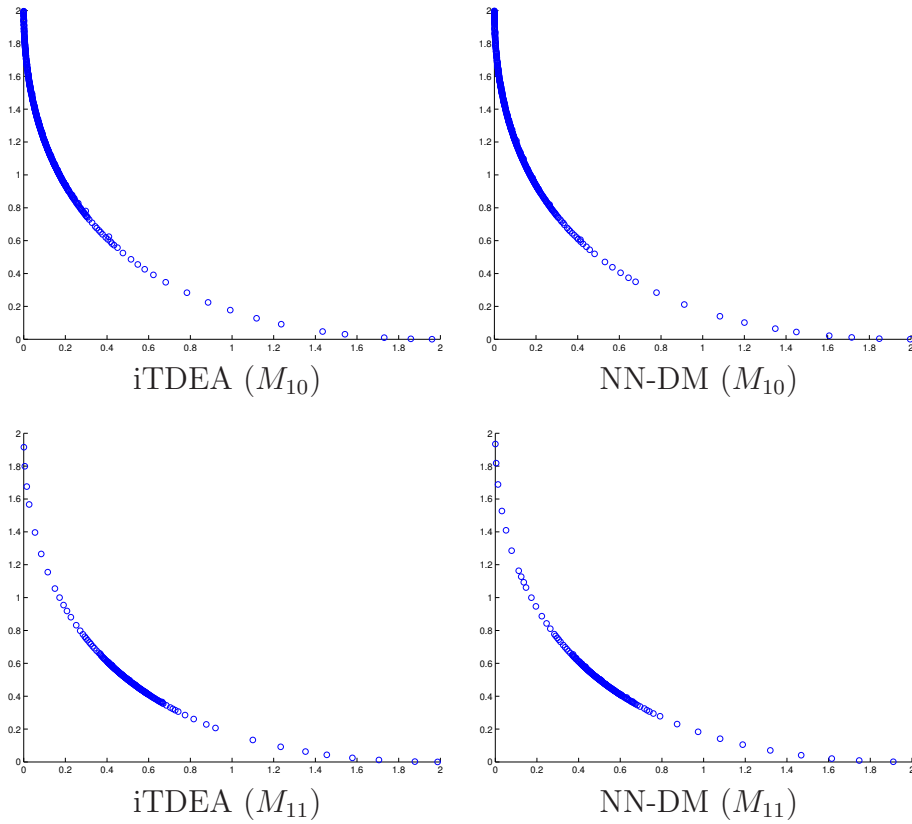


Figure 6.2: Estimates of the Pareto-optimal front from iTDEA and NN-DM methods.

Figure 6.3 presents the KTD and the total number of queries presented to the DM considering the number of alternatives in the domain between 10 and 50. This figure is constructed employing the BOXPLOT function<sup>2</sup>, a shelf routine from MATLAB<sup>®</sup>. The selected parameters values are highlighted in Figure 6.2.

<sup>2</sup>The BOXPLOT function produces a box plot of the data in X. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points the algorithm considers to be not outliers, and the outliers are plotted individually.

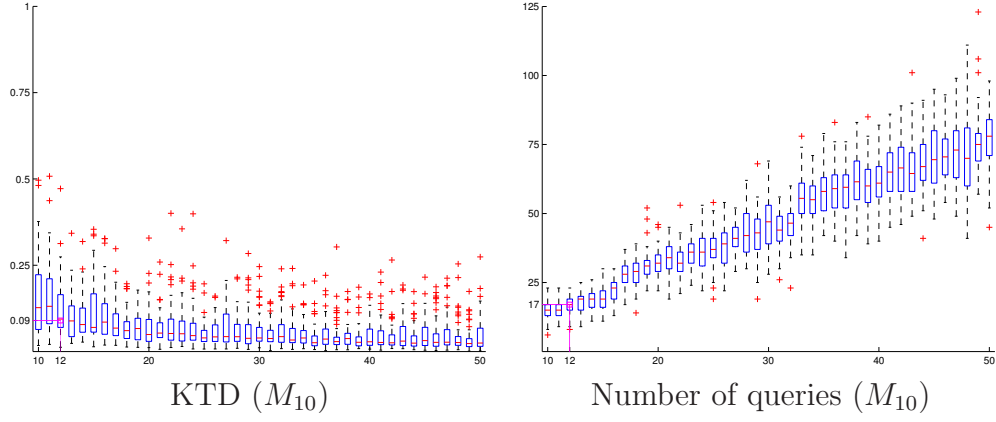


Figure 6.3: Statistical values.

Now an optimization problem with three objectives and three variables proposed in Equation 6.6 is considered. The resulting EPFs are presented in Figure 6.4.

$$\begin{aligned}
 \mathbf{p} &= \{p_1, p_2, p_3\}, \mathbf{f} = (f_1, f_2, f_3), \\
 f_i(\mathbf{p}) &= (\mathbf{p} - \mu_i) \cdot S \cdot (\mathbf{p} - \mu_i)^t, i = 1, 2, 3 \\
 S &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \begin{aligned} \mu_1 &= [1 \ 0 \ 0] \\ \mu_2 &= [0 \ 1 \ 0] \\ \mu_3 &= [0 \ 0 \ 1] \end{aligned}
 \end{aligned} \tag{6.6}$$

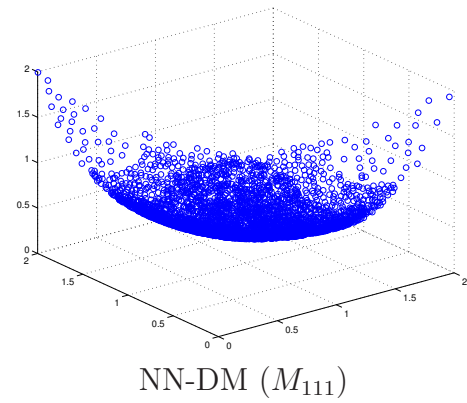
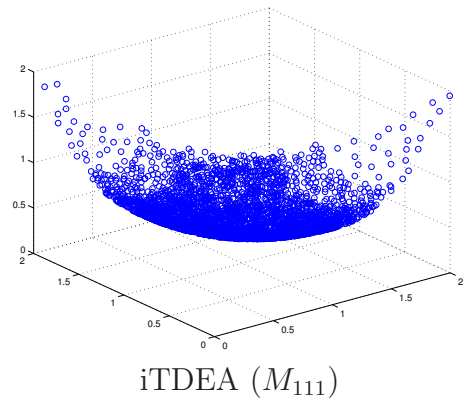
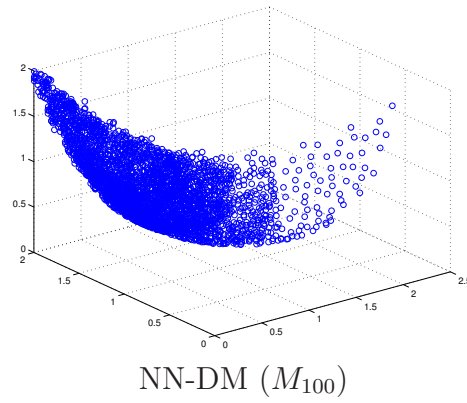
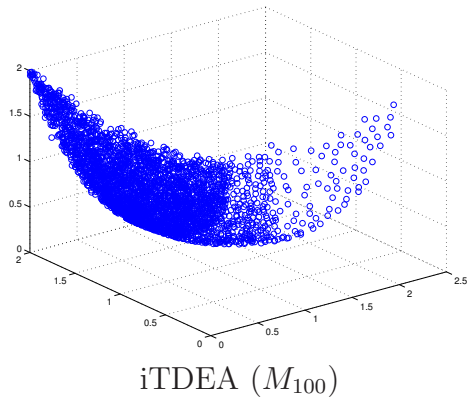


Figure 6.4: Estimates of the Pareto-optimal front from iTDEA and NN-DM methods.

Figure 6.5 presents the same information as Figure 6.3 considering now the three-dimensional problem.

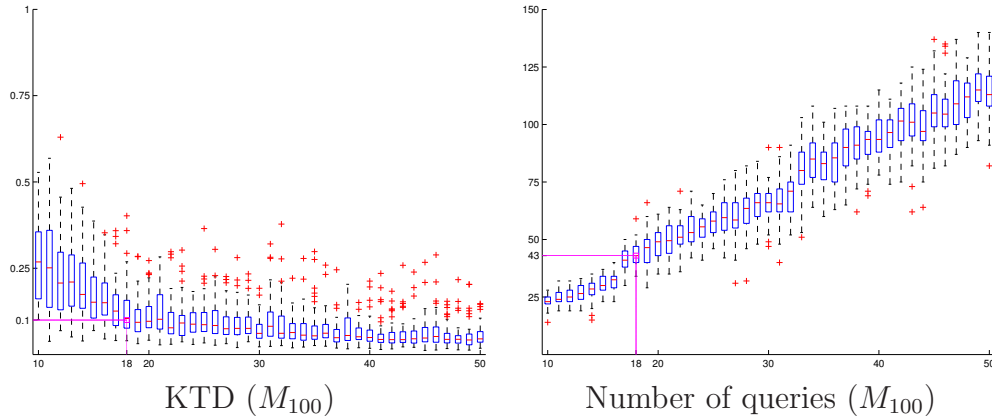


Figure 6.5: Statistical values.

The resulting EPFs obtained by both iTDEA and NN-DM methods do not present any relevant difference between the results of the two algorithms. The NN-DM model has  $\tau = 0.1$  in relation to the ideal utility function  $\mathcal{U}$  in all examples, with error of  $\pm 1\%$ , and the number of calls to the DM is similar to the one performed by iTDEA.

## 6.4 Discussion

This chapter presented a methodology to obtain information concerning the DM's preferences in one run of the Interactive Territory Defining Evolutionary Algorithm (iTDEA). After that specific run the DM's preferences become available for being considered in other decision-making processes. The preference information is stored on a Neural Network (the NN-DM model) trained considering ordinal information only, as provided by the queries to the DM. The proposed procedure is suitable when recurrent decisions with the same DM over different sets of alternatives are performed.

The information extracted from the DM by iTDEA is useful only inside the current optimization process and new queries are demanded from the DM whenever a similar problem is required to be solved. However, the information obtained from the DM should not be discarded, but employed in constructing a model for the DM's preferences able to solve similar decision-making problems on that specific region.

# Chapter 7

## The NN-DM Method And NSGA-II

### 7.1 Introduction

The development of multi-objective approaches for the design of an increasing number of real-world systems is a current trend. Although there are available, at this moment, several Evolutionary Multi-objective Optimization (EMO) techniques that aim to provide representative samplings of the Pareto-sets in multi-objective optimization problems [Fonseca and Fleming, 1995; Knowles and Corne, 2000; Deb *et al.*, 2002; Zitzler *et al.*, 2002], their application to the actual design of real systems still requires a further step in which, given a set of possible solution alternatives, a specific alternative should be chosen to be implemented. This step is usually recognized as a task that is attributed to a DM.

Current EMO techniques generally assume an a posteriori preference articulation scheme in which an entire sampling of the PF is preprocessed before being presented to the DM. Such a detailed sampling represents an



inefficient allocation of computational effort. On the one hand, the knowledge about the solutions in the non-preferable regions has the only role of informing the DM about the value of the trade-offs that are aggregated in the preferable regions. For this purpose, a rough sampling of the non-preferable regions would be enough. On the other hand, a fine sampling of the preferable regions, allowing the choice of a well-tuned solution, would be desirable. This fine sampling could become compromised if the computational budget were spent on the construction of a uniform sampling that covers all the PF including the non-preferable regions. This issue is even more concerning if the number of objectives is higher than three. In this situation, due to the exponential growth of the PFs related to the number of objectives, it may become even computationally impossible to produce a fine sampling of the PF. Therefore, procedures in which the DM progressively states her/his preferences in an interactive environment, which steers the multi-objective optimization algorithm in the search for PF solutions, are nowadays relevant [Köksalan and Karahan, 2010; Deb *et al.*, 2010].

This chapter proposes a modified NSGA-II algorithm called **Interactive Non-dominated Sorting algorithm with Preference Model** (INSPM). NSGA-II [Deb *et al.*, 2002] is a non-dominated sorting-based multi-objective algorithm in which a fast non-dominated sorting approach and a new selection operator are presented. INSPM reproduces NSGA-II algorithm while progressively interacts with the DM. The original crowding distance (CD) of NSGA-II is replaced in INSPM with a dynamic crowding distance (DCD) and combined with the NN-DM model bringing to INSPM the ability to distinguish preferable regions within the PF. The resulting mechanism is called here Neural Network Dynamic Crowding Distance (NN-DCD) and it is the

only modification introduced in the original NSGA-II algorithm proposed by [Deb \*et al.\* \[2002\]](#). The NN-DCD is responsible for the density control of solutions in INSPM providing a diversity maintenance strategy and a specific control on preferable and non-preferable regions. A specific instance of the NN-DM method is developed to allow the interaction between the DM and the NSGA-II (or any other EMO technique). These interactions occur in the initial stage of INSPM until the necessary information to construct a stable model is obtained. While the optimization process is running, the NN-DM model is tested and updated when necessary.

The DM's preferences inside INSPM allow an outcome sampling of the PF which is denser in the preferable regions and sparser in the non-preferable regions. The developed EMO methodology combines two features: (i) it interacts with the DM along the execution of the optimization task, such that the result of the optimization is guided by that interaction in a progressive preference articulation, and (ii) it ends the optimization procedure with a model for the DM's preferences, which becomes available for re-utilization in other instances of the same problem.

## 7.2 The Adapted NN-DM Methodology

INSPM considers the interaction between the NN-DM method and the NSGA-II algorithm. Therefore, Steps 1, 2, and 4 of the NN-DM method are adapted to work inside an evolutionary environment, as described next.

### 7.2.1 Step 1 - Domain Establishment

Since the considered methodology is evolutionary the domain may vary from one generation to another because the alternatives in the EPF are evolving to the final front (PF). Therefore, in the proposed methodology the domain  $\mathcal{D}$  is first represented by a set  $\mathcal{F}$  of  $n_{in}$  random simulated alternatives, which are taken into account to construct an initial NN-DM model. If the domain  $\mathcal{D}$  becomes outdated  $n_{step}$  random simulated alternatives belonging to the new domain are added to the set  $\mathcal{F}$  and a new model is constructed and evaluated. This step is executed until the model becomes stable in the new domain. This procedure updates the domain  $\mathcal{D}$  leading to an upgrade in the NN-DM model. The values of  $n_{in}$  and  $n_{step}$  vary regarding the amount of information available from the DM, the required model precision, and the problem dimension.

### 7.2.2 Step 2 - Ranking Construction

In the methodology presented here, although the partially ordered set is considered in constructing the NN-DM model, the set  $\mathcal{F}$  is kept sorted to enhance the efficiency of the interactions with DM. The total sorting is necessary to avoid additional queries to the DM when new EPFs stimulate an update in the NN-DM model. By keeping the set  $\mathcal{F}$  sorted the expected number of queries presented to the DM to insert any new alternative becomes approximately two.

The method for finding the sorting is Mergesort which has an average and worst-case performance of  $\mathcal{O}(n \log n)$ . The DM's ability to answer ordinal

queries is converted inside of Mergesort as the number of comparisons meaning that for each comparison the DM is asked to provide an answer to that query. By considering the Mergesort, the average and the worst-case number of comparisons are of the order  $\mathcal{O}(n \log n)$ . It is noteworthy that, as the alternatives to be sorted are the  $n_{in}$  alternatives, conceptually a small number, other sorting algorithms can be competitive with Mergesort. Appendix B presents a comparison between Mergesort and Quicksort corroborating the choice of Mergesort.

Even knowing that a total sorting provides more information than a partial sorting the total sorting is not suitable for the interpolation technique due to the integer ranking scale. Therefore, the alternatives are grouped into clusters, with a uniform number of alternatives per group, possibly excepting the higher level group. The number of levels, inspired by the partial sorting in the NN-DM method, is  $\log n$ , providing a number of alternatives in each level approximately given by  $(\log n)/n$ .

Figure 7.1 presents three graphics for exemplifying the ranking procedure in the original and adapted NN-DM methods. The first graphic is constructed from samples of the underlying utility function  $\mathcal{U}$  given by the Gaussian presented in Figure 4.7 and introduces its shape. The second graphic provides the ranking of the original NN-DM method in which the pivots are sorted and the remaining alternatives are compared and classified according to these pivots. The third graphic presents the ranking built by the adapted NN-DM method in which all the data is sorted and then clustered. The domain  $\mathcal{D}$  is represented by a grid with 400 alternatives in an attempt to provide a better visualization of the difference between the two procedures.

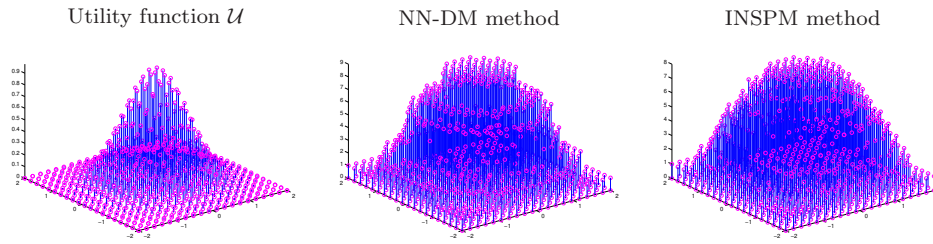


Figure 7.1: Ranking examples.

A comparison among these graphics shows that the shape of the function  $\mathcal{U}$  is captured by both rankings and the artificial neural network has the whole of constructing a function which approximates this shape and introduces answers to other alternatives within the same domain. It is important to remembering that the preference information is entirely captured by the level sets making the final ranking scale quite irrelevant to the process.

In the construction of an initial NN-DM model  $n_{in}$  simulated alternatives are sorted by the Mergesort algorithm. After clustering, a RBF network is employed in constructing the initial NN-DM model  $\hat{\mathcal{U}}$ . For each other step new solutions to the optimization problem may be available as the EPF is evolving. These alternatives may induce the model update demanding the construction of new simulated alternatives that must be inserted in the right position to keep the set  $\mathcal{F}$  sorted. The total ranking is not appropriate for the artificial neural network approximation, but it is crucial for the insertion of new alternatives induced by the genetic algorithm evolution. Since the set  $\mathcal{F}$  is already sorted the insertion of each new alternative costs to the DM approximately only two new queries. There are two steps to insert a new alternative on the current sorted set  $\mathcal{F}$ .

1. Estimate given by the current NN-DM model

Once there is already an NN-DM model, this model is employed in establishing a candidate position for the new alternative in the set  $\mathcal{F}$ . The current NN-DM model  $\hat{U}_c$  evaluates this new alternative and each alternative within the set  $\mathcal{F}$  producing a second sorting for the set  $\mathcal{F}$ . The position of the alternative immediately greater than the new alternative is the candidate position.

2. Adjustments provided by the DM to maintain the set  $\mathcal{F}$  sorted

Consider one more time the set  $\mathcal{F}$  sorted according to the DM's preferences. The new alternative is tested in the candidate position by asking two new queries to the DM concerning the new alternative and the eligible neighboring alternatives of the candidate position. If the preference is coherent the alternative is inserted in the candidate position and the procedure is finished. Otherwise, the candidate position is shifted to the position in which there is a ranking reversal. This last step, similar to a bubble sort step, is executed until there is no ranking reversal, leading the new alternative to be inserted in the position which maintains the set  $\mathcal{F}$  sorted.

The choice of starting the INSPM method with a totally sorted set is made to reduce the average number of queries presented to the DM. The algorithm can start by including one alternative at each time or with a large number of alternatives already sorted by the DM. In the first scenario the sorting algorithm is not required, but the quality of the first models is poor, leading to a high number of queries to adjust the estimated position of new alternatives. In the second scenario a better initial model is estimated, but several queries would be required to sort the initial set. In the experiments

reported here the value  $n_{in} = 10$  has presented a good trade-off concerning this issue. This value demands an inexpensive initial number of queries to the DM and produces an initial model which is able to find good estimated positions. As the optimization process goes on, the NN-DM model becomes better adjusted, demanding fewer queries to the DM and leading to estimated positions which are expected to be correct.

It is worth noticing that the dominance is considered during the whole procedure, replacing the DM when one alternative dominates another. For instance, in the process of updating the NN-DM model, if the model is already adjusted to the domain, new alternatives may be inserted in the set  $\mathcal{F}$  without new queries to the DM.

### 7.2.3 Step 4 - Performance Assessment

In the current methodology the KTD is calculated over two sets:  $\mathcal{F}$  and  $\mathcal{P}_{NN}$ . For each set, two sorting functions are considered in establishing the KTD:  $\hat{\mathcal{U}}_c$  and  $\mathcal{U}$ .

## 7.3 NN-DM Method and NSGA-II

This section presents the adaptation performed in the NSGA-II algorithm to allow it to indicate the preferable regions according to the DM's model. Section 7.3.1 reviews the Dynamic Crowding Distance (DCD) which provided improvements in the original Crowding Distance (CD) allowing the achievement of an EPF with well-distributed solutions. Section 7.3.2 introduces the Neural Network Dynamic Crowding Distance (NN-DCD). The NN-DCD is a

diversity maintenance strategy weighted by the NN-DM model which allows the INSPM algorithm to obtain a final sampling of the EPF in which the DM's preferable regions are denser.

### 7.3.1 Dynamic Crowding Distance

For improving NSGA-II [Luo \*et al.\* \[2008\]](#) introduced a dynamic crowding distance (DCD) based Diversity Maintenance Strategy (DMS) (DCD-DMS). This modification in the classical crowding distance (CD) was proposed to solve uniformity problems. The current strategy computes individual's DCD dynamically (avoiding situations in which there are individuals with small CD near to each other and all of them are deleted, creating a gap) on the basis of the difference degree between the CD of different objectives (avoiding situations in which this difference is significant).

Consider  $CD_i$  as the original crowding distance of the  $i^{\text{th}}$  individual in NSGA-II. The  $i^{\text{th}}$  individual's DCD is provided by Equation 7.1.

$$DCD_i = \frac{CD_i}{\log(1/V_i)}. \quad (7.1)$$

The term  $V_i$ , the variance of CDs of individuals which are neighbors of the  $i^{\text{th}}$  individual, is stated in Equation 7.2,

$$V_i = \frac{1}{M} \sum_{k=1}^M (|f_{i+1}^k - f_{i-1}^k| - CD_i)^2, \quad (7.2)$$

in which  $M$  is the number of objectives.

A similar idea is considered in this work, with a different purpose: the current crowding distance has the role of encoding both diversity and pref-



erence information providing a solution density control which expresses the DM's preferences.

### 7.3.2 Neural Network Dynamic Crowding Distance

In this chapter the NSGA-II algorithm is adapted to employ the DCD and the DCD is weighted by the NN-DM model to introduce different sampling densities according to the DM's preferences within the EPF.

Apart the DM's utility function structure the obtained model is able to provide the sorting of the alternatives. However, as the required information from the NN-DM method is ordinal no information is obtained related to the intensity of the preferences. For this reason, before the interaction with the DCD, the NN-DM model is linearized providing a smoother influence in the EPF. A parameter  $w$  is inserted to control the balance of how much the model may influence the results. The final diversity maintenance strategy, called Neural Network Dynamic Crowding Distance (NN-DCD), is given by Equation 7.3.

$$\text{NN-DCD} = \text{DCD} \cdot (\hat{\mathcal{U}})^w. \quad (7.3)$$

The NN-DCD modifies the original crowding distance by changing the perimeter estimated for the cuboid formed by considering the nearest neighbors as vertices. This change modifies the density of each region of the EPF allowing a greater number of alternatives in the DM's preferable regions. The NN-DM model interacts with the crowding distance operator as a by-product which makes the final effect in the concentration of the EPF sampling to continuously vary according to the NN-DM model.

The NN-DCD maintains in the current population the individuals in which the crowding distance is assigned to have an infinite value. In situations in which the model  $\hat{\mathcal{U}}$  provides a non-zero value the product handles the situation automatically; when  $\hat{\mathcal{U}}$  provides zero for some individual, the product inconsistency is solved manually by defining  $\infty \cdot 0 = \infty$ . This rule guarantees that in each instance the extent of the whole EPF is preserved.

The smoothness provided by the NN-DCD contrasts with the approach of reference points for representing the preferences that causes a discrete effect which may not be continuously controlled. In the current INSPM algorithm the weighting  $w = 0$  provides an EPF without any interference of the DM and, as the value of  $w$  grows, the influence of the DM's preferences in the optimization process becomes greater.

## 7.4 The Algorithm

The next sections explain the implementation details related to the adapted NN-DM method employed in constructing the NN-DM model inside the INSPM algorithm. The process starts with the initialization of the genetic algorithm and the construction of an initial NN-DM model. While the INSPM algorithm evolves, new estimates of the NN-DM model are constructed to maintain the model updated according to the current EPF. After a pre-established number of generations *gen* the final EPF and the updated NN-DM model are obtained.

### 7.4.1 NN-DM Model

The initial NN-DM model is constructed on the basis of the first population of the optimization process. In this step the population is random and there is no EPF because the optimization process has not yet begun. Fortunately, this fact is irrelevant for the construction of an initial model, since the relevant information is the domain of the alternatives and the information required from the DM about the simulated alternatives within this domain. The initial NN-DM model is stated according to the following procedures.

- The initial genetic population of INSPM provides the domain of the NN-DM model. The domain is set as the box constructed considering the minimum and maximum values of the objective functions. In that domain  $n_{in}$  simulated alternatives are randomly created.
- Mergesort algorithm is adapted to construct a total sorting of the simulated alternatives considering pairwise comparisons only. After that, the ranking is uniformly clustered in  $\log(n_{in})$  classes.
- The simulated alternatives are considered as inputs and their rank values as outputs to adjust the artificial neural network. The RBF network,  $\hat{\mathcal{U}}$ , is an approximation of the DM's utility function  $\mathcal{U}$ .

The NN-DM model should be kept updated while the optimization process finds new alternatives in each generation. The update procedure is similar to the construction of the initial NN-DM model, but certain differences must be mentioned. The following steps present the changes in the maintenance procedure of the NN-DM model along the INSPM iterations.

- The domain of the approximation is established considering the current genetic population. The domain definition procedure is the same as before and every time the NN-DM model is updated, at least  $n_{step}$  simulated alternatives are added to the set  $\mathcal{F}$ , which stores all previous simulated alternatives.
- As the set  $\mathcal{F}$  is kept sorted the previous NN-DM model is employed in finding an estimate of the position of each new alternative. After that, the DM is required to compare each new alternative and the neighboring alternatives of the candidate position to verify the model's accuracy. If the right position is selected, only two queries are made to insert each new alternative, regardless the number of alternatives in the sorted set  $\mathcal{F}$ . Otherwise, a procedure similar to a bubble sort step is employed in finding the right position of the alternative to be inserted. After the insertion of new alternatives the ranking is uniformly clustered. As stated previously, the dominance is checked before each query being presented to the DM, likely avoiding the submission of queries to her/him.
- The process of finding an artificial neural network to approximate the DM's utility function remains the same as the one employed in the initial NN-DM model's construction.
- The KTD is calculated for  $nvs$  validation sets with  $nvp$  points each one sorted according to the functions  $\hat{\mathcal{U}}_c$  and  $\mathcal{U}$  under a tolerance  $tol_{st}$ . The resulting KTD is the average of the obtained values of each set.

## 7.4.2 INSPM Main Program

The INSPM algorithm starts by the initialization of the variables in the optimization process and the construction of an initial NN-DM model. An initial population is created with  $N_{pop}$  individuals, which is the fixed size for the population in the beginning of each generation. A non-dominated sorting procedure sorts the initial population by the front rank followed by the NN-DCD value. Then the following steps are executed iteratively:

### Tournament Selection

A binary tournament selection is employed in the INSPM algorithm. In this procedure two individuals are selected randomly and their fitness are compared. Selection is based on the EPF rank and if the individuals have the same front rank then their NN-DCD are compared. A lower EPF rank followed by a higher NN-DCD are the selection criteria. The individual with better fitness is selected as a parent. The tournament selection is carried out until the pool size is filled.

### Crossover and Mutation Operators

INSPM employs Simulated Binary Crossover (SBX) and polynomial mutation. The crossover probability is  $pc = 0.9$  and the mutation probability is  $pm = 1/V$ , in which  $V$  represents the number of objective functions. The distribution indices for crossover and mutation operators are both equal to 20.

### Offspring Generation

The parents are selected for reproduction to produce offspring. The intermediate population is the combined population of parents and offspring individuals of the current generation. The size of this intermediate population is two times the size of the initial population ( $2N_{pop}$ ).

### **Non-dominated Sorting**

The non-dominated sorting procedure updates the values of rank, CD, and NN-DM model for each individual of the intermediate population.

### **New Population Selection**

This procedure selects the  $N_{pop}$  individuals of the new population on the basis of rank and NN-DCD. Initially each front of the intermediate population is copied into the new population, one by one, from the best one to the worst one, until the moment in which the inclusion of a new complete front results in exceeding the size of the new population. At this point a dynamic procedure involving this last front starts: DCD and NN-DCD are calculated and the worst individual is removed on the basis of the NN-DCD value. This procedure is repeated until the new population size becomes equal to  $N_{pop}$ .

### **NN-DM Model Update**

In the beginning of the first generation an initial NN-DM model is constructed:  $\hat{\mathcal{U}}_c$ . The model  $\hat{\mathcal{U}}_c$  is compared under a tolerance  $tol_{up}$  with the underlying utility function  $\mathcal{U}$  by calculating the KTD in relation to the current population  $\mathcal{P}_{NN}$ . If the KTD value does not satisfy the tolerance  $tol_{up}$  a new NN-DM model is constructed considering

the information about the current population and the former model is disregarded.

Steps 1-3 are the same present in the NSGA-II algorithm, Steps 4-5 are modified versions to allow the interaction with the NN-DM model, and the Step 6 is executed by the INSPM algorithm to maintain the model updated. Algorithm 3 presents the INSPM main algorithm and Table 7.1 presents the parameters and their description.

---

**Algorithm 3** INSPM Algorithm

---

```

1: Parameters initialization ( $tol_{up}$ ,  $gen$ ,  $w$ )
2: Objective function description
3: Initial population construction
4: Initial NN-DM model construction
5: Non-dominated sorting
6: for  $i \leftarrow 1$  to  $\#gen$  do
7:   Tournament selection
8:   Crossover and mutation operators
9:   Offspring generation
10:  Non-dominated sorting
11:  New population selection
12:  if  $KTD < tol_{up}$  then
13:    NN-DM model updating
14:  end if
15: end for

```

---

```

16: procedure NEW POPULATION SELECTION( $pop$ ,  $rbf$ )
17:   Sort  $pop$  based on the rank
18:   Add each front based on the rank
19:   while  $\#pop \neq N_{pop}$  do
20:     Calculate the CD
21:     Calculate the DCD
22:     Calculate the NN-DCD
23:     Remove the worst individual of  $pop$ 
24:   end while
25: end procedure

```

---

Name	Description	Comment	Value
$gen$	Generations	This parameters defines the duration of the program.	50
$N_{pop}$	Population size	A bigger population provides a better sampling of the EPF.	50
$nvs$	Validation sets	This parameters establishes a valid statistical analysis.	30
$nvp$	Points in each validation set	This parameters controls the cover of the space.	50
$n_{in}$	Alternatives in the initial NN-DM model	This parameter defines a cover of the space for initial NN-DM model.	10
$n_{step}$	Alternatives added in each model update	This parameters controls the updating rate on the NN-DM model.	2
$tol_{up}$	Tolerance for the difference between $\hat{\mathcal{U}}_c$ and $\mathcal{U}$ over $\mathcal{P}_{NN}$	A more tolerant parameter demands more generations.	0.1
$tol_{st}$	Tolerance for the difference between $\hat{\mathcal{U}}_c$ and $\mathcal{U}$ over $\mathcal{F}$	A more tolerant parameter demands more model updates.	0.1

Table 7.1: Parameters of the INSPM algorithm.

## 7.5 Computational Experiments

This section presents the results obtained by INSPM in the following situations: (i) guided by the DM’s utility function  $\mathcal{U}$  (Section 7.5.1), (ii) guided by the NN-DM model  $\hat{\mathcal{U}}$  (Section 7.5.2), and (iii) compared with iTDEA (Section 7.5.3). The DM’s underlying utility function  $\mathcal{U}$  is simulated considering the same function given by Equation 5.3 and restated here in Equation 7.4:



$$\mathcal{U}(\mathbf{p}) = \exp(-\mathbf{p} \cdot M \cdot \mathbf{p}^t) \quad (7.4)$$

in which  $\mathbf{p}$  represents an alternative. Three different  $M$  matrices are chosen to simulate different kinds of preferences: a balanced preference between the two objective functions ( $M_{11}$ ), a matrix that favors the first objective ( $M_{01}$ ), and a matrix that favors the second objective ( $M_{10}$ ). Figure 7.2 presents the matrices  $M_{11}$ ,  $M_{01}$ , and  $M_{10}$  and their respective underlying utility functions  $\mathcal{U}_1$ ,  $\mathcal{U}_2$ , and  $\mathcal{U}_3$ .

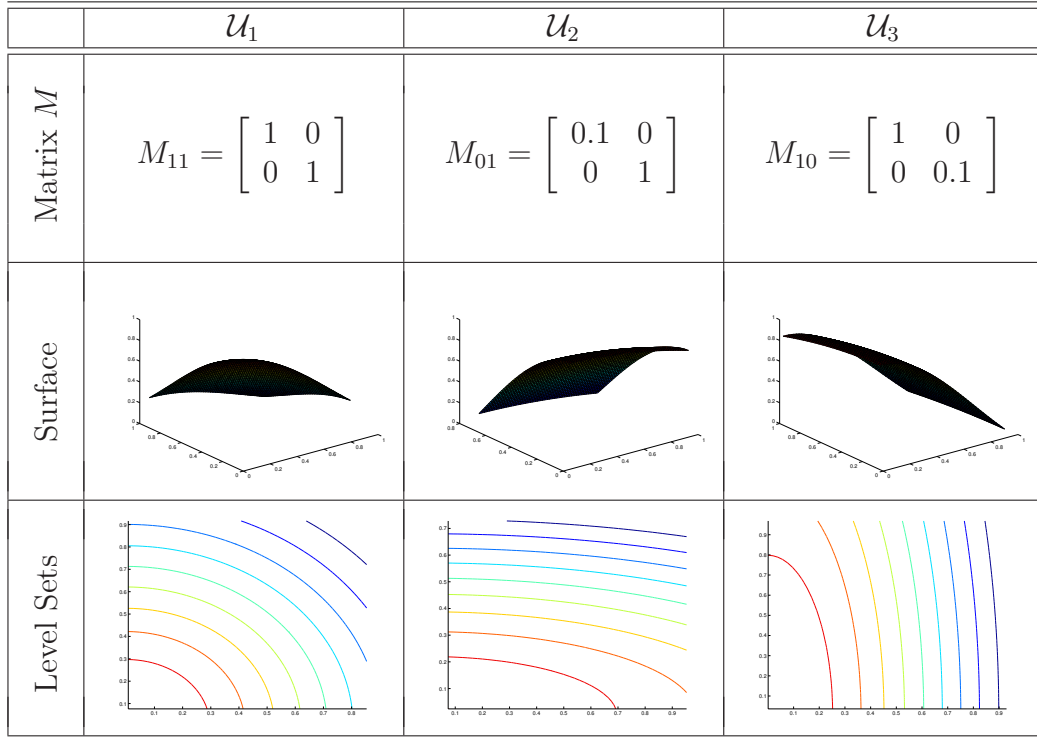


Figure 7.2: DM's underlying utility functions.

The multi-objective optimization problem solved for all the tests in this chapter is given by the ZDT4 function [Zitzler *et al.*, 2000]. The test function ZDT4 contains  $21^9$  local PFs and, therefore, tests the EMO's ability to deal

with multi-modality. Equation 7.5 describes ZDT4 function in which  $m = 10$ ,  $x_1 \in [0, 1]$ , and  $x_2, \dots, x_m \in [-5, 5]$ .

$$\begin{cases} f_1(x_1) = x_1 \\ g(x_2, \dots, x_m) = 1 + 10(m - 1) + \sum_{i=2}^m (x_i^2 - 10 \cos(4\pi x_i)) \\ h(f_1, g) = 1 - \sqrt{f_1/g} \end{cases} \quad (7.5)$$

The global PF is formed with  $g(x) = 1$  and the best local PF with  $g(x) = 1.25$ . Note that not all local PF sets are distinguishable in the feature space.

### 7.5.1 INSPM and Utility Function

This section presents the results obtained considering directly the DM's underlying utility function  $\mathcal{U}$  instead of the NN-DM model in the INSPM algorithm. The NN-DCD diversity maintenance strategy is tested here and the results are taken into account to estimate the right moment to stop the INSPM algorithm with the NN-DM model guiding the search (Section 7.5.2).

Figures 7.3 and 7.4 present the results of the INSPM algorithm guided by the utility function  $\mathcal{U}_1$ . Each graphic has a specific  $w$  value and shows the  $\mathcal{U}$  level sets and the EPF  $\mathcal{P}_{\text{DM}}$ . The values of the parameter  $w$  are indicated in the figures. The average time for each run is 25 seconds.

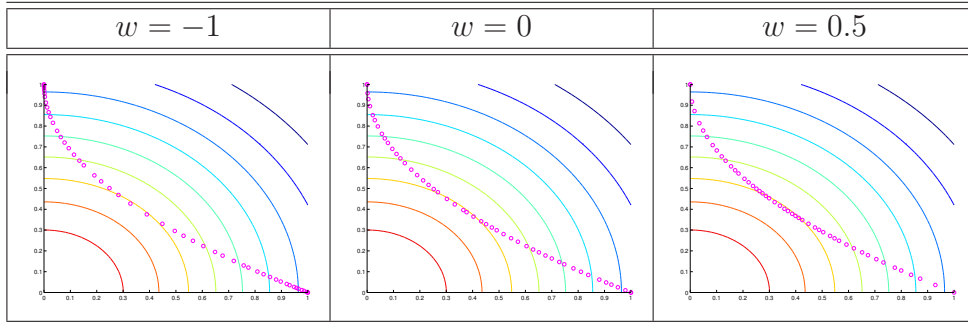


Figure 7.3: Utility function  $\mathcal{U}_1$ : results for  $w = -1$ ,  $w = 0$ , and  $w = 0.5$ .

Figure 7.3 shows that a uniform sampling of the EPF is obtained for  $w = 0$  while a positive value  $w = 0.5$  produces a sampling that is denser in the region of the solutions that are preferable in the utility function  $\mathcal{U}_1$ . As a curiosity, it is also shown that a negative value of  $w$  produces a denser sampling of the regions that are non-preferable in  $\mathcal{U}_1$ .

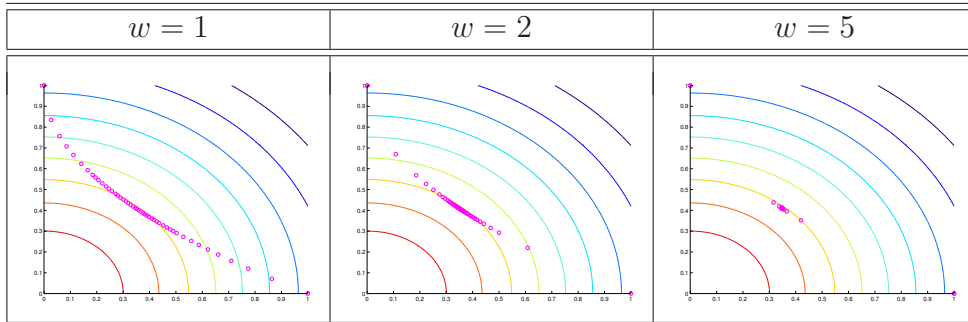


Figure 7.4: Utility function  $\mathcal{U}_1$ : results for  $w = 1$ ,  $w = 2$ , and  $w = 5$ .

Figure 7.4 presents the effect of increasing the value of  $w$ . Small values of  $w$  lead to a sampling of the EPF which is well-distributed while slightly concentrated on the preferable regions. High values of  $w$  produce a more concentrated sampling of the EPF nearby the DM's preferable regions. It

should be noticed that in all these scenarios the whole extent of the EPF is covered; in particular, the extremal solutions always appear.

The tests show that the replacement of the conventional CD for the NN-DCD is an effective strategy to produce a denser sampling of the DM's preferable regions, as long as the DM's utility function  $\mathcal{U}$  is available.

### 7.5.2 INSPM and NN-DM Method

Section 7.5.1 has established that the proposed dynamic crowding distance is effective as a diversity maintenance strategy in a simplified situation in which the algorithm relies on queries presented directly to the DM. Now the complete INSPM algorithm is presented considering the NN-DM model. Figure 7.5 presents the results obtained by INSPM guided by the NN-DM model considering the underlying utility function  $\mathcal{U}_1$ . The level sets on the figures correspond to the level sets of the resulting NN-DM model. The  $w$  values are indicated in each graphic.

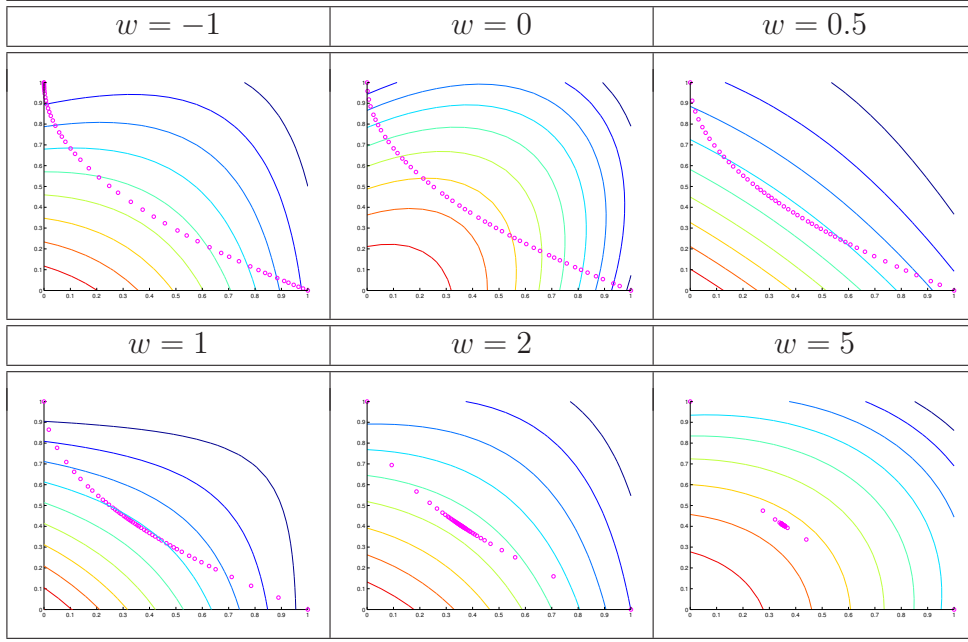


Figure 7.5: Utility function  $\mathcal{U}_1$ : INSPM results.

The results presented in Figure 7.5 show that under the condition of preference pressure ( $w > 0$ ) INSPM employing the NN-DM model produces a concentrated sampling around the DM's preferable regions. The obtained region is the same one constructed when the DM's underlying utility function is employed in giving answers to all queries instead of the NN-DM model (Figures 7.3 and 7.4). Figure 7.5 also presents the level sets of each obtained model showing that, even when the model is slightly different from the underlying utility function, the DM's preferences are well represented in the final EPF.

Table 7.2 presents the average values obtained for the KTD and the number of queries presented to the DM considering 50 runs. With a higher  $w$  there is less diversity in the genetic population favoring ranking reversals and

increasing the demand to the DM. The author is at present studying other ways of validating the DM’s model to avoid this extra demand, considering situations in which previous models are already available, as shown in Figure 7.5.

	$w = -1$	$w = 0$	$w = 0.5$	$w = 1$	$w = 2$	$w = 5$
KTD	0.0664	0.0513	0.0451	0.0762	0.0431	0.0538
Queries	20	16	18	23	17	15

Table 7.2: KTD and number of queries in INSPM.

The average time for each INSPM run is 1500 seconds. This time is much higher than the running time presented in Section 7.5.1. The extra cost comes from the routine that evaluates the RBF network for each individual in each generation. However, as the main goal of the current methodology is to reduce the number of queries to the DM, the computational time for the RBF function evaluation does not constitute an important problem. Once the NN-DM model becomes fitted there is no further requirement of presenting queries to the DM.

The resulting NN-DM model running together with the NSGA-II algorithm in other instances of the same problem spends approximately the same time. It is worth mentioning that when the model is employed just in sorting the final EPF of other instances less than a minute is spent.

### 7.5.3 Comparison with iTDEA

A comparison is now provided with iTDEA. In the original paper the DM’s preferences are simulated considering Tchebycheff utility functions with

weights that favors the objectives in a different way.. Here, the utility functions employed are the same presented in Equation 7.4 with  $M$  matrices given by Figure 7.2. Once more the number of queries required from iTDEA is estimated considering that only ordinal information is available from the DM.

Figure 7.6 presents the models for the DM's preferences with underlying utility functions given by  $\mathcal{U}_1$ ,  $\mathcal{U}_2$ , and  $\mathcal{U}_3$ . The first row presents the level sets of the utility function  $\mathcal{U}$  and the second row presents the level sets of the model  $\hat{\mathcal{U}}$  obtained in a sample run of the INSPM method with  $w = 1.25$ .

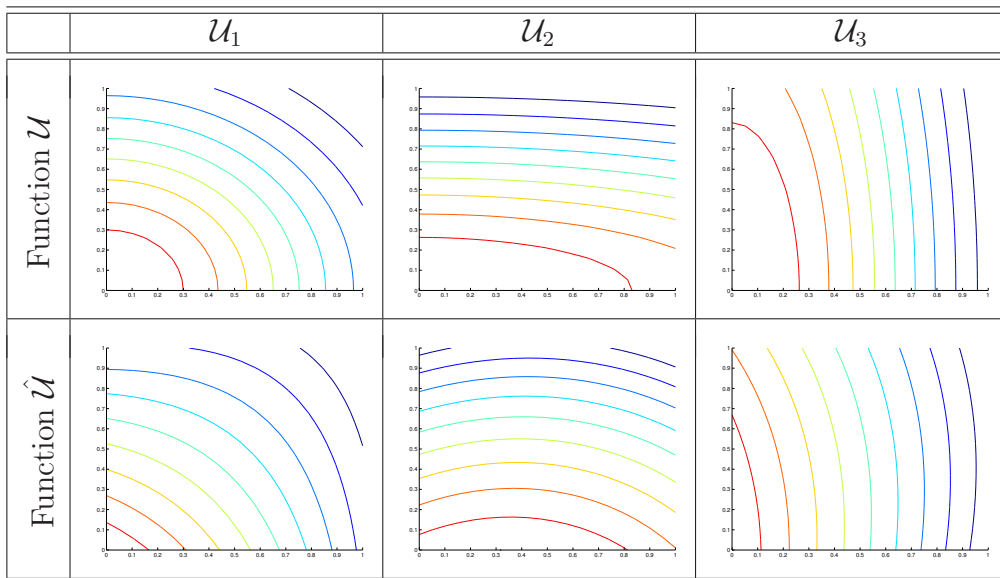


Figure 7.6: Level sets of the functions  $\mathcal{U}$  and  $\hat{\mathcal{U}}$ .

Figure 7.7 displays plots of sample runs of iTDEA considering the filtered mode with four and six interactions and INSPM considering  $w = 1.25$ . The  $w = 1.25$  value was determined by a trial-and-error process in an attempt to estimate a final EPF resembling iTDEA with a higher number of interactions.

The number of queries required from both methods is the average of the three considered scenarios. The average time for each iTDEA run is 50 seconds. Table 7.3 presents the chosen parameters for the iTDEA method; the INSPM parameters are the same presented in Table 7.1.

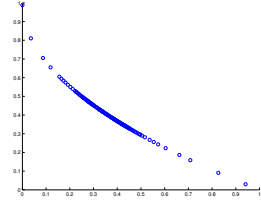
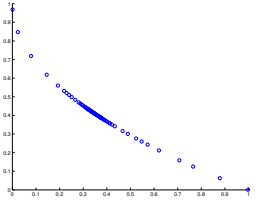
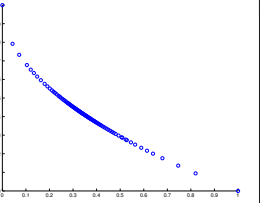
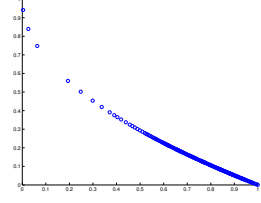
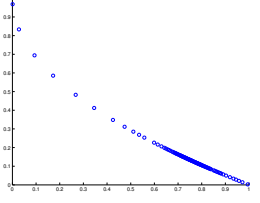
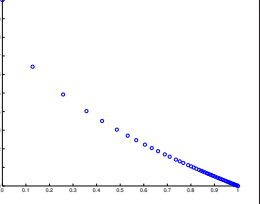
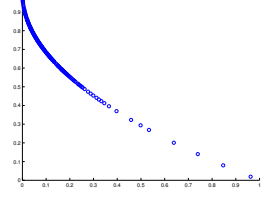
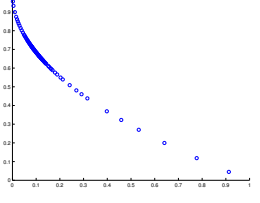
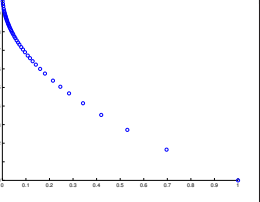
	iTDEA Four interactions	iTDEA Six interactions	INSPM $w = 1.25$
$\mathcal{U}_1$			
$\mathcal{U}_2$			
$\mathcal{U}_3$			
Queries	16	22	32

Figure 7.7: Comparison between iTDEA and INSPM methods.



Parameter	Value
Ideal vector, $f^*$	(0, 0)
Population size	50
$\tau_0$	0.1
$\tau_H$	0.001
Number of iterations $T$	2000

Table 7.3: iTDEA parameters.

The iTDEA algorithm in the unfiltered mode is more similar to the INSPM algorithm, since all the individuals of the current population are affected by the NN-DM model. However, the iTDEA's final results considering both modes are similar and, since the number of queries required in the unfiltered mode is a multiple of the population size, the author has considered the iTDEA's performance in the filtered mode a fairer comparison.

In iTDEA a higher number of interactions with the DM allows a better convergence to the desired location. As the number of queries depends on the number of interactions, a restriction on this amount prevents the determination of a sampling of the EPF that becomes smooth.

The INSPM algorithm provides results that exhibit a smooth behavior because the NN-DM model is active during the whole process. Furthermore INSPM also provides, at the end of the algorithm execution, a model for the DM's preferences that can be employed repeatedly in recurrent problems.

## 7.6 Discussion

This chapter presented an algorithm for interactive multi-objective optimization: INSPM. This new algorithm is based on the NSGA-II algorithm

with the replacement of the usual crowding distance operator for a dynamic crowding distance weighted by a DM's preference model (the NN-DM model). The NN-DM model is constructed considering the ordinal information provided by the DM about her/his preferences acquired along the algorithm interactions with the DM. INSPM requires a number of queries from the DM compatible with other methods from the literature and still delivers a model for the DM's preferences.

The NSGA-II algorithm was the chosen EMO algorithm to interact with the NN-DM method, but it is not difficult to embed a model for the DM's preferences in different EMO algorithms. The main idea is to adapt the diversity maintenance strategy to consider the preferences while the diversity is controlled. The CD limitations on finding solutions uniformly distributed has prevented the NN-DM method of achieving the same quality of results considering the NSGA-II algorithm in higher dimensions. Different algorithms guided by the NN-DM model should fix this problem.

The INSPM algorithm has, as an advantage, a fine-tuning ability to control the EPF sampling density, which is not provided by other methods. A parameter  $w$  controls the asymmetry intensity of the sampling density along the EPF: for  $w = 0$  there is no asymmetry (the EPF is uniformly sampled) and the higher the  $w$ , the greater the asymmetry, until the limiting situation in which only the preferred solution and the extremes of the EPF are sampled. Moreover, INSPM provides a sampling that covers the whole EPF while guaranteeing that the most preferable regions receive a denser sampling and the non-preferable regions have a sparser sampling.

In new instances of the same decision-making problem (the recurrent

decision-making problems), the resulting NN-DM model constructed inside INSPM can be employed without new queries to the DM, except when the domain requires updates. In this situation new queries can be proposed to the DM at a low cost.

# Chapter 8

## Polymer Extrusion Process

### 8.1 Introduction

This chapter presents further adaptations in the NN-DM method to find a model for the DM's preferences in a real scenario: the single screw extrusion, which is an important polymer processing technology, allowing the production of products such as pipes, film, profiles, fibers, and so forth. The process performance depends on three different type of parameters: the polymer properties, the system geometry, and the operating conditions. The objectives considered in the definition of the multi-objective optimization problem usually studied are: the mass output of the machine ( $\mathbf{Q}$ ), the average melt temperature of the polymer at die exit ( $\mathbf{T}_{\text{melt}}$ ), the power consumption required to rotate the screw ( $\mathbf{P}$ ), the capacity of pressure generation ( $\mathbf{P}_{\text{max}}$ ), the mixing capacity measure by the average of deformation ( $\mathbf{W}$ ), and the length of screw required to melt the polymer ( $\mathbf{L}_{\text{melt}}$ ).

The decision-making problem considered here works with solutions of the multi-objective optimization problem, that is, estimates of the PF pro-

vided by the Reduced Pareto Set Genetic Algorithm with Elitism (RPSGAe) [Gaspar-Cunha and Covas, 2004; Gaspar-Cunha, 2009]. RPSGAe is an algorithm based on the assignment of the fitness through a ranking function obtained using a clustering algorithm. This optimization methodology has already been applied to the optimization of the operating conditions and to the design of screws for polymer extrusion. The results obtained by Gaspar-Cunha and Covas [2004] showed that RPSGAe is able to find solutions with physical meaning. Further details of the modeling routine implemented can be found elsewhere [Gaspar-Cunha, 2009].

Two multi-objective optimization problems are investigated and in each scenario three sets of estimates of the PF are available considering different decision variables. The DM is required to indicate preference relations among alternatives in a specific domain leading the adapted NN-DM method to the construction of a model for the DM's preferences. The resulting NN-DM model is responsible for providing the sorting of the solutions within the EPFs from the best to the worst one according to the DM's preferences.

## 8.2 Available Data

As the single screw extrusion is a computationally expensive multi-objective optimization problem this application deals directly with estimates of different PFs obtained by RPSGAe. The objectives considered in the multi-objective optimization problems are: the process performance characterized by the mass output of the machine ( $\mathbf{Q}$ ), the power consumption required to rotate the screw ( $\mathbf{P}$ ), and the degree of mixing capacity measure by the average of deformation induced, denoted WATS ( $\mathbf{W}$ ). Two problems are

then considered: Mass Output  $\times$  Power Consumption and Mass Output  $\times$  WATS. In each problem three EPFs are available considering different decision variables. The first set considers the operating conditions (four decision variables), the second set considers the geometry (six decision variables) and, finally, in the third set both types of decision variables are considered. Table 8.1 resumes the information about the available EPFs and Table 8.2 provides the objectives, aim of optimization, and range of variation.

EPF	Objectives	Optimization Type	Number of Decision Variables
$\mathbf{QP}_1$	<b>Q and P</b>	Operating conditions	Four variables
$\mathbf{QP}_2$	<b>Q and P</b>	Geometry	Six variables
$\mathbf{QP}_3$	<b>Q and P</b>	Both	Ten variables
$\mathbf{QW}_1$	<b>Q and W</b>	Operating conditions	Four variables
$\mathbf{QW}_2$	<b>Q and W</b>	Geometry	Six variables
$\mathbf{QW}_3$	<b>Q and W</b>	Both	Ten variables

Table 8.1: Multi-objective optimization problems in a single screw extrusion process.

Objective	Aim of optimization	Range of variation
Mass Output (kg/hr)	Maximization	[1, 20]
Power consumption (W)	Minimization	[0, 9200]
WATS	Maximization	[0, 1300]

Table 8.2: Objectives, aim of optimization, and range of variation.

Figure 8.1 presents the available EPFs considering  $\mathbf{Q} \times \mathbf{P}$  (EPFs  $\mathbf{QP}_1$ ,  $\mathbf{QP}_2$ , and  $\mathbf{QP}_3$ ) and  $\mathbf{Q} \times \mathbf{W}$  (EPFs  $\mathbf{QW}_1$ ,  $\mathbf{QW}_2$ , and  $\mathbf{QW}_3$ ). The domain is established by the range of variation presented in Table 8.2.

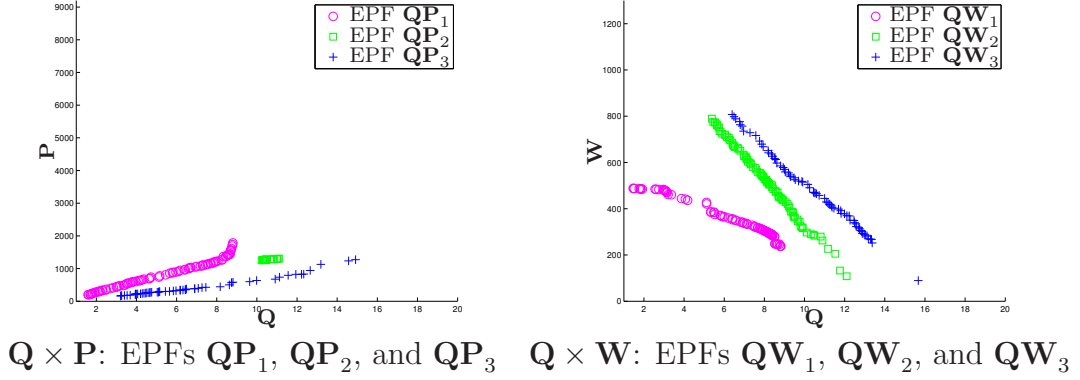


Figure 8.1: Available estimates of the Pareto-optimal fronts.

### 8.3 Interaction with the DM

The decision-making matrix  $\mathcal{M}$  is a matrix filled by the DM to assist the NN-DM method in the construction of a model for the DM's preferences. Each element  $m_{ij}$  of  $\mathcal{M}$  is defined as given in Equation 8.1.

$$\begin{cases} m_{i,j} = -1, & \text{if } a_i \text{ is preferable than } a_j; \\ m_{i,j} = 0, & \text{if } a_i \text{ and } a_j \text{ are equivalents;} \\ m_{i,j} = 1, & \text{if } a_j \text{ is preferable than } a_i. \end{cases} \quad (8.1)$$

Considering  $n$  the number of partitions in each dimension and  $m$  the number of objectives the total number of simulated alternatives is given by  $n^m$ . Therefore, the total number of pairwise comparisons is given by  $n^{2m}$  which corresponds to the number of entries of the decision-making matrix. The information required from the DM is reduced by the dominance and the comparisons between the same alternative (the matrix diagonal). The symmetry also develops an important role: given a utility function  $\mathcal{U}$  and two alternatives  $a$  and  $b$ , with  $\mathcal{U}(a, b) = \mathcal{U}(b, a) = a$ , the entries in the decision-

making matrix are  $m_{\delta(a),\delta(b)} = -1$  and  $m_{\delta(b),\delta(a)} = 1$ , so that only one query is required from the DM<sup>1</sup>.

For exemplifying the whole described process consider a decision-making problem with two objective functions  $F_1$  and  $F_2$  whose aim is to minimize. The function  $F_1$  is defined in the interval  $[a_1, b_1]$  and the function  $F_2$  is defined in the interval  $[a_2, b_2]$ . Assuming the number of partitions in each dimension of the grid established as 2 the interval partition matches with the interval extremes. Therefore, the decision-making process consists of 16 queries demanded by the combination of elements in the set  $\{[a_1, a_2], [a_1, b_2], [b_1, a_2], [b_1, b_2]\}$ . Table 8.3 presents the unfilled decision-making matrix  $\mathcal{M}$  for this example, with variables  $s_i, i = 1, \dots, 16$ , representing the entries.

Table 8.3: Example of a decision-making matrix  $\mathcal{M}$ .

$F_1 \times F_2$	$[a_1, a_2]$	$[a_1, b_2]$	$[b_1, a_2]$	$[b_1, b_2]$
$[a_1, a_2]$	$s_1$	$s_2$	$s_3$	$s_4$
$[a_1, b_2]$	$s_5$	$s_6$	$s_7$	$s_8$
$[b_1, a_2]$	$s_9$	$s_{10}$	$s_{11}$	$s_{12}$
$[b_1, b_2]$	$s_{13}$	$s_{14}$	$s_{15}$	$s_{16}$

The solutions to the queries  $s_i$  are divided into four groups.

**Equivalence** The solutions  $s_1, s_6, s_{11}$ , and  $s_{16}$  derive from queries made between the same alternative (the matrix diagonal). Therefore, the answer is 0 since the solutions are equivalent.

**Dominance** The solutions  $s_2, s_3, s_4, s_8$ , and  $s_{12}$  are obtained considering

<sup>1</sup>The function  $\delta$  represents the grid's position of an alternative.



the dominance, since  $a_1 < a_2$ ,  $b_1 < b_2$  and the aim of the optimization for both objectives is minimization.

**Symmetry** The solutions  $s_5$ ,  $s_9$ ,  $s_{13}$ ,  $s_{14}$ , and  $s_{15}$  result from symmetry, since if the preferred alternative between  $a$  and  $b$  is, for example,  $a$ , the preferred alternative between  $b$  and  $a$  is also  $a$ .

**Decision-maker** The solutions  $s_7$  and  $s_{10}$  demand the DM's expertise. Considering that  $s_7$  and  $s_{10}$  are provided from queries between the same alternatives only one query has to be presented to the DM.

Table 8.4 shows the decision-making matrix partially filled by considering the equivalence, the dominance, and the symmetry among the alternatives. This matrix is then presented to the DM who needs to provide an answer to the remaining queries.

Table 8.4: Example of a filled decision-making matrix  $\mathcal{M}$ .

$F_1 \times F_2$	$[a_1, a_2]$	$[a_1, b_2]$	$[b_1, a_2]$	$[b_1, b_2]$
$[a_1, a_2]$	0	1	1	1
$[a_1, b_2]$	-1	0	$s_7$	1
$[b_1, a_2]$	-1	$s_{10}$	0	1
$[b_1, b_2]$	-1	-1	-1	0

In the real scenario considered here the number of partitions in each dimension of the grid is established as 4. This value provides enough information for the NN-DM method to construct suitable NN-DM models for the DM's preferences. Considering the range of variation in Table 8.2 the partitions are displayed in Table 8.5.

Symbol	Objective	Partition
<b>Q</b>	Mass Output	[1, 7, 14, 20]
<b>P</b>	Power consumption	[0, 3067, 6134, 9200]
<b>W</b>	WATS	[0, 434, 867, 1300]

Table 8.5: Partitions of each objective.

As each optimization problem is composed of two objective functions there are 16 pairs of simulated alternatives which generate a total of 256 pairwise comparisons. Excluding the comparison of pairs composed by the same alternatives (the matrix diagonal) and considering that  $\mathcal{M}$  is anti-symmetric the resulting number of queries is given by 120. Among these 120 queries the dominance is applied considering the aim of optimization in each scenario and 84 queries are solved. Therefore the DM had to answer to only 36 among those 256 queries in each optimization problem. The resulting matrix was presented to the DM who had to choose the best alternative of each pair of simulated alternatives whose answer was not obtained by one of those described decision criteria. The decision-making matrices employed in estimating the DM's preferences in the polymer extrusion process are available in [Appendix C](#).

Once the NN-DM model is constructed it can be applied to the EPF of the considered scenario. For this purpose the MOOP's solutions are loaded and the model is employed in establishing a scalar value for each one. From this point forward the alternatives can be sorted from the best to the worst one according to the DM's preferences represented by the NN-DM model.

## 8.4 The Adapted NN-DM Methodology

Since a real DM is considered here it is assumed that his preferences can also be reproduced by a utility function  $\mathcal{U}$  and hence by a NN-DM model  $\hat{\mathcal{U}}$ . However, the absence of an underlying utility function demands adaptations in some steps of the NN-DM method to consider the available information. The interactions with the DM are also made in a different way: the DM has to fill the decision-making matrix regarding the unsolved queries (Section 8.3).

As presented in Chapter 5, the NN-DM method is divided into four steps. For this application the domain  $\mathcal{D}$  is previously provided by the DM. Thereby it is not necessary to establish the domain as the original **Step 1** has proposed. **Step 2** introduces the ranking of alternatives which is now built from a total sorting (the decision-making matrix). **Step 3** is unchanged, but additional changes are made in **Step 4** since the performance of the resulting model, assessed by the KTD in the original method, is now evaluated by the DM himself. The reported changes are better described next.

### 8.4.1 Step 1: Domain Establishment

In this application the DM provides the decision-making domain which is employed in establishing the domain  $\mathcal{D}$  of the model  $\hat{\mathcal{U}}$ . Into the domain  $\mathcal{D}$  a grid of simulated alternatives  $F$  is constructed to extract information about the DM's preferences. The grid is considered in an attempt to make the DM's answers easier. The equivalence, the dominance, and the symmetry are first considered here in taking the decision in situations in which an

answer is acquired without consulting the DM. The remaining queries are then presented to the DM as a matrix (the decision-making matrix  $\mathcal{M}$ ) relating each pair of simulated alternatives. The matrix  $\mathcal{M}$  captures the DM's preferences within the domain  $\mathcal{D}$  and the adapted NN-DM method is now able to proceed to the next step.

### 8.4.2 Step 2: Ranking Construction

The NN-DM method builds a partial ranking  $\mathcal{R}$  for the alternatives assigning a scalar value to each alternative. The partial ranking is an interactive procedure in which the DM has to be consulted to sort the pivots and then classify each remaining alternative through comparisons with the pivots. In an attempt to simplify the process to the real DM, the decision-making matrix  $\mathcal{M}$  is constructed and filled by the DM in his own time. Since the answers to all the queries are supplied by the matrix  $\mathcal{M}$  a total ranking  $\mathcal{R}$  is now available, but once more the alternatives are clustered as in INSPM for the convenience of the approximation technique.

### 8.4.3 Step 4: Performance Assessment

The NN-DM method relies on the KTD as an efficiency metric. The lists to be compared are generated by sorting the available alternatives according to the DM's utility function and the resulting NN-DM model. The KTD is an applicable metric because an underlying utility function is available to provide information about the quality of the resulting model.

As this chapter focus in a real DM there is no available utility function

which demands another validating process. The advantage is that here the process is validated by the DM himself. Once the model for the DM's preferences is constructed it is applied to sort the available data and the DM can verify the results accuracy.

#### 8.4.4 Algorithm

Algorithm 4 presents the adapted NN-DM method to the real scenario introduced in Section 8.1. A grid of alternatives is constructed in the domain  $\mathcal{D}$  provided by the DM. The decision-making matrix  $\mathcal{M}$  is filled by the answers provided by the DM related to the alternatives belonging to the grid. A total ranking  $\mathcal{R}$  is constructed and then clustered into  $\log n$  levels. The RBF network converts the ranking  $\mathcal{R}$  into a function  $\hat{\mathcal{U}}$  able to provide answers to alternatives belonging to the entire domain  $\mathcal{D}$ . The NN-DM model is now fit to be employed in the EPF solutions.

---

**Algorithm 4** Adapted NN-DM Method

---

- 1: Read the domain  $\mathcal{D}$
  - 2: Read the decision-making matrix  $\mathcal{M}$
  - 3: Built the total ranking of alternatives  $\mathcal{R}$
  - 4: Classify the alternatives into  $\log n$  levels
  - 5: Construct the RBF network  $\hat{\mathcal{U}}$
- 

Algorithm 5 introduces the NN-DM model applied to the polymer extrusion process. In each considered scenario, the EPF and the NN-DM model  $\hat{\mathcal{U}}$  are loaded. The model  $\hat{\mathcal{U}}$  is then employed in evaluating each solution generating a sorting of the solutions from the best to the worst one.

---

**Algorithm 5** NN-DM model applied to the polymer extrusion process

---

- 1: Load the estimates of the PF (EPFs)
  - 2: Load the NN-DM model  $\hat{\mathcal{U}}$
  - 3: Evaluate each available solution
  - 4: Sort the solutions from the best to the worst one
- 

## 8.5 Computational Experiments

The filled decision-making matrices provided by the DM are taken into account to construct general NN-DM models as described in Section 8.4. Figures 8.2 and 8.3 present respectively the models for the two considered scenarios:  $\mathbf{Q} \times \mathbf{P}$  and  $\mathbf{Q} \times \mathbf{W}$ . The models are trained in the provided domain  $\mathcal{D}$  (Table 8.2).

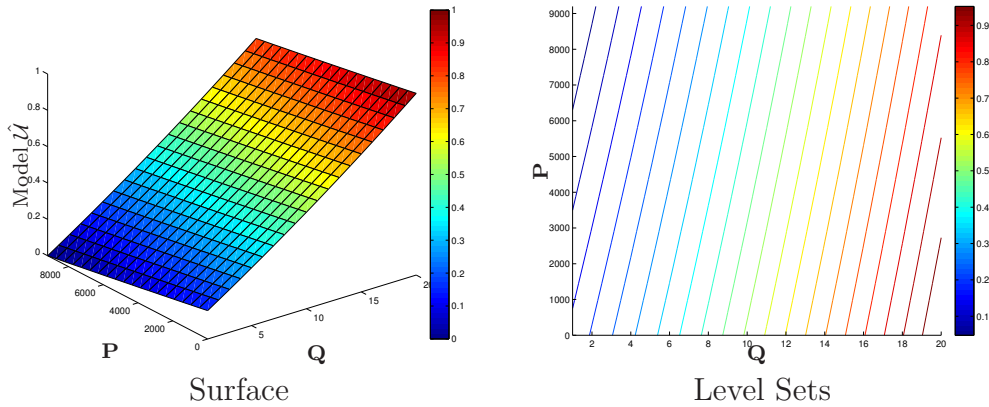


Figure 8.2: General NN-DM model:  $\mathbf{Q} \times \mathbf{P}$ .

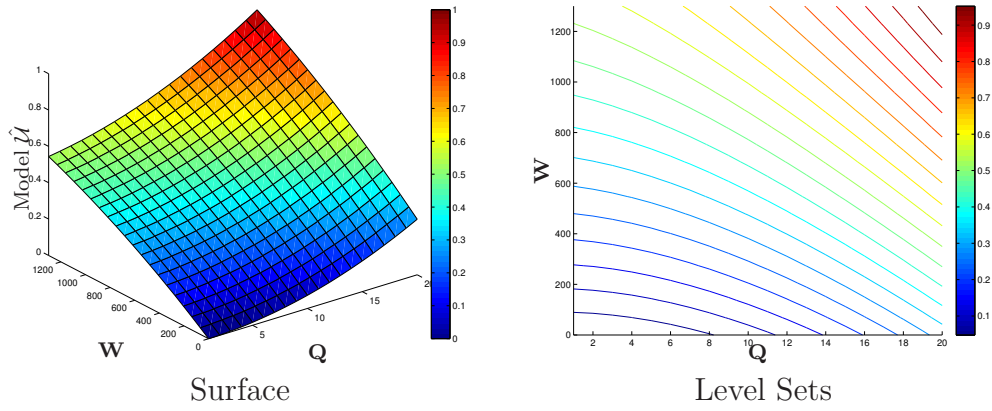


Figure 8.3: General NN-DM model:  $\mathbf{Q} \times \mathbf{W}$ .

Figures 8.4 and 8.5 present the general NN-DM models applied to sort the EPFs considering the objectives Mass Output and Power Consumption (EPFs  $\mathbf{QP}_1$ ,  $\mathbf{QP}_2$ , and  $\mathbf{QP}_3$ ) and Mass Output and WATS (EPFs  $\mathbf{QW}_1$ ,  $\mathbf{QW}_2$ , and  $\mathbf{QW}_3$ ). The models' level sets are illustrated in the figures and the DM's preferences are represented by the external scale.

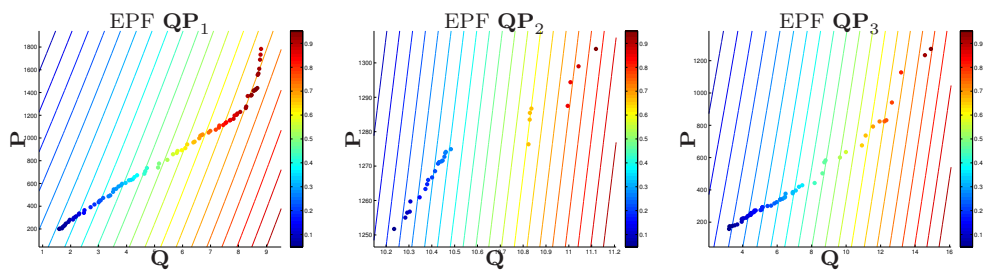


Figure 8.4: NN-DM model sorting in the problem  $\mathbf{Q} \times \mathbf{P}$ .

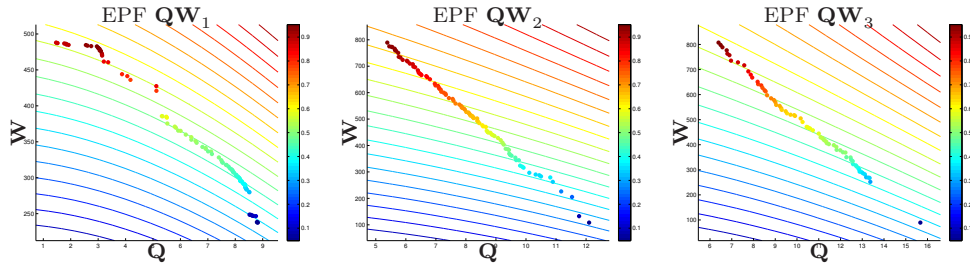


Figure 8.5: NN-DM model sorting in the problem  $\mathbf{Q} \times \mathbf{W}$ .

The DM's preferences, captured by the decision-making matrices, are now represented by the NN-DM models which are employed in sorting the alternatives belonging to the EPFs. Moreover, the resulting models are now available to represent the DM's preferences in any other situations without demanding further information from the DM.

## 8.6 Case Study

Consider now three decision-making matrices similar to the original matrix  $\mathcal{M}$  for the Mass Output  $\times$  WATS problem. This case study analyzes the effect in the resulting NN-DM models and reveals that small differences in the matrices can lead to similar models with different sorting and matrices with higher differences can lead to models with the same sorting.

The first difference between the original and the new matrices is the position  $[8, 434] \times [1, 1300]$  which now is filled with  $-1$ . The other differences belong to the block  $[8/9/10, 434] \times [1/8/14, 1300]$  which is filled as described in Table 8.6. The complete matrices are available in Appendix C.



Table 8.6: Sub-matrices with different preferences.

Matrix $\mathcal{M}_1$ (Original Matrix)			
$F_1 \times F_2$	[1, 1300]	[8, 1300]	[14, 1300]
[8, 867]	1	1	1
[14, 867]	1	1	1
[20, 867]	-1	-1	1

Matrix $\mathcal{M}_2$			
$F_1 \times F_2$	[1, 1300]	[8, 1300]	[14, 1300]
[8, 867]	-1	1	1
[14, 867]	-1	-1	1
[20, 867]	-1	-1	-1

Matrix $\mathcal{M}_3$			
$F_1 \times F_2$	[1, 1300]	[8, 1300]	[14, 1300]
[8, 867]	1	1	1
[14, 867]	-1	-1	1
[20, 867]	-1	-1	-1

Matrix $\mathcal{M}_4$			
$F_1 \times F_2$	[1, 1300]	[8, 1300]	[14, 1300]
[8, 867]	1	1	1
[14, 867]	1	1	1
[20, 867]	1	1	1

The matrices  $\mathcal{M}_1$ ,  $\mathcal{M}_2$ , and  $\mathcal{M}_3$  have been filled by the DM while the matrix  $\mathcal{M}_4$  has been filled by the author. The level sets of the resulting NN-DM models, called respectively models  $\hat{\mathcal{U}}_1$ ,  $\hat{\mathcal{U}}_2$ ,  $\hat{\mathcal{U}}_3$ , and  $\hat{\mathcal{U}}_4$ , are provided by Figure 8.6. The domain of the approximation is still given by Table 8.2 and the sorting of each EPF can be visualized by the external scale in each figure.

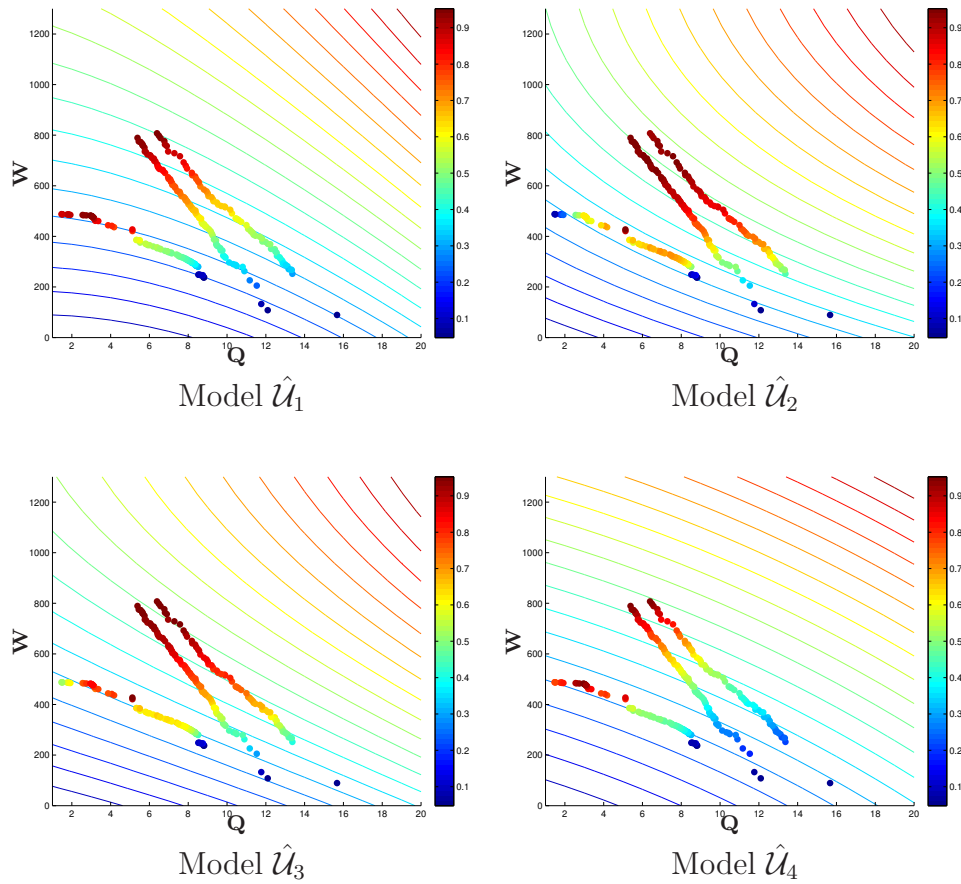


Figure 8.6: NN-DM models applied to the estimates of the Pareto-optimal front.

EPFs  $\mathbf{QW}_2$  and  $\mathbf{QW}_3$  received the same sorting provided by the NN-DM models corresponding to all the matrices, but EPF  $\mathbf{QW}_1$  showed variations in the sorting. The reason for this behavior relies on the similarities between the curvature of the level sets of the model and the curvature of the EPF. Since EPF  $\mathbf{QW}_1$  is almost parallel to the level sets of the resulting NN-DM model this EPF presents a situation in which the alternatives seem to be similar in the DM's viewpoint which makes the decision-making sensible to small changes.

Figure 8.7 presents the level sets of the four models and EPF  $\mathbf{QW}_1$  in its own domain which enhances the differences in the sorting. Four areas are highlighted in the figures to assist the analysis: A, B, C, and D. The sorting of these areas provided by the models are  $\hat{\mathcal{U}}_1: \langle B, A-C, D \rangle$ ;  $\hat{\mathcal{U}}_2: \langle C, B, A-D \rangle$ ;  $\hat{\mathcal{U}}_3: \langle C, B, A, D \rangle$ ; and  $\hat{\mathcal{U}}_4: \langle B, A-C, D \rangle$ . The regions connected by  $-$  indicate regions with similar preferences.

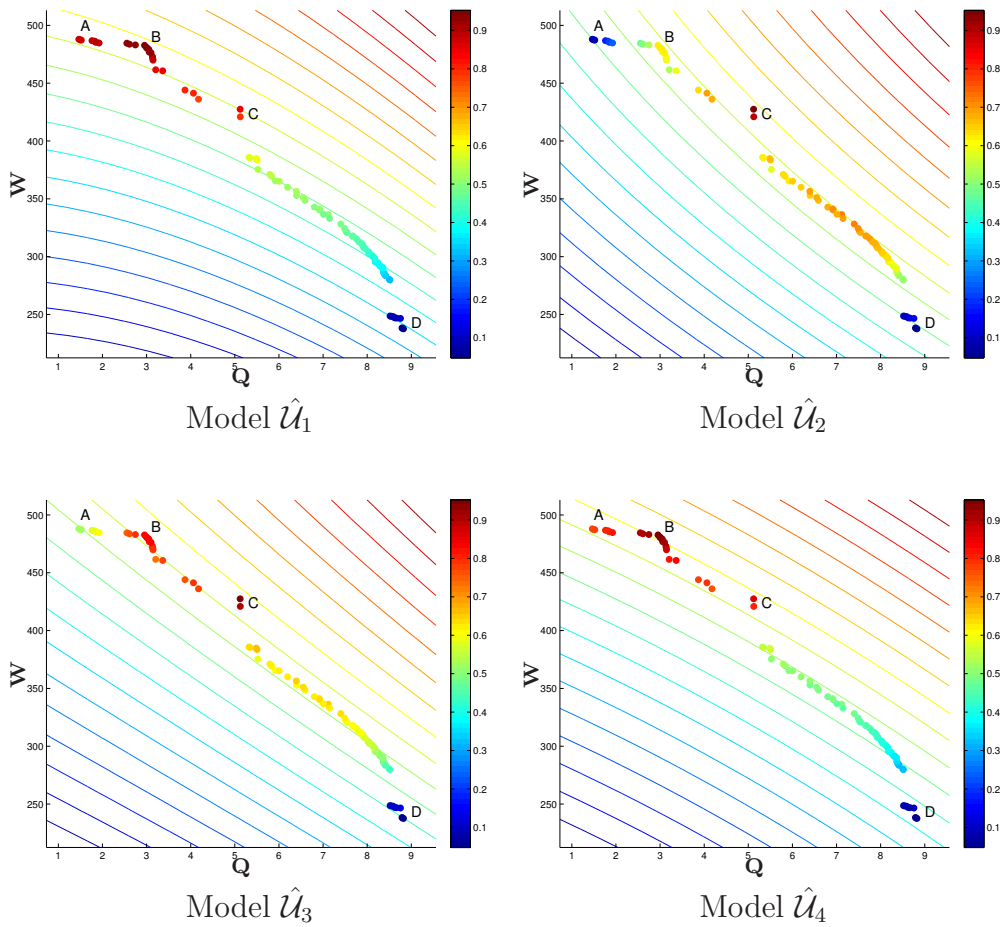


Figure 8.7: Comparison among models  $\hat{\mathcal{U}}_1$ ,  $\hat{\mathcal{U}}_2$ ,  $\hat{\mathcal{U}}_3$ , and  $\hat{\mathcal{U}}_4$  in EPF  $\mathbf{QW}_1$ .

Figure 8.7 shows that models  $\hat{\mathcal{U}}_2$  and  $\hat{\mathcal{U}}_3$  provide similar sorting; the same

happens with models  $\hat{\mathcal{U}}_1$  and  $\hat{\mathcal{U}}_4$ . Even matrices  $\mathcal{M}_2$  and  $\mathcal{M}_3$  possessing only one different entry their respective models quantify the region A differently. Meanwhile model  $\hat{\mathcal{U}}_4$  has been constructed with three different entries and possess almost the same preference distribution than model  $\hat{\mathcal{U}}_1$ . All the models are good in prediction the region D as the least favorite, but the most preferable region varies between B and C.

This analysis illustrates how different decision-making matrices affect the construction of different decision-making models. Small differences in the matrices can lead to similar models with different sorting and matrices with higher differences can lead to models with similar sorting. The model is extra-sensitive when the PF has its curvature almost parallel to the level sets of the model making slightly changes in the decision-making matrix generate models in which the preferable regions are different (EPF  $\mathbf{QW}_1$ ). However, similar models applied to EPFs  $\mathbf{QW}_2$  and  $\mathbf{QW}_3$  provide the same sorting, since the fronts and the level sets of the models are not similar, making the final model robust to small changes in the decision-making matrices.

In an attempt to obtain a robust model for EPF  $\mathbf{QW}_1$ , the DM was consulted once more to provide answers to queries concerning a new domain. Figure 8.8 presents EPF  $\mathbf{QW}_1$  and grids in two considered domains: the original domain provided by the objectives and the minimum domain which contains the alternatives in EPF  $\mathbf{QW}_1$ . The number of partitions in each dimension of the new grid is established as 3. After considering the automatic decision criteria (Section 8.3) the resulting matrix was presented to the DM who had to answer to only 9 new queries. The decision-making matrix is available in the supplementary material.

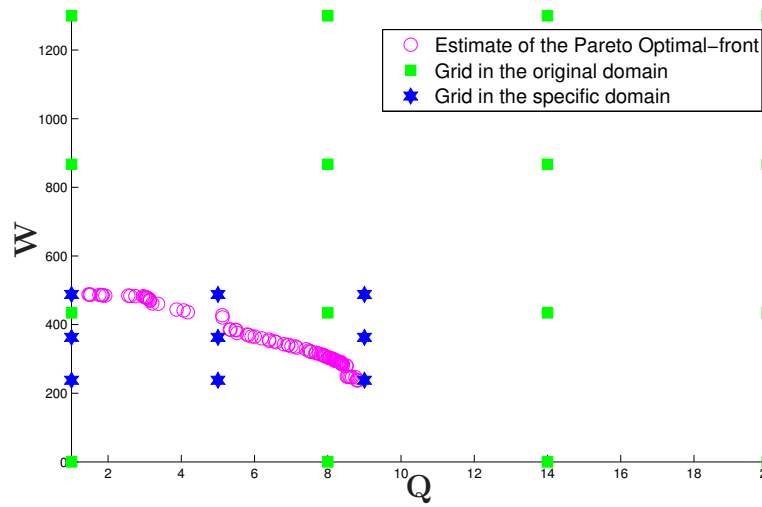


Figure 8.8: EPF  $QW_1$  embedded in two different domains.

Figure 8.9 shows the level sets of the resulting model obtained by the NN-DM method considering the domain of EPF  $QW_1$ . The level sets are now parallel to the  $x$ -axis leading to a robust model for the DM's preferences. Thereafter the alternatives can now be easily sorted from the best to the worst one.

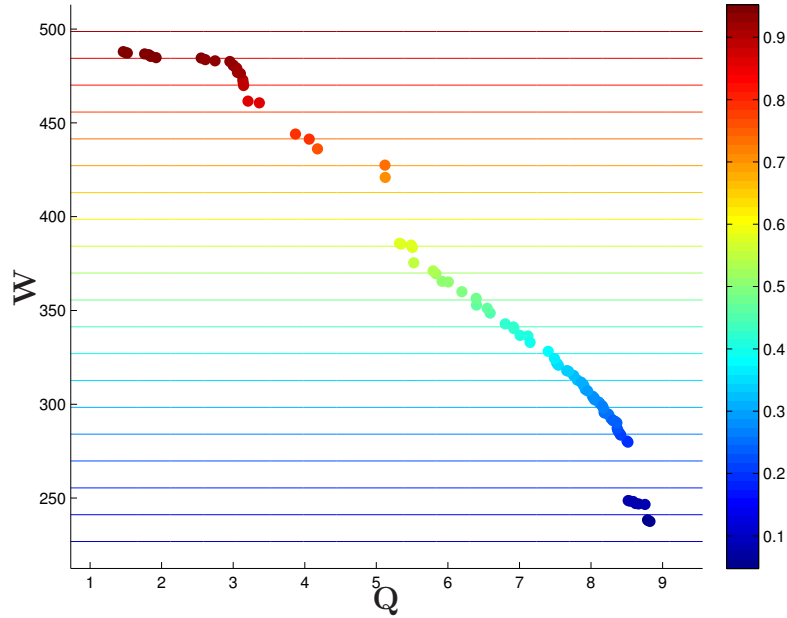


Figure 8.9: Level sets of the resulting NN-DM model constructed based on the domain of EPF  $\mathbf{QW}_1$ . The colorbar indicates the modeled DM's preferences.

## 8.7 Discussion

Adaptations in the NN-DM method have been executed to obtain a model for the DM's preferences in a real scenario considering a polymer extrusion process: the single screw extrusion. In the adapted NN-DM method the DM has to fill a decision-making matrix constructed by alternatives belonging to a grid in the domain  $\mathcal{D}$  provided by the DM. The adapted NN-DM method is able to construct an accurate NN-DM model  $\hat{\mathcal{U}}$  for the DM's preferences in each considered scenario. Once the model  $\hat{\mathcal{U}}$  is available it can be employed in quantifying any alternative according to the DM's preferences and sort them from the best to the worst one. Six EPFs derived from the polymer

extrusion process are sorted by the resulting NN-DM models. The model  $\hat{\mathcal{U}}$  can be repeatedly employed without further queries to the DM as long as the alternatives belong to the domain  $\mathcal{D}$ .

The results show that the NN-DM method is able to construct models that correspond to the DM's expectation in sorting the alternatives belonging to the considered EPFs. The DM's demand is reasonable and henceforth the resulting models  $\hat{\mathcal{U}}$  can replace the DM in recurrent decisions within the trained domain  $\mathcal{D}$ . Additionally, a case study reveals that the NN-DM method is robust to small variations in the decision-making matrices, leading to models which provide the same sorting (EPFs  $\mathbf{QW}_2$  and  $\mathbf{QW}_3$ ). However, EPF  $\mathbf{QW}_1$  presented different sorting, implying that the method is sensible to small changes in situations in which the EPF is almost parallel to the model's level sets.

The average melt temperature of the polymer at die exit ( $\mathbf{T}_{\text{melt}}$ ) and the length of screw required to melt the polymer ( $\mathbf{L}_{\text{melt}}$ ) also characterize the process performance and could have been considered in the multi-objective optimization problem. However, in a five-objective problem the decision-making matrix is inappropriate since it is difficult for a person to decide between two alternatives considering five conflicting objectives. Therefore, it is necessary a different approach to extract information from the DM. The author is studying improvements in the NN-DM method to consider a more complete polymer extrusion process. Additionally, since the optimization problem in this real scenario is computationally expensive, the NN-DM model may be employed in guiding the optimization process directly to the most preferable region avoiding computational effort expended in the non-preferable regions.

## Chapter 9

# Conclusions and Ideas for Future Work

The work developed in this thesis presented the construction of a model for the DM's preferences considering the existence of a utility function. In the NN-DM method the preference information is extracted from the DM involving ordinal description only and is structured considering a partial ranking procedure. An artificial neural network which approximates this partial ranking is constructed and the resulting model is able to reproduce the DM's preferences in a specific domain.

The proposed methodology is suitable in those situations in which a recurrent decision process must be performed, for instance several runs of a multi-objective optimization algorithm over the same problem with different parameters in each run, assuming that the utility function is not dependent on the changing parameters. Examples of such a situation are: (i) the choice of the operation point of an electric power system under different load constraints (intra-day or intra-week periods); (ii) the manufacturing of a compound which may be composed of different raw materials under different



relative prices of such materials; and (iii) the choice of routes, in any routing problem, under different situations of the costs associated to the problem links, or under situations of unavailability of certain links.

A characteristic of the NN-DM method is the ability to represent arbitrary dependencies among the decision criteria, including non-linear ones, in situations in which the DM should evaluate a solution as a whole, instead of weighting the criteria. The outcome of the proposed method has the purpose of being a representation of the DM's preference structure in a region of the feature space instead of being oriented to solve a specific decision problem.

The main point raised here is: the information obtained from the DM should not be discarded leading to a new complete interaction each time similar decision-making problems with the same DM are required. It is also straightforward to notice that in new situations it is possible to perform either a refinement of the NN-DM model or its validation through further interactions with the DM in new runs of the algorithm.

The interaction with the DM on the course of a multi-objective optimization process might provide good solutions in situations in which the complete PF cannot be achieved properly. By considering the DM guiding the optimization process the computational effort usually spent in searches inside non-preferable regions can now be employed to map the preferable regions accurately.

The methodology developed here was also employed in developing an EMO methodology called INSPM which combines two characteristics: (i) it interacts with the DM along the execution of the optimization task, such that the result of the optimization is guided by that interaction in a progressive

preference articulation, and (ii) it ends the optimization procedure with a model for the DM's preferences, which becomes available in other instances of the same problem. In the INSPM algorithm, the EPF is modeled densely according to the DM's preferences and the NN-DM model is constructed on the EPF domain. A parameter  $w$  controls the influence of the DM in the density of the EPF:  $w = 0$  means that there is no DM interference in the optimization process and the higher the  $w$ , the greater the interference, conducting to a scenario in which only the preferred solution and the extremes of the EPF are selected. In new instances of the same optimization problem, the resultant NN-DM model can be employed without new queries to the DM, except when the domain requires updates. In this situation new queries can be proposed to the DM at a low cost.

Some ideas for future work are presented next.

- Investigating a methodology inspired on a tournament procedure to improve the NN-DM method reducing the requirements to the DM.
- Analyzing further developments of the human-machine interaction applied to multi-objective optimization in the context of certain engineering problems.
- Employing the NN-DM method in the construction of a function  $P$  that works in methods such as Electre ou Promethee.
- Finding the ranking only with the EPF alternatives.
- Placing the alternatives as centers of the RBF functions of the artificial neural network.

- Finding a better cover for the space, since grid and random are already tested and are inefficient with higher dimension problems.

# Bibliography

- G. Aiello, G. La Scalia, and M. Enea. A non dominated ranking multi objective genetic algorithm and ELECTRE method for unequal area facility layout problems. *Expert Systems with Applications*, 40(12):4812–4819, 2013.
- S. Angilella, S. Greco, F. Lamantia, and B. Matarazzo. Assessing non-additive utility for multicriteria decision aid. *European Journal of Operational Research*, 158:734–744, 2004.
- S. Angilella, S. Greco, and B. Matarazzo. Non-additive robust ordinal regression: A multiple criteria decision model based on the Choquet integral. *European Journal of Operational Research*, 201:277–288, 2010.
- R. Battiti and A. Passerini. Brain-computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker. *IEEE Transactions on Evolutionary Computation*, 14(5):671–687, 2010.
- K. N. Berk. Comparing Subset Regression Procedures. *Technometrics*, 20(1):1–6, 1978.
- D. S. Broomhead and D. Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems 2*, pages 321–355, 1988.

- J. Chen and S. Lin. An interactive neural network-based approach for solving multiple criteria decision-making problems. *Decision Support Systems*, 36:137–146, 2003.
- S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, March 1991.
- D. H. Cope. *Computer Models of Musical Creativity*. The MIT Press, 2005.
- D. H. Cope. *Comes the Fiery Night: 2,000 Haiku by man and machine*. CreateSpace Independent Publishing Platform, Santa Cruz, CA, 2011.
- S. Corrente, S. Greco, M. Kadziński, and R. Słowiński. Robust ordinal regression in preference learning and ranking. *Machine Learning*, 93(2–3):381–422, 2013.
- K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *International Journal of Computational Intelligence Research*, pages 635–642. Springer-Verlag, 2006.
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
- K. Deb, A. Sinha, P. J. Korhonen, and J. Wallenius. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation*, 14(5):723–739, 2010.
- C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms

- in multiobjective optimization. *Evolutionary Computation*, 7(3):205–230, 1995.
- A. Gaspar-Cunha and J. A. Covas. RPSGAe–Reduced Pareto Set Genetic Algorithm: Application to Polymer Extrusion. In *Metaheuristics for Multiobjective Optimisation*, pages 221–249. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, Berlin Heidelberg, 2004.
- A. Gaspar-Cunha. *Modelling and Optimisation of Single Screw Extrusion Using Multi-Objective Evolutionary Algorithms*. Lambert Academic Publishing, Koln, Germany, 1st edition, 2009.
- D. Golmohammadi. Neural network application for fuzzy multi-criteria decision making problems. *International Journal of Production Economics*, 131(2):490–504, 2011.
- S. Greco, V. Mousseau, and R. Słowiński. Ordinal regression revisited: Multiple criteria ranking using a set of additive value functions. *European Journal of Operational Research*, 191(2):416–436, 2008.
- E. Jacquet-Lagrange and J. Siskos. Assessing a set of additive utility functions for multicriteria decision-making, the UTA method. *European Journal of Operational Research*, 10(2):151–164, 1982.
- I. Karahan and M. Köksalan. A territory defining multiobjective evolutionary algorithms and preference incorporation. *IEEE Transactions on Evolutionary Computation*, 14(4):636–664, 2010.
- I. Karahan. Preference-based flexible multiobjective evolutionary algorithms. Master’s thesis, Dept. Ind. Eng., Middle East Technical University, Ankara, Turkey, 2008.

- A. P. Karpenko, D. T. Mukhlisullina, and V. A. Ovchinnikov. Multicriteria optimization based on neural network approximation of decision maker's utility function. *Optical Memory and Neural Networks*, 19(3):227–236, 2010.
- A. P. Karpenko, D. A. Moor, and D. T. Mukhlisullina. Multicriteria optimization based on neural network, fuzzy and neuro-fuzzy approximation of decision maker's utility function. *Optical Memory and Neural Networks*, 21(1):1–10, 2012.
- R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.
- M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
- J. D. Knowles and D. W. Corne. M-PAES: a memetic algorithm for multiobjective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 325–332, 2000.
- D. E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.
- D. E. Knuth. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.
- T. Kohonen. *Self-organization and Associative Memory: 3rd Edition*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

- M. Köksalan and I. Karahan. An interactive territory defining evolutionary algorithm: iTDEA. *IEEE Transactions on Evolutionary Computation*, 14(5):702–722, 2010.
- D. C. Lay. *Linear Algebra and Its Applications*. Pearson Education, 2002.
- R. Lazimy. Interactive polyhedral outer approximation (IPOA) strategy for general multiobjective optimization problems. *Annals of Operations Research*, 210(1):73–99, 2013.
- B. Luo, J. Zheng, J. Xie, and J. Wu. Dynamic crowding distance - a new diversity maintenance strategy for MOEAs. In *Natural Computation, 2008. ICNC'08. Fourth International Conference, ICNC '08*, pages 580–585, Washington, DC, USA, 2008. IEEE Computer Society.
- J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
- The MathWorks. *MATLAB – Version 6.9.0 (R2009b)*. The MathWorks, Inc., Natick, Massachusetts, 2009.
- S. P. S. Matias. A multicriteria decision aiding assignment methodology for assisted reproductive technology. Master’s thesis, Technical University of Lisbon, Lisbon, Portugal, 2008.
- K. M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, USA, 1999.
- J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, June 1989.



- L. R. Pedro and R. H. C. Takahashi. Modeling the decision-maker utility function through artificial neural networks. In *Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)*, volume 1, pages 550–563, Ouro Preto, Brasil, 2009.
- L. R. Pedro and R. H. C. Takahashi. Modeling decision-maker preferences through utility function level sets. In *6th International Conference on Evolutionary Multi-criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 550–563. Springer Berlin Heidelberg, Ouro Preto, Brasil, 2011.
- L. R. Pedro and R. H. C. Takahashi. Decision-maker preference modeling in interactive multiobjective optimization. In *7th International Conference on Evolutionary Multi-criterion Optimization*, volume 7811 of *Lecture Notes in Computer Science*, pages 811–824. Springer Berlin Heidelberg, Sheffield, UK, 2013.
- L. R. Pedro and R. H. C. Takahashi. INSPM: An interactive evolutionary multi-objective algorithm with preference model. *Information Sciences*, 268(0):202–219, 2014.
- L. A. D. Rangel, L. F. A. M. Gomes, and R. A. Moreira. Decision theory with multiple criteria: an application of ELECTRE IV and TODIM to SEBRAE/RJ. *Pesquisa Operacional*, 29:577–590, 2009.
- N. Roussat, C. Dujet, and J. Méhu. Choosing a sustainable demolition waste management strategy using multicriteria decision analysis. *Waste Management*, 29(1):12–20, 2009.
- B. Roy. Classement et choix en présence de points de vue multiples: La

- méthode ELECTRE. *Revue Francaise d'Informatique et de Recherche Opérationnelle*, 8:57–75, 1968.
- T. L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15:234–281, 1977.
- T. L. Saaty. Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, 1:83–98, 2008.
- D. F. Specht. Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. *IEEE Transactions on Neural Networks and Learning Systems*, 1(1):111–121, March 1990.
- C. Steiner. *Automate this: how algorithms came to rule our world*. Portfolio/Penguin, New York, 2012.
- R. E. Steuer and E. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(1):326–344, 1983.
- R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York, 1986.
- M. Sun, A. Stam, and R. E. Steuer. Solving multiple objective programming problems using feed-forward artificial neural networks: The interactive FFANN procedure. *Management Science*, 42:835–849, 1996.
- M. Sun, A. Stam, and R. E. Steuer. Interactive multiple objective programming using Tchebycheff programs and artificial neural networks. *Computers and Operations Research*, 27(7–8):601–620, 2000.

- A. S. Tangian. Constructing a quasi-concave quadratic objective function from interviewing a decision maker. *European Journal of Operational Research*, 141(3):608–640, 2002.
- T. Tervonen, I. Linkov, J. R. Figueira, J. Steevens, M. Chappell, and M. Merad. Risk-based classification system of nanomaterials. *Journal of Nanoparticle Research*, 11(4):757–766, 2009.
- D. S. Todd and P. Sen. Directed multiple objective search of design spaces using genetic algorithms and neural networks. In *Genetic and Evolutionary Computation Conference*, pages 1738–1743. Morgan Kaufmann, 1999.
- E. Triantaphyllou and S. H. Mann. An examination of the effectiveness of multi-dimensional decision-making methods: a decision-making paradox. *Decision Support Systems*, 5(3):303–312, 1989.
- B. Widrow and M. E. Hoff. Neurocomputing: Foundations of research. In *Neurocomputing: Foundations of Research*, chapter Adaptive Switching Circuits, pages 123–134. MIT Press, Cambridge, MA, USA, 1988.
- J. B. Yang and P. Sen. Preference modelling by estimating local utility functions for multiobjective optimization. *European Journal of Operational Research*, 95(1):115–138, 1996.
- E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.
- E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA 2: Improving the Strength Pareto Evolutionary Algorithms. In *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2002.

# Appendix A

## The NEWRB Function

This appendix reproduces the information about the NEWRB function, as described in MATLAB<sup>®</sup> documentation. The related MATLAB<sup>®</sup> functions and the function parameters are designed in capital letters.

### A.1 Definition

The NEWRB function designs a radial basis network. The call for the NEWRB function is  $[\text{net}, \text{tr}] = \text{NEWRB}(\text{P}, \text{T}, \text{GOAL}, \text{SPREAD}, \text{MN}, \text{DF})$ .

### A.2 Description

Radial basis networks can be used to approximate functions. The NEWRB function adds neurons to the hidden layer of a radial basis network until it meets the specified mean squared error goal. The NEWRB function takes the arguments introduced by Table [A.1](#) and returns a new radial basis network.

Parameter	Description
P	RxQ matrix of Q input vectors.
T	SxQ matrix of Q target class vectors.
GOAL	Mean squared error goal, default = 0.0.
SPREAD	Spread of radial basis functions, default = 1.0.
MN	Maximum number of neurons, default is Q.
DF	Number of neurons to add between displays, default = 25.

Table A.1: Parameters of the NEWRB function.

The larger that SPREAD the smoother the function approximation; a spread too large means a lot of neurons is required to fit a fast changing function. A spread too small means many neurons is required to fit a smooth function and the network may not generalize well.

### A.3 Algorithm

The NEWRB function creates a two layer network. The first layer has RADBAS<sup>1</sup> neurons, calculates its weighted inputs with DIST<sup>2</sup>, and its net input with NETPROD<sup>3</sup>. The second layer has PURELIN<sup>4</sup> neurons, calculates its weighted input with DOTPROD<sup>5</sup>, and its net inputs with NETSUM<sup>6</sup>. Both layers have biases.

<sup>1</sup>RADBAS is a neural transfer function. Transfer functions calculate a layer's output from its net input.

<sup>2</sup>DIST is the Euclidean distance weight function. Weight functions apply weights to an input to get weighted inputs.

<sup>3</sup>NETPROD is a net input function. Net input functions calculate a layer's net input by combining its weighted inputs and biases.

<sup>4</sup>PURELIN is a neural transfer function. Transfer functions calculate a layer's output from its net input.

<sup>5</sup>DOTPROD is the dot product weight function. Weight functions apply weights to an input to get weighted inputs.

<sup>6</sup>NETSUM is a net input function. Net input functions calculate a layer's net input by combining its weighted inputs and biases.

Initially the RADBAS layer has no neurons. The following steps are repeated until the network's mean squared error falls below GOAL or the maximum number of neurons are reached.

1. The network is simulated.
2. The input vector with the greatest error is found.
3. A RADBAS neuron is added with weights equal to that vector.
4. The PURELIN layer weights are redesigned to minimize error.

## A.4 Simulating the Network

The SIM function simulates neural networks. The call for the SIM function is  $[Y, X_f, A_f] = \text{sim}(\text{net}, X, X_i, A_i, T)$ , with inputs and outputs described by Tables A.2 and A.3, respectively.

Parameter	Description
net	Network.
X	Network inputs.
X <sub>i</sub>	Initial layer delay conditions (default = zeros).
A <sub>i</sub>	Initial layer delay conditions (default = zeros).
T	Network targets (default = zeros).

Table A.2: Inputs of the SIM function.

Parameter	Description
Y	Network outputs.
Xf	Final input delay conditions.
Af	Final layer delay conditions.

Table A.3: Outputs of the SIM function.



# Appendix B

## A Comparison Between Mergesort and Quicksort

The comparison of sorting algorithms has the computation time and the memory requirement as the main merit factors in most of the situations. However, here the sorting algorithms have the **number of element comparisons** as the only relevant merit factor. This occurs because the “limited resource” that should be saved now is the number of queries to the Decision-Maker (DM). The answers to those queries are employed in solving the comparisons between the elements of the set to be sorted. Since the DM is usually a human being the number of queries hardly could reach the order of some hundreds making any asymptotic analysis irrelevant. Due to the specificity of the requirements on the sorting algorithms for the present work an analysis of those algorithms in the relevant framework is presented in this appendix.

The DM’s ability to answer ordinal queries is translated inside of sorting algorithms as the number of comparisons realized between the elements in the set. In essence the DM is asked to provide an answer to solve each comparison

required by the sorting algorithm. In this work Mergesort was chosen against Quicksort as the sorting algorithm due to its smaller number of comparisons. The main reason behind this fact seems to be the uneven splitting at each Quicksort step. This appendix presents a comparison between Mergesort and Quicksort performances considering both time and number of comparisons.

## B.1 Algorithms

### B.1.1 Quicksort

Quicksort is a sorting algorithm developed by Tony Hoare in 1960 [Knuth, 1997]. Quicksort is a divide and conquer algorithm that first selects a pivot and divides a large list into two smaller sub-lists (the low and the high elements) which are recursively sorted. The base of the recursion are lists of size zero or one which never needs to be sorted. The steps are the following.

1. Select an element, called pivot, from the list.
2. Reorder the list so that all elements with values lower than the pivot come before the pivot while all elements with values higher than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.
3. Recursively apply the above steps to the sub-list of elements with smaller values and to the sub-list of elements with higher values.

Quicksort, on average, makes  $\mathcal{O}(n \log n)$  comparisons to sort  $n$  items. In the worst-case, it makes  $\mathcal{O}(n^2)$  comparisons. Quicksort takes  $\mathcal{O}(n \log n)$  time on average, when the input is a random permutation, and  $\mathcal{O}(n^2)$  in the worst-case.

### B.1.2 Mergesort

Mergesort is a divide and conquer algorithm invented by John von Neumann in 1945 [Knuth, 1997]. Conceptually, Mergesort works as follows.

1. Divide the unsorted list into  $n$  sub-lists each containing 1 element (a list of 1 element is considered sorted).
2. Repeatedly merge sub-lists to produce new sub-lists until there is only 1 sub-list remaining. This will be the sorted list.

Mergesort is an  $\mathcal{O}(n \log n)$  comparison-based sorting algorithm. In the worst-case, Mergesort does about 39% fewer comparisons than Quicksort in the average-case. Mergesort's performance has an average and worst-case performance of  $\mathcal{O}(n \log n)$  time. The drawback of Mergesort is that most implementations must be done with  $2n$  space whereas Quicksort can be done in-place (which is not a concern in the proposed application). The additional  $n$  locations are commonly required because merging two sorted sets in place is more complicated and would need additional comparisons and move operations.

## B.2 Results

A comparison between Quicksort and Mergesort is now presented. Figure B.1 shows the obtained results displaying on  $y$ -axis (a) time and (b) number of comparisons versus the list size on  $x$ -axis. The  $x$ -axis represents the number of elements in the set which is not uniform; otherwise the behavior would not be clear enough for the first entries. The list size is up to 10,000 and the results are average values taken in 50 executions.

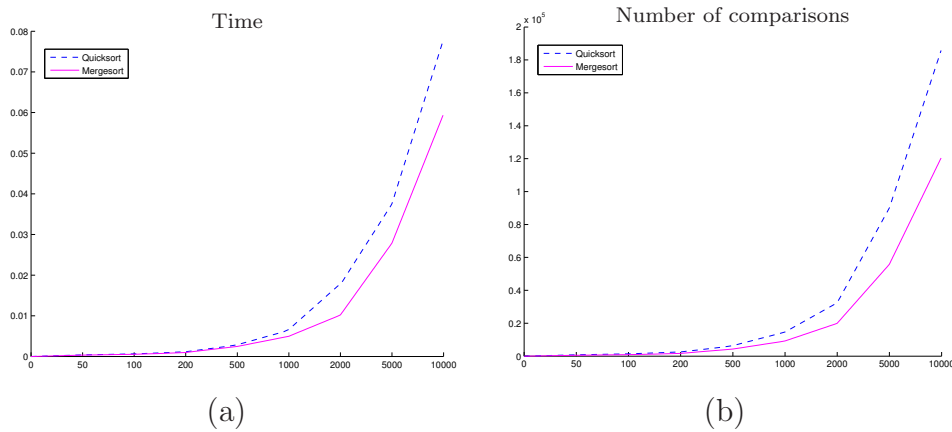


Figure B.1: Average number of comparisons considering Quicksort and Mergesort.

Figure B.2 presents the average number of comparisons considering four sorting algorithms: Bubble Sort, Insertion Sort, Quicksort, and Mergesort. Since the number of queries presented to the DM should be small the list size is now up to 200. The results are average values taken in 50 executions. Table B.1 reproduces the values of Figure B.2.

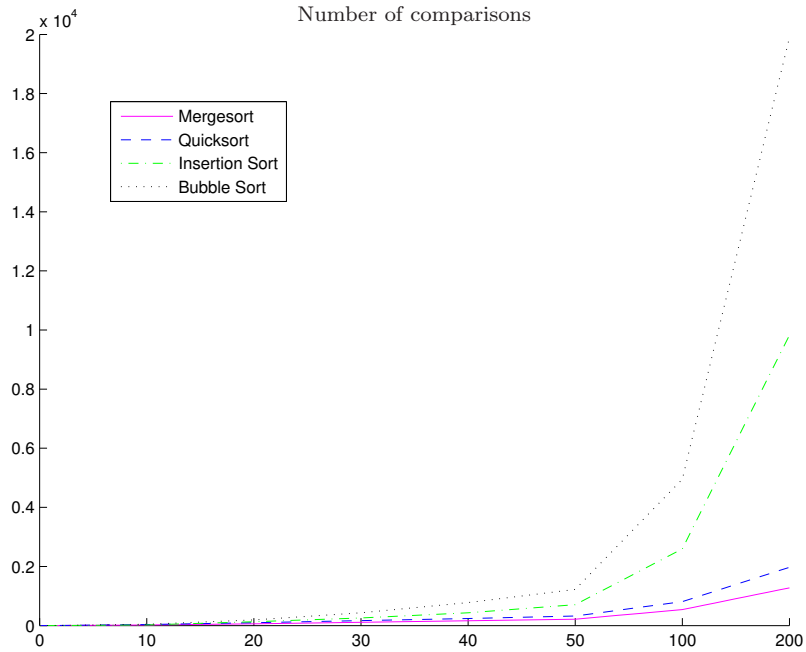


Figure B.2: Average number of comparisons considering four sorting algorithms.

List Size	Bubble Sort	Insertion Sort	Quicksort	Mergesort
10	45	22	34	22
20	190	129	95	61
30	435	256	169	112
40	780	431	238	164
50	1,225	706	323	217
100	4,950	2,595	810	540
200	19,900	9,826	1,973	1,277

Table B.1: Average number of comparisons spent for sorting a list.

Table B.1 shows that Insertion Sort is suitable if the number of elements is 10. Since the  $n_{in}$  parameter has assumed 10 in all tests, the Insertion Sort, which has an easier implementation, could have been chosen. However,

considering that  $n_{in}$  is a parameter of the INSPM algorithm and the difference between the number of comparisons spent by Mergesort and Insertion Sort for higher values is significant, Mergesort was chosen to be the sorting algorithm inside INSPM.

## Appendix C

### Decision-making Matrices - Polymer Extrusion Process

Table C.1: Unfilled decision-making matrix:  $\mathbf{Q} \times \mathbf{P}$ .

$\mathbf{Q} \times \mathbf{W}$	$[f_{10}, f_{20}]$	$[f_{11}, f_{20}]$	$[f_{12}, f_{20}]$	$[f_{13}, f_{20}]$	$[f_{10}, f_{21}]$	$[f_{11}, f_{21}]$	$[f_{12}, f_{21}]$	$[f_{13}, f_{21}]$	$[f_{10}, f_{22}]$	$[f_{11}, f_{22}]$	$[f_{12}, f_{22}]$	$[f_{13}, f_{22}]$	$[f_{10}, f_{23}]$	$[f_{11}, f_{23}]$	$[f_{12}, f_{23}]$	$[f_{13}, f_{23}]$
$[f_{10}, f_{20}]$	0	1	1	1	-1	X	X	X	-1	X	X	X	-1	X	X	X
$[f_{11}, f_{20}]$	-1	0	1	1	-1	-1	X	X	-1	-1	X	X	-1	-1	X	X
$[f_{12}, f_{20}]$	-1	-1	0	1	-1	-1	-1	X	-1	-1	-1	X	-1	-1	-1	X
$[f_{13}, f_{20}]$	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
$[f_{10}, f_{21}]$	1	1	1	1	0	1	1	1	-1	X	X	X	-1	X	X	X
$[f_{11}, f_{21}]$	X	1	1	1	-1	0	1	1	-1	-1	X	X	-1	-1	X	X
$[f_{12}, f_{21}]$	X	X	1	1	-1	-1	0	1	-1	-1	-1	X	-1	-1	-1	X
$[f_{13}, f_{21}]$	X	X	X	1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
$[f_{10}, f_{22}]$	1	1	1	1	1	1	1	1	0	1	1	1	-1	X	X	X
$[f_{11}, f_{22}]$	X	1	1	1	X	1	1	1	-1	0	1	1	-1	-1	X	X
$[f_{12}, f_{22}]$	X	X	1	1	X	X	1	1	-1	-1	0	1	-1	-1	-1	X
$[f_{13}, f_{22}]$	X	X	X	1	X	X	X	1	-1	-1	-1	0	-1	-1	-1	-1
$[f_{10}, f_{23}]$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
$[f_{11}, f_{23}]$	X	1	1	1	X	1	1	1	X	1	1	1	-1	0	1	1
$[f_{12}, f_{23}]$	X	X	1	1	X	X	1	1	X	X	1	1	-1	-1	0	1
$[f_{13}, f_{23}]$	X	X	X	1	X	X	X	1	X	X	X	1	-1	-1	-1	0



Table C.2: Filled decision-making matrix:  $\mathbf{Q} \times \mathbf{P}$ .

$\mathbf{Q} \times \mathbf{W}$	$[f_{10}, f_{20}]$	$[f_{11}, f_{20}]$	$[f_{12}, f_{20}]$	$[f_{13}, f_{20}]$	$[f_{10}, f_{21}]$	$[f_{11}, f_{21}]$	$[f_{12}, f_{21}]$	$[f_{13}, f_{21}]$	$[f_{10}, f_{22}]$	$[f_{11}, f_{22}]$	$[f_{12}, f_{22}]$	$[f_{13}, f_{22}]$	$[f_{10}, f_{23}]$	$[f_{11}, f_{23}]$	$[f_{12}, f_{23}]$	$[f_{13}, f_{23}]$
$[f_{10}, f_{20}]$	0	1	1	1	-1	1	1	1	-1	1	1	1	-1	-1	-1	1
$[f_{11}, f_{20}]$	-1	0	1	1	-1	-1	1	1	-1	-1	-1	1	-1	-1	-1	-1
$[f_{12}, f_{20}]$	-1	-1	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
$[f_{13}, f_{20}]$	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
$[f_{10}, f_{21}]$	1	1	1	1	0	1	1	1	-1	1	1	1	-1	-1	-1	1
$[f_{11}, f_{21}]$	-1	1	1	1	-1	0	1	1	-1	-1	1	1	-1	-1	-1	-1
$[f_{12}, f_{21}]$	-1	-1	1	1	-1	-1	0	1	-1	-1	-1	1	-1	-1	-1	-1
$[f_{13}, f_{21}]$	-1	-1	1	1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
$[f_{10}, f_{22}]$	1	1	1	1	1	1	1	1	0	1	1	1	-1	-1	1	1
$[f_{11}, f_{22}]$	-1	1	1	1	-1	1	1	1	-1	0	1	1	-1	-1	1	1
$[f_{12}, f_{22}]$	-1	1	1	1	-1	-1	1	1	-1	-1	0	1	-1	-1	-1	1
$[f_{13}, f_{22}]$	-1	-1	1	1	-1	-1	-1	1	-1	-1	-1	0	-1	-1	-1	-1
$[f_{10}, f_{23}]$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
$[f_{11}, f_{23}]$	1	1	1	1	1	1	1	1	1	1	1	1	-1	0	1	1
$[f_{12}, f_{23}]$	1	1	1	1	1	1	1	1	-1	-1	1	1	-1	-1	0	1
$[f_{13}, f_{23}]$	-1	1	1	1	-1	1	1	1	-1	-1	-1	1	-1	-1	-1	0









Table C.7: Filled decision-making matrix:  $\mathbf{Q} \times \mathbf{W}$  – Matrix  $M_4$ .

$\mathbf{Q} \times \mathbf{W}$	$[f_{10}, f_{20}]$	$[f_{11}, f_{20}]$	$[f_{12}, f_{20}]$	$[f_{13}, f_{20}]$	$[f_{10}, f_{21}]$	$[f_{11}, f_{21}]$	$[f_{12}, f_{21}]$	$[f_{13}, f_{21}]$	$[f_{10}, f_{22}]$	$[f_{11}, f_{22}]$	$[f_{12}, f_{22}]$	$[f_{13}, f_{22}]$	$[f_{10}, f_{23}]$	$[f_{11}, f_{23}]$	$[f_{12}, f_{23}]$	$[f_{13}, f_{23}]$
$[f_{10}, f_{20}]$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$[f_{11}, f_{20}]$	-1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$[f_{12}, f_{20}]$	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
$[f_{13}, f_{20}]$	-1	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1	1
$[f_{10}, f_{21}]$	-1	-1	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1
$[f_{11}, f_{21}]$	-1	-1	-1	-1	-1	0	1	1	-1	1	1	1	-1	1	1	1
$[f_{12}, f_{21}]$	-1	-1	-1	-1	-1	-1	0	1	-1	-1	1	1	-1	-1	1	1
$[f_{13}, f_{21}]$	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	1	-1	-1	-1	1
$[f_{10}, f_{22}]$	-1	-1	-1	-1	-1	1	1	1	0	1	1	1	1	1	1	1
$[f_{11}, f_{22}]$	-1	-1	-1	-1	-1	-1	1	1	-1	0	1	1	1	1	1	1
$[f_{12}, f_{22}]$	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	0	1	1	1	1	1
$[f_{13}, f_{22}]$	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	1	1	1	1
$[f_{10}, f_{23}]$	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	1	0	1	1	1
$[f_{11}, f_{23}]$	-1	-1	-1	-1	-1	-1	1	1	-1	-1	-1	1	-1	0	1	1
$[f_{12}, f_{23}]$	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	0	1
$[f_{13}, f_{23}]$	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0

# Index

- Adaptive Neuro-Fuzzy Inference System, 19
- AHP, 12, 18
- ALENA, 4
- algorithm, 49, 60, 99, 121
- artificial neural network, 15, 39
- AWTP, 16
- BC-EMO, 21
- Choquet integral, 13
- crowding distance, 92
- decision-maker, 24
- decision-making, 1, 11, 23
- decision-making matrix, 115, 120, 124
- DM calls, 48, 59
- DNN, 17
- domain establishment, 34, 56, 87, 119
- dominance, 27, 57, 64, 115, 116
- Downhill Simplex Method, 19
- dynamic crowding distance, 92
- ELECTRE, 11
- Emily Howell, 4
- evolutionary multi-objective optimization, 69, 84
- FFANN, 15, 16, 18
- Fuzzy Inference System, 19
- grid of simulated alternatives, 35, 54, 119
- holistic judgments, 3, 13, 21
- INSPM, 30, 85
- interactive algorithms, 70, 85
- IPOA, 21
- iTDEA, 71, 75, 106
- ITWP, 16
- KTD, 46, 66, 79, 91, 105
- ktd, 96, 98
- linguistic function, 18
- MATLAB<sup>®</sup>, 6, 43, 79
- MAUT, 2, 33
- MCDM, 2, 23

MCDM applications, 12  
MCDS system, 18  
Mergesort, 58, 87, 95, 151  
MLP network, 17, 19  
multi-objective optimization, 4, 71, 78, 80  
NN-DCD, 93, 102  
NN-DM method, 6, 30, 33, 55, 72, 86, 95  
NN-DM method examples, 50, 61  
NSGA-II, 30, 91, 97  
number of queries, 60, 66, 79, 105  
partial ranking examples, 51, 53, 62, 65, 88  
performance assessment, 46, 91, 120  
PI-EMO-VF, 20  
pivot alternative, 32, 37, 58  
polymer extrusion process, 112  
preference function, 24  
prTDEA, 74  
Quicksort, 88, 150  
R-NSGA-II, 70  
ranking construction, 37, 57, 87, 120  
RBF network, 19, 39, 51, 63, 65, 67  
reactive search optimization, 20  
real DM, 119  
real scenario, 112  
recurrent problem, 4, 132  
regression problem, 3, 5, 33, 39  
ROR, 13  
RPSGAe, 113  
simulated decision-making problem, 29, 35  
single screw extrusion, 112  
stable, 86  
stable model, 48, 52, 67  
Tchebycheff, 16, 74, 106  
TDEA, 73  
underlying utility function, 50, 61, 64, 78, 100  
utility function, 2, 24, 33  
validation sets, 48  
ZDT4, 101