

UNIVERSIDADE FEDERAL DO PARANÁ

THIAGO DO NASCIMENTO FERREIRA

A PREFERENCE-BASED APPROACH FOR REDUCING THE NUMBER OF OBJECTIVES  
APPLIED TO THE VARIABILITY TESTING OF SOFTWARE PRODUCT LINE

CURITIBA PR

2019

THIAGO DO NASCIMENTO FERREIRA

A PREFERENCE-BASED APPROACH FOR REDUCING THE NUMBER OF OBJECTIVES  
APPLIED TO THE VARIABILITY TESTING OF SOFTWARE PRODUCT LINE

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Silvia Regina Vergilio.

Coorientador: Marouane Kessentini.

CURITIBA PR

2019

Catálogo na Fonte: Sistema de Bibliotecas, UFPR  
Biblioteca de Ciência e Tecnologia

F383p

Ferreira, Thiago do Nascimento

A preference-based approach for reducing the number of objectives applied to the variability testing of software product line [recurso eletrônico] / Thiago do Nascimento Ferreira. – Curitiba, 2019.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2019.

Orientador: Silvia Regina Vergilio – Coorientador: Marouane Kessentini.

1. Software – Desenvolvimento. 2. Engenharia de software. 3. Algoritmos de computador. I. Universidade Federal do Paraná. II. Vergilio, Silvia Regina. III. Kessentini, Marouane. IV .Título.

CDD: 005.1

Bibliotecário: Elias Barbosa da Silva CRB-9/1894



MINISTÉRIO DA EDUCAÇÃO  
SETOR DE CIÊNCIAS EXATAS  
UNIVERSIDADE FEDERAL DO PARANÁ  
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO  
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA -  
40001016034P5

## TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **THIAGO DO NASCIMENTO FERREIRA** intitulada: **A PREFERENCE-BASED APPROACH FOR REDUCING THE NUMBER OF OBJECTIVES APPLIED TO THE VARIABILITY TESTING OF SOFTWARE PRODUCT LINE**, sob orientação da Profa. Dra. SILVIA REGINA VERGILIO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa. A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 07 de Novembro de 2019.

SILVIA REGINA VERGILIO  
Presidente da Banca Examinadora

MARIA CLAUDIA FIGUEIREDO PEREIRA EMER  
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO  
PARANÁ)

MARCIO DE OLIVEIRA BARROS  
Avaliador Externo (UNIVERSIDADE FEDERAL DO ESTADO DO RIO  
DE JANEIRO)

AURORA TRINIDAD RAMIREZ POZO  
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MAROUANE KESSENTINI  
Coorientador (UNIVERSITY OF MICHIGAN-DEARBORN)

JERFFESON TEIXEIRA DE SOUZA  
Avaliador Externo (UNIVERSIDADE ESTADUAL DO CEARÁ)



*To my wife Rebeca and my parents*

## ACKNOWLEDGEMENTS

I would like to sincerely express my gratitude:

- To my beloved wife, Rebeca, for standing by my side along the journey, and for all her sacrifice, patience, understanding, and love. Thanks for supporting me through the most challenging times;
- To my family, specially my parents, Bernardino e Cleide, and my sister, Thais, for all the support provided. Thank you for always believing in me and for always being by my side. Simply, thank you for existing;
- To my advisor, Dr. Silvia R. Vergilio for her excellent guidance, patience, advice, dedication, and availability. Thanks for the relevant comments, and corrections that led to the success of this work, and, mainly, for contributing to my research career;
- To my co-advisor, Dr. Marouane Kessentini, for agreeing to supervise me during my studies in Dearborn, USA, as a visiting student. You brought valuable contributions to my research;
- To all the committee members, Dr. Jerffeson T. de Souza, Dr. Maria Claudia F. P. Emer, Dr. Márcio de O. Barros, and Dr. Aurora T. R. Pozo, for the careful reading and the valuable suggestions;
- To the colleagues of Bio-inspired Computation Group (C-Bio) and Research Group on Software Engineering (GrES) for the rich exchange of ideas, friendship, stimulating discussions, and the great time we had together;
- To my friends, Micael and Kwanna, for great affection, friendship, provided support, and travels, and Thainá, for the enthusiasm, conversations, constant motivation, and the support during my sandwich PhD in the USA;
- To my Michigan friends, Wang, Hussein, Soumaya, and Terry for the valuable friendship. Specially, Lynn who hosted me in her house in Dearborn, and made me feel at home;
- To all staff of Federal University of Paraná (UFPR), specially its Graduate Programme in Informatics, for the acceptance of my PhD proposal, and provided courses;
- To Federal University of Technology - Paraná (UTFPR), Campus Curitiba, for the opportunity to be a substitute professor, and all my students who taught me more than I could ever teach them. Special thanks for Professor Rita, for the daily coffee that often led me to think of several relevant topics about the academic life;
- To the Brazilian Coordination for the Improvement of Higher Education (CAPES) for the scholarship and financial support;
- To all those people who, in one way or another, contributed to the completion of this dissertation. I extend to all my grateful thanks.

## RESUMO

Algoritmos evolutivos para múltiplos e muitos objetivos têm sido aplicados para selecionar produtos para o teste de variabilidade de Linhas de Produtos de Software (LPS). Esse problema refere-se à seleção de um conjunto adequado de produtos para testar uma LPS, pois testar todos os existentes é inviável. O problema é impactado por muitos fatores, como número de produtos a serem testados, critérios de cobertura a serem satisfeitos e eficácia em revelar defeitos. É possível notar que muitas funções objetivo conflitantes precisam ser otimizadas ao mesmo tempo. No entanto, alguns problemas surgem quando o número de objetivos a serem otimizados aumenta, por exemplo, as soluções geradas pelos algoritmos de otimização se tornam incomparáveis, projetar uma frente de Pareto requer um grande número de soluções e a visualização de tais soluções exige técnicas especiais. Várias técnicas são propostas na literatura para resolver esse problema, como técnicas de decomposição e algoritmos baseados em indicadores. Entre eles, os algoritmos baseados na redução de dimensionalidade e algoritmos baseados nas preferências do usuário são amplamente utilizados. Embora utilizados em diferentes abordagens de maneira separada, não há estudos na literatura que abordam o uso de redução de dimensionalidade e algoritmos baseados em preferências de forma combinada. Diante disso, este trabalho propõe uma abordagem chamada MaDRUP, Otimização de Muitos Objetivos com Redução de Dimensionalidade baseada em Preferências dos Usuários. Esta abordagem visa a reduzir o número de objetivos a serem otimizados com base nas preferências declaradas durante o processo de geração da solução, e tem como objetivo principal a geração de um conjunto reduzido de soluções que leva menos tempo de execução, mas mantém competitivamente os atributos de qualidade em comparação com outros algoritmos. Para avaliar a aplicabilidade da abordagem proposta, MaDRUP foi instanciada com o algoritmo NSGA-II. Essa instanciação é chamada COR-NSGA-II (NSGA-II com Redução de Objetivos baseada em Confiança). O COR-NSGA-II define para cada função objetivo um nível de confiança calculado pelas preferências do usuário fornecidas interativamente. Os objetivos com valores mais altos de confiança são removidos da próxima execução do algoritmo. Para avaliar a viabilidade do COR-NSGA-II, também foi implementada uma ferramenta chamada Nautilus, e foram realizados experimentos usando seis LPSs diferentes, dois tipos de pontos de referência representando as preferências do usuário, cinco algoritmos e dois cenários para simular diferentes perfis de usuário. Os resultados mostram que o COR-NSGA-II supera os algoritmos avaliados na maioria dos casos, gerando um número menor de soluções, com um menor tempo de execução. Uma análise qualitativa também foi realizada com um conjunto de 12 usuários que, respondendo a um questionário, indicaram ser mais fácil escolher uma solução gerada pelo COR-NSGA-II do que escolher uma solução gerada pelos outros algoritmos.

Palavras-chave: linha de produto de software. engenharia de software baseada em busca. algoritmos baseados em preferências. redução da dimensionalidade.

## ABSTRACT

Multi- and Many-Evolutionary Algorithms have been applied to derive products for the variability testing of Software Product Lines (SPLs). This problem refers to the selection of an adequate set of products to test a SPL, since to test all the existing products is infeasible. The problem is impacted by many factors, such as the number of products to be tested, testing criteria to be satisfied, and efficacy to reveal faults. We can see that many conflicting objective functions need to be optimized at the same time. However, some problems emerge when the number of objectives to be optimized increases, for example, the solutions generated by the optimization algorithms become incomparable, designing a Pareto-front in this context requires a large number of solutions, and the visualization of such solutions requires special techniques. Several techniques are proposed in the literature to tackle this problem, such as decomposition and algorithms based on indicators. Among them, the algorithms based on dimensionality reduction and algorithms based on the user preferences are widely used. Even though used in different approaches in a separated way, there are no studies in the literature that investigate the usage of dimensionality reduction and preference-based algorithms in a combined way. In light of this, this work proposes an approach called MaDRUP, a Many-objective Optimization with Dimensionality Reduction based on User Preferences for reducing the number of objectives to be optimized based on preferences stated during the solution generation process. This approach has as main goal the generation of a reduced set of solutions taking less execution time but competitively maintaining the quality attributes in comparison with other algorithms. To evaluate the applicability of the proposed approach, we instantiated MaDRUP with the NSGA-II algorithm. Such instantiation is called COR-NSGA-II (Confidence-based Objective Reduction NSGA-II). COR-NSGA-II defines for each objective function a confidence-level calculated with the user preferences provided interactively. The objectives with higher values of confidence are removed from the next algorithm execution. For assessing the feasibility of COR-NSGA-II, a tool called Nautilus was also implemented, and experiments were conducted by using six different SPLs, two types of reference points representing the user preferences, five algorithms, and two scenarios to simulate different user profiles. The results show COR-NSGA-II outperforms most of the evaluated algorithms generating a lower number of solutions with a lower execution time. A qualitative analysis was also performed with a set of 12 potential users. The results show that for such users, the task of selecting a solution generated by COR-NSGA-II is easier than to select a solution generated by other algorithms.

**Keywords:** software product Line testing. search-based software engineering. preference-based algorithms. dimensionality reduction.



## LIST OF FIGURES

2.1	Example of a MOP with two objectives to be optimized. . . . .	22
2.2	Classification of Many-objective Evolutionary Algorithms. Adapted from [51]. . . . .	25
2.3	Preference-based algorithm framework [27]. . . . .	27
2.4	ROI's Representation. Adapted from [52]). . . . .	28
2.5	Non-dominated sorting process for NSGA-II. Adapted from [20]. . . . .	29
2.6	Example of a normalized reference plane for a three-objective problem. Adapted from [47]. . . . .	30
2.7	The effect of $\epsilon$ and its impact in the Pareto-front [22]. . . . .	32
2.8	R-Metric Steps (Adapted from [52]). . . . .	34
2.9	R-Metric example. . . . .	35
2.10	Steps of the study selection process. . . . .	36
3.1	PSBSE framework [31]. . . . .	41
3.2	Feature diagram of Mobile Phone. Adapted from [29]. . . . .	42
3.3	Example of products generated from the FM in Figure 3.2. . . . .	42
3.4	Example of a mutant generated for FM in Figure 3.2. . . . .	44
3.5	Individual representation [33]. . . . .	45
4.1	MaDRUP Overview. . . . .	50
4.2	Example of numbered objective values. . . . .	53
4.3	COR-NSGA-II Overview. . . . .	53
4.4	Example of application of the confidence level. . . . .	57
5.1	Nautilus Architecture. . . . .	61
5.2	Nautilus Core's packages. . . . .	61
5.3	Nautilus Plugin's packages. . . . .	62
5.4	Nautilus Web's packages. . . . .	63
5.5	Nautilus's screenshot from home page . . . . .	64
5.6	Nautilus's screenshot from problem page. . . . .	65
5.7	Nautilus's screenshot from problem instance page. . . . .	66
5.8	Nautilus's screenshot from optimize page. . . . .	66
5.9	Nautilus's screenshot from execution page. . . . .	67
5.10	Nautilus's screenshot from solution page. . . . .	68
5.11	Nautilus's screenshot from compare page. . . . .	69
5.12	Nautilus's screenshot from customization page. . . . .	69

6.1	Simulated user representation. . . . .	73
6.2	Example for user simulator evaluation. . . . .	74
6.3	Results from the Participant Questionnaire. . . . .	75
6.4	Preferred Objectives from the user's point of view.. . . .	86
6.5	Easiest algorithms from the user's point of view.. . . .	87
6.6	Best algorithms from the user's point of view. . . . .	88
6.7	<i>How much time?</i> answers from questionnaire. . . . .	89
6.8	<i>How difficult time?</i> answers from questionnaire. . . . .	89
6.9	<i>Agree or not?</i> answers from questionnaire.. . . .	90
6.10	<i>Clear or not?</i> answers from questionnaire.. . . .	90
6.11	Nautilus' best features. . . . .	91
C.1	Feature Model for James (Adapted from [6]). . . . .	113
C.2	Feature Model for CAS (Adapted from [84]). . . . .	114
C.3	Feature Model for WS (Adapted from [7]).. . . .	115
C.4	Feature Model for E-Shop (Adapted from [71]). . . . .	116
C.5	Feature Model for Drupal (Adapted from [62]). . . . .	117
C.6	Feature Model for Smarthome (Adapted from [43]). . . . .	118

## LIST OF TABLES

2.1	Search Terms. . . . .	36
2.2	Number of found studies in each electronic database. . . . .	37
2.3	Inclusion and exclusion criteria applied to the studies. . . . .	37
3.1	Objective Functions used in this work. . . . .	48
3.2	Features from the instance example. . . . .	48
3.3	Products from the instance example. . . . .	48
4.1	Ordinal Scale for the Required Information. . . . .	54
4.2	Confidence Level for removing an objective. . . . .	56
4.3	Confidence Level Example. . . . .	57
6.1	Characteristics of the FMs used in the experiments. . . . .	72
6.2	Groups Organization. . . . .	76
6.3	Reference Points. . . . .	78
6.4	Parameter Settings. . . . .	79
6.5	COR-NSGA-II versus Random Algorithm in Scenario 2D. . . . .	80
6.6	COR-NSGA-II versus Random Algorithm in Scenario 3D. . . . .	81
6.7	COR-NSGA-II Versus NSGA-II and NSGA-III in both scenarios. . . . .	83
6.8	COR-NSGA-II Versus R-NSGA-II in both scenarios. . . . .	84
6.9	COR-NSGA-II Versus COR-NSGA-II in both scenarios. . . . .	85
6.10	COR-NSGA-II's results for each participant. . . . .	87
6.11	Best algorithm by group. . . . .	88
G.1	COR-NSGA-II Versus NSGA-II and NSGA-III in Scenario 2D. . . . .	123
G.2	COR-NSGA-II Versus NSGA-II and NSGA-III in Scenario 3D. . . . .	124
G.3	COR-NSGA-II Versus R-NSGA-II in Scenario 2D. . . . .	125
G.4	COR-NSGA-II Versus R-NSGA-II in Scenario 3D. . . . .	126
G.5	COR-NSGA-II Versus PCA-NSGA-II in Scenario 2D. . . . .	127
G.6	COR-NSGA-II Versus PCA-NSGA-II in Scenario 3D. . . . .	128
H.1	Preferred and final subset of objectives for each participant. . . . .	129

## LIST OF ACRONYMS

ACO	Ant Colony Optimization
AETG	Automatic Efficient Test Generator
AGM	Arcade Game Maker
AHP	Analytic Hierarchy Process
AOS	Adaptive Operator Selection
CAS	Car Audio System
COR-NSGA-II	Confidence-based Objective Reduction NSGA-II
DM	Decision Maker
DR	Dominance Resistance
EA	Evolutionary Algorithm
EC	Evolutionary Computation
FaMa	Feature Model Analyser
FM	Feature Model
FMTS	Feature Mutation-based Test Suite
FRRMAB	Fitness Rate based Multi-Armed Bandit
FODA	Feature-oriented Domain Analysis
GA	Genetic Algorithm
HH	Hyper-heuristic
HV	Hypervolume
IBEA	Indicator-Based Evolutionary Algorithm
IGD	Inverted Generational Distance
JSON	JavaScript Object Notation format
WS	Weather Station
ROI	Region of Interest
RP	Reference Point
MaOP	Many-objective Problems
MaOEA	Many-objective Evolutionary Algorithm
MaDRUP	Many-objective Optimization With Dimensionality Reduction based on User Preferences
MCDM	Multi-criteria Decision Making
MOEA	Multi-objective Evolutionary Algorithm
MOEA/D	Multi-objective Evolutionary Algorithm Based on Decomposition
MOEA/D-DRA	MOEA/D with Dynamical Resource Allocation
MOP	Multi-objective Optimization Problem
MVU	Maximum Variance Unfolding (MVU)

NSGA-II	Non-dominated Sorting Genetic Algorithm II
NSGA-III	Non-dominated Sorting Genetic Algorithm III
PC	Principal Component
PCA	Principal Component Analysis
PCA-NSGA-II	Non-dominated Sorting Genetic Algorithm with Principal Component Analysis
PSBSE	Preference and Search-based Software Engineering
R-HV	Hypervolume indicator with R-Metric
R-IGD	IGD indicator with R-Metric
R-NSGA-II	Reference Point-based NSGA-II
r-NSGA-II	Reference Solution-based NSGA-II
RQ	Research Question
SBSE	Search-based Software Engineering
SPL	Software Product Line
SOP	Scalar objective Optimization Problems
TC	Threshold Cut
XML	eXtensible Markup Language

## LIST OF PUBLISHED WORK

The following works were published in the past 4 years within the subject of search-based software engineering, software testing, software product lines, and preference-based algorithms:

### JOURNAL PAPERS

1. Helson Jakubovski-Filho, Thiago Nascimento Ferreira, Silvia Regina Vergilio (2019). **Preference Based Multi-Objective Algorithms Applied to the Variability Testing of Software Product Lines**. *Journal of Systems and Software*, 1515, pp. 194-209 .
2. Thiago Nascimento Ferreira, Jackson A. Prado Lima, Andrei Strickler, Josiel N. Kuk, Silvia Regina Vergilio, Aurora Pozo (2017). **Hyper-heuristic Based Product Selection for Software Product Line Testing**. *IEEE Computational Intelligence Magazine*, 12(2), pp. 34-45.
3. Thiago Nascimento Ferreira, Silvia Regina Vergilio, Jerffeson Teixeira de Souza (2017). **Incorporating User Preferences in Search Based Software Engineering: A Systematic Mapping Study**. *Information and Software Technology*, 90, pp. 55-69.

### CONFERENCE PAPERS

1. Helson Jakubovski-Filho, Thiago Nascimento Ferreira, Silvia Regina Vergilio (2018). **Incorporating User Preferences in a Software Product Line Testing Hyper-Heuristic Approach**. In *Proceedings of the 20th IEEE Congress on Evolutionary Computation (CEC '18)*, pp. 2283-2290., Rio de Janeiro, Brazil.
2. Helson Jakubovski-Filho, Thiago Nascimento Ferreira, Silvia Regina Vergilio (2018). **Multiple Objective Test Set Selection for Software Product Line Testing: Evaluating Different Preference-based Algorithms**. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering (SBES '18)*, pp. 162-171, São Carlos, Brazil.
3. Thiago Nascimento Ferreira, Josiel Neumann Kuk, Aurora Pozo, Silvia Regina Vergilio (2016). **Product Selection Based on Upper Confidence Bound MOEA/D-DRA for Testing Software Product Lines**. In *Proceedings of the 18th IEEE Congress on Evolutionary Computation (CEC '16)*, pp. 4135-4142, Vancouver, Canada.
4. Édipo Luis Féderle, Thiago Nascimento Ferreira, Thelma Elita Colanzi, Silvia Regina Vergilio (2015). **Optimizing Software Product Line Architectures with OPLA-Tool**. In *Proceedings of the 7th International Symposium on Search Based Software Engineering (SSBSE '15)*, pp. 325-331, Bergamo, Lombardy, Italy.

5. Édipo Luis Féderle, Thiago Nascimento Ferreira, Thelma Elita Colanzi, Silvia Regina Vergilio (2015). **OPLA-Tool: A Support Tool for Search-Based Product Line Architecture Design**. In Proceedings of the 19th International Software Product Line Conference (SPLC '15), pp. 370-373, Nashville, Tennessee, USA.

## **WORKSHOP PAPERS**

1. Thiago Nascimento Ferreira, Silvia Regina Vergilio, Jerffeson Teixeira de Souza (2016). **Engenharia de Software Baseada em Busca e em Preferência: Uma Visão Geral**. In Proceedings of the 7th Brazilian Workshop on Search-Based Software Engineering (WESB '16), pp. 1-10, Maringá, PR, Brazil. [In Portuguese]
2. Thiago Nascimento Ferreira, Thainá Mariani, Silvia Regina Vergilio (2016). **Reviewing Six Years of Brazilian Workshop on Search-Based Software Engineering**. In Proceedings of the 7th Brazilian Workshop on Search-Based Software Engineering (WESB '16), pp. 11-20, Maringá, PR, Brazil.
3. Thiago Nascimento Ferreira, Silvia Regina Vergilio (2015). **Utilizando Otimização por Colônia de Formigas na Seleção de Produtos para o Teste de Mutação do Diagrama de Características**. In Proceedings of the 6th Brazilian Workshop on Search-Based Software Engineering (WESB '15), pp. 61-70, Belo Horizonte, MG, Brazil. [In Portuguese]

## CONTENTS

<b>1</b>	<b>INTRODUCTION . . . . .</b>	<b>17</b>
1.1	MOTIVATION . . . . .	18
1.2	OBJECTIVES. . . . .	19
1.3	TEXT ORGANIZATION. . . . .	20
<b>2</b>	<b>OPTIMIZATION ALGORITHMS. . . . .</b>	<b>21</b>
2.1	OPTIMIZATION PROBLEMS. . . . .	21
2.2	EVOLUTIONARY ALGORITHMS . . . . .	22
2.2.1	Multi-objective Evolutionary Algorithms . . . . .	23
2.2.2	Many-objective Evolutionary Algorithms . . . . .	24
2.3	NSGA-II. . . . .	27
2.4	NSGA-III . . . . .	29
2.5	R-NSGA-II . . . . .	30
2.6	PCA-NSGA-II. . . . .	31
2.7	QUALITY INDICATORS . . . . .	33
2.7.1	Hypervolume with R-Metric (R-HV) . . . . .	34
2.7.2	Inverted Generational Distance with R-Metric (R-IGD) . . . . .	35
2.8	WORK ON DIMENSIONALITY REDUCTION BASED USER PREFERENCES	35
2.8.1	Dimensionality Reduction and User Preferences . . . . .	37
2.8.2	Dimensionality Reduction in Software Engineering Problems. . . . .	38
2.9	FINAL REMARKS . . . . .	38
<b>3</b>	<b>VARIABILITY TESTING OF SOFTWARE PRODUCT LINE. . . . .</b>	<b>40</b>
3.1	PREFERENCE AND SEARCH BASED SOFTWARE ENGINEERING . . . . .	40
3.2	SOFTWARE PRODUCT LINE TESTING. . . . .	41
3.2.1	Pairwise Testing in the FM Context. . . . .	43
3.2.2	Mutation Testing in the FM Context . . . . .	43
3.3	WORK ON SEARCH-BASED VARIABILITY TESTING OF SPL. . . . .	44
3.4	SELECTING PRODUCTS WITH A SEARCH-BASED APPROACH . . . . .	45
3.4.1	Solution Representation. . . . .	45
3.4.2	Objective Functions . . . . .	46
3.5	FINAL REMARKS . . . . .	49
<b>4</b>	<b>PROPOSED APPROACH. . . . .</b>	<b>50</b>
4.1	MANY-OBJECTIVE OPTIMIZATION WITH DIMENSIONALITY REDUC- TION BASED ON USER PREFERENCES . . . . .	50



4.2	CONFIDENCE-BASED OBJECTIVE REDUCTION NSGA-II. . . . .	52
4.2.1	Confidence-based Selection Method . . . . .	54
4.3	FINAL REMARKS . . . . .	58
<b>5</b>	<b>NAUTILUS. . . . .</b>	<b>59</b>
5.1	MOTIVATION . . . . .	59
5.2	DESIGN GOALS . . . . .	60
5.3	ARCHITECTURE AND IMPLEMENTATION . . . . .	60
5.3.1	Nautilus Core . . . . .	60
5.3.2	Nautilus Plugin . . . . .	62
5.3.3	Nautilus Web . . . . .	63
5.4	USING NAUTILUS. . . . .	64
5.5	AVAILABLE FEATURES . . . . .	68
5.6	FINAL REMARKS . . . . .	70
<b>6</b>	<b>EMPIRICAL STUDY . . . . .</b>	<b>71</b>
6.1	RESEARCH QUESTIONS . . . . .	71
6.2	TARGET FEATURE MODELS . . . . .	72
6.3	USERS . . . . .	73
6.3.1	Simulated Users. . . . .	73
6.3.2	Real Users. . . . .	75
6.4	QUALITY INDICATORS . . . . .	76
6.4.1	Average Number of Solutions in the ROI. . . . .	77
6.4.2	# of Targets in the Last Subset . . . . .	77
6.4.3	Reduction Capacity . . . . .	77
6.4.4	Reduction Efficiency . . . . .	77
6.4.5	Execution Time . . . . .	77
6.5	DEFINITION OF THE REFERENCE POINTS (RP) . . . . .	77
6.6	PARAMETER SETTINGS . . . . .	78
6.7	RESULTS . . . . .	79
6.7.1	RQ1 - Sanity Check. . . . .	80
6.7.2	RQ2 - Comparing COR-NSGA-II to Multi- and Many-objective Evolutionary Algorithms . . . . .	82
6.7.3	RQ3 - Evaluating the Solutions. . . . .	85
6.7.4	RQ4 - Evaluating Nautilus . . . . .	88
6.8	DISCUSSION. . . . .	91
6.9	THREATS TO VALIDITY . . . . .	92
6.9.1	Internal Validity. . . . .	93
6.9.2	Construct Validity. . . . .	93

6.9.3	External Validity . . . . .	94
6.9.4	Conclusion Validity . . . . .	94
6.10	FINAL REMARKS . . . . .	94
<b>7</b>	<b>CONCLUSION . . . . .</b>	<b>96</b>
7.1	LIMITATIONS AND FUTURE WORK . . . . .	97
	<b>REFERENCES . . . . .</b>	<b>100</b>
	<b>APPENDIX A – ALGORITHMS QUESTIONNAIRE . . . . .</b>	<b>107</b>
	<b>APPENDIX B – NAUTILUS QUESTIONNAIRE. . . . .</b>	<b>110</b>
	<b>APPENDIX C – FEATURE MODELS. . . . .</b>	<b>113</b>
	<b>APPENDIX D – PARTICIPANT QUESTIONNAIRE . . . . .</b>	<b>119</b>
	<b>APPENDIX E – PRE-STUDY QUESTIONNAIRE . . . . .</b>	<b>120</b>
	<b>APPENDIX F – CONSENT TERM . . . . .</b>	<b>122</b>
	<b>APPENDIX G – RQ2 DETAILED RESULTS. . . . .</b>	<b>123</b>
	<b>APPENDIX H – RQ3 DETAILED RESULTS. . . . .</b>	<b>129</b>

## 1 INTRODUCTION

The field known as Search-Based Software Engineering (SBSE) [38] is devoted to the application of search-based techniques to solve different optimization problems from the Software Engineering (SE) area. Search-based techniques include algorithms from the optimization field, such as Genetic Algorithms (GAs), and other evolutionary and bio-inspired ones. Such algorithms search, in a huge potential space, the best solution (or solutions) to solve a problem according to some criteria, generally represented by a fitness function (or objective function) that determines the solution quality.

The SE problems focused by SBSE are hard problems for which, in general, a simple and exact solution does not exist. For example, to find the best refactoring sequence for a program, to allocate the task resources in a best way, to structure the architecture of a system satisfying factors such as cohesion and coupling, and so on.

These problems have some characteristics that make them suitable to be solved by search-based techniques [38]: they are complex; they have a large solution space and the optimal solutions are unknown; to obtain the solutions is very hard and a labor-intensive task for the software engineer; and existence of acceptable metrics to be used in the fitness functions.

Search-based techniques are widely applied to solve SE problems such as software testing, software modularization, software refactoring, software planning [37, 39]. Harman [39] states that some pieces of work address SE problems from a single-objective point of view, in which the main goal is to maximize or minimize one objective function, for example, correctness, quality, etc. However, as pointed out in [59], most SE problems are naturally complex in which many conflicting objective functions need to be optimized at the same time.

One example of such a problem addressed in this work is the variability testing of Software Product Line (SPL). A SPL can be defined as a set of common products from a particular market segment or domain [78]. In this context, the Feature Model (FM) diagram is used for easing feature management in most SPL methodologies. The growing adoption of SPLs in industry demands specific testing techniques, which should guarantee that the products can be derived from the FM match their requirements. Ideally, to ensure this, all products should be tested [74].

The increasing size and complexity of applications can make testing all products almost impossible in practice in terms of resources and execution time [13]. Hence, it makes necessary to select the most representative set of products from FM. However, many factors can impact this selection such as number of products, coverage of testing criteria such as mutation testing and pairwise, dissimilarity of products, importance or cost of the implemented features, and so on.

The number of objective functions to be considered for our problem and most of SE problems is, in general, high (more than three objectives). Such problems are called many-objective ones. However, a survey about this topic [70] reports that 50% of the proposed algorithms address SE problems from only a bi-objective perspective, 30% consider three objectives, and 20% of the existing studies address more than four objectives. As claimed by Mkaouer et al. [59], one reason for SE problems have not been formulated as many-objective ones is due to the challenges in constructing a many-objective solution, since the use of traditional multi-objective techniques is, clearly not sufficient.

There is a growing demand for scalable SBSE approaches that address SE problems in which a large number of objectives are considered. In this perspective, improving the scalability of SBSE approaches will increase their applicability in industry and real-world settings [59].

However, some problems arise when the number of objectives increases. Deb and Jain [19] state that selecting a solution turns into harder because most of the solutions become incomparable, generating a Pareto-front requires a large number of solutions, and visualizing the found solutions needs special techniques.

Several techniques are proposed in the literature [51] for addressing a large number of objectives to be optimized, such as new preference ordering relations, decomposition, and so on. Among them, one of the most-used techniques is dimensionality reduction, widely used in the optimization field, as described in [51].

In the context of dimensionality reduction, many-objective evolutionary algorithms (MaOEAs) are executed seeking to reduce the number of objectives, by removing the redundant ones, that is, objectives where there may not exist any conflict among them. It is possible to cite as an example of MaOEAs, PCA-NSGA-II, an optimization algorithm that uses the concept of Principal Analysis Component jointly with the NSGA-II algorithm for reducing the number of objectives.

Li et al. [51] point out that dimensionality reduction approaches have three main advantages. Firstly, they can reduce the computational load of a MaOEAs. Secondly, they can help decision makers to better understand the many-objective problem by pointing out the non-conflicting objectives, and as third advantage, they are easy to be combined with other approaches. However, the authors also state that if the addressed problem has just conflicting objectives, this one may limit the application of the approach once these algorithms may fail in reducing the number of objectives to be optimized or return a solution set that does not cover the complete Pareto-front. In this perspective, the combination of two or more approaches for tackling many-objective problems would be very interesting [51], as, for example, to take into account the user preferences, since the human knowledge and judgment can be used to guide the search to reach the best solutions.

Regarding the dimensionality reduction and user preferences, there are no studies in the literature that address both topics in an interactive way (or in-the-loop). In addition to this, several works in the literature [28–30, 40, 43, 74, 82] address the variability testing of SPL by using SBSE techniques in which the problem is encoded as an optimization one, and search-based algorithms are applied. Nevertheless, the proposed approaches use a maximum of four objectives to be optimized. In the literature, there are several approaches by addressing several objectives. In this work, we merge all of them in an optimization problem with seven ones.

## 1.1 MOTIVATION

Regarding the above-mentioned context, we have the following motivations to our work:

1. To take into account the user preferences to solve SE problems may generate solutions more reliable and useful in practice according to the user's point of view;
2. To derive products for the variability testing of SPL is a many-objective problem and many-objective approaches can be used for tackling the natural complexity of this one. However, we did not find work in the literature that considers more than four objectives;
3. Recent studies by using algorithms based on dimensionality reduction have presented good results in the optimization field. However, there are no applications of such techniques in SPL testing;

4. The Pareto-front generated to many-objective problems are composed of a large number of solutions. This makes difficult for the user the task of visualizing and choosing a solution;
5. Dimensionality reduction approaches are widely used in the literature to solve many-objective problems. However, it fails if all objectives are conflicting ones;
6. There are no studies in the literature merging dimensionality reduction, and the user preferences provided interactively. The use of this combination may reduce the set of solutions generated by the algorithms, and, at the same time, may reduce the effort in the task of selecting a solution. As a consequence, the use of user preferences in combination with dimensionality reduction is a subject still not explored in SBSE;
7. Some studies point out as future work the integration between dimensionality reduction and user preferences provided by the users, either to select which one to eliminate or to revise the fitness function formulation (for example, aggregating some objectives).

## 1.2 OBJECTIVES

This work intends to explore possible advantages of incorporating the user preferences provided interactively during the search for the reduction of objectives in many-objective optimization. Thus, the hypothesis of this work is that a preference-based dimensionality reduction approach is capable of taking less execution time and generating a reduced set of solutions that takes into account the user preferences. In addition to this, the solutions are as good as those ones generated by multi- and many-objective algorithms with respect to quality indicators from the literature. We expect that an approach such as this may be capable of reducing the set of solutions generated, and with this improve the scalability of SBSE approaches aiming to increase their applicability in industry and real-world scenarios.

Based on that, the specific goals of this work are:

- To provide an approach for guiding the generation of preference-based dimensionality reduction algorithms;
- To provide an algorithm that captures the user preferences interactively and reduces the problem dimensionality;
- To provide a tool that supports the proposed approach as well as distinct optimization algorithms applied to different optimization problems;
- To apply the proposed algorithm for the variability testing of SPL and evaluate the obtained results in comparison with those ones found by multi- and many-evolutionary algorithms.

To achieve the above-mentioned goals, this study introduces an approach called MaDRUP (Many-objective Optimization with Dimensionality Reduction based on User Preferences) that can be instantiated to derive preference-based dimensionality reduction algorithms by providing the concepts, activities, and elements they should implement.

Seeking to evaluate MaDRUP, we derived COR-NSGA-II (Confidence-based Objective Reduction NSGA-II), an algorithm that reduces the problem dimensionality guided by the user preferences. This one has main characteristics capturing the user preferences interactively (or in-the-loop) and performing an online dimensionality reduction. The user can express his/her

preferences about the solutions by using an ordinal scale composed of items *Non-preferred*, *No Opinion*, and *Preferred*. Besides, the algorithm is categorized in the Stay Out class, that is, it considers which objectives should be removed from the next execution. To reach this, the algorithm uses the concept of a confidence level for each objective.

Also, aiming to provide a new tool for incorporating several optimization problems and optimization algorithms, this study introduces Nautilus, a cloud-computing web-platform tool.

To evaluate COR-NSGA-II, we designed a set of experiments on six FMs widely used in the literature. Besides, the obtained results are compared to those ones found by multi- and many-objective evolutionary algorithms such as R-NSGA-II, NSGA-II, NSGA-III, and PCA-NSGA-II. Furthermore, the provided tool and the solutions generated by COR-NSGA-II were evaluated in order to measure their usefulness in the user's point of view.

### 1.3 TEXT ORGANIZATION

This work is organized into chapters. In this chapter, the context, motivations, and objectives of this study were addressed. Chapter 2 presents the basic concepts for the understanding of this work, such as the main concepts about optimization problems, dimensionality reduction approaches, multi- and many-objective algorithms, and related work on this subject. Chapter 3 reviews the variability testing of SPL (the optimization problem addressed in this dissertation), along with the related work on this topic. Chapter 4 introduces the approach proposed in this work, MaDRUP, with its activities and elements, and COR-NSGA-II, the algorithm derived by instantiating MaDRUP and used in the experiments. Chapter 5 describes Nautilus, the tool developed and its main modules, and a use example. Chapter 6 shows the experimental evaluation conducted to assess the feasibility of COR-NSGA-II. In this chapter, we describe the experimental setup, report the obtained results, and discuss these ones aiming to answer some research questions proposed. Finally, Chapter 7 concludes this work by showing the limitations and future work.

The dissertation also has seven appendices. Appendixes A and B show the questionnaire used for evaluating qualitatively the approach proposed in this work. Appendix C contains the FMs used in our experiment. Appendixes D, E, and F present the questionnaires used for collecting the participant profile and consent term used in the experimental study. Appendixes G and H show detailed results for the empirical study.

## 2 OPTIMIZATION ALGORITHMS

This chapter introduces some fundamental concepts for understanding the problem and algorithms used in this work. Optimization problems are described in Section 2.1. Section 2.2 presents the multi- and many-objective evolutionary algorithms, as well as the preference-based and dimensionality reduction mechanisms. The next sections describe the algorithms used in this dissertation: NSGA-II (Section 2.3), NSGA-III (Section 2.4), R-NSGA-II (Section 2.5), and PCA-NSGA-II (Section 2.6). The quality indicators used in this work are described in Section 2.7. Section 2.8 describes related work on algorithms for reduction of dimensionality, and, finally, Section 2.9 highlights some final remarks of this chapter.

### 2.1 OPTIMIZATION PROBLEMS

An optimization problem aims to find one or more feasible solutions which correspond to extreme values of one or more objectives (or objective functions) regarding the problem constraints [18]. These problems are very common, and the people face with them when, for example, they try to design a solution with the minimum possible cost of fabrication, or either finding the best route (the cheaper, shorter, or faster one) for delivering products in a city.

The number of objective functions to be optimized defines which category the optimization problem belongs to. For example, when an optimization problem involves a single-objective function, it is called Mono-objective Optimization Problem. On the contrary, when the number of objectives to be optimized holds more than one objective function, the problem is called as Multi-objective Optimization Problem (MOP).

A Mono-objective optimization problem can be modeled as follows:

$$\begin{aligned}
 & \underset{x}{\text{minimize}} && f(x) \\
 & \text{subject to} && g_i(x) \leq 0, \text{ for } i = \{1, 2, 3, \dots, m\} \\
 & && h_j(x) = 0, \text{ for } j = \{1, 2, 3, \dots, p\} \\
 & && x \in \Omega
 \end{aligned} \tag{2.1}$$

where  $f(x)$  is the objective function to be optimized,  $g(x)$  and  $h(x)$  are functions meaning the problem constrains and  $\Omega$  the set of all possible solutions for the addressed problem. In this model,  $x$  means a solution from  $\Omega$  and this one should be valid, i.e., it is required to be satisfied by the problem constraints.

Regarding to MOP, Zhang et al. [88] describe that these ones can be formulated as:

$$\begin{aligned}
 & \underset{x}{\text{minimize}} && f_1(x), f_2(x), f_3(x), \dots, f_n(x) \\
 & \text{subject to} && g_i(x) \leq 0, \text{ for } i = \{1, 2, 3, \dots, m\} \\
 & && h_j(x) = 0, \text{ for } j = \{1, 2, 3, \dots, p\}
 \end{aligned} \tag{2.2}$$

where  $x$  is a solution or a vector of decision variable,  $f_i(x)$  is the  $i$ -th objective function to be optimized, and  $g(x)$  and  $h(x)$  are the problem constraints.

In this category, we wish to find the best solution that optimizes the set of objective functions addressed. However, in general, there is no single solution that satisfies all of them. This happens because some objectives are conflicting, that is, a given solution is extreme or the best one with respect to one objective, but it is not for the other ones. The solutions are in such a

way that the values of each objective cannot be improved without sacrificing the values of the other objective functions [1].

This behavior is a kind of alternate trade-off and it generates a set of conflicting solutions called Pareto-front or Pareto optimal set. The solutions in this set are called non-dominated solutions and these ones follow the Pareto-dominance concept. To explain the latter consider the following example: supposing that all objective functions to be optimized ( $f \in F$ ) in a given optimization problem are minimization ones, a solution  $x$  is said to dominate a solution  $y$  ( $x < y$ ) if:

$$\begin{aligned} \forall f \in F : f(x) &\leq f(y) \\ \exists f \in F : f(x) &< f(y) \end{aligned} \quad (2.3)$$

As a consequence, if a solution  $x$  is better or equal to a solution  $y$  in all objectives and better in at least one objective, then  $x$  *dominates*  $y$ . On the contrary, if a solution  $x$  does not dominate  $y$  and vice versa, these solutions are said to be non-dominated and both are part of the Pareto-front. Hence,  $x$  and  $y$  can be chosen as equally acceptable solutions for the addressed optimization problem.

To illustrate this concept, Figure 2.1 shows two objective functions  $f_1$  and  $f_2$  in which the goal is to minimize both of them. In this figure, the solutions  $a$ ,  $b$ ,  $c$ ,  $d$ , and  $e$  are considered non-dominated ones (i.e. it is not possible to conclude what is the best solution) and the solutions  $f$  and  $g$  are dominated by the other ones and, therefore, they should be discarded.

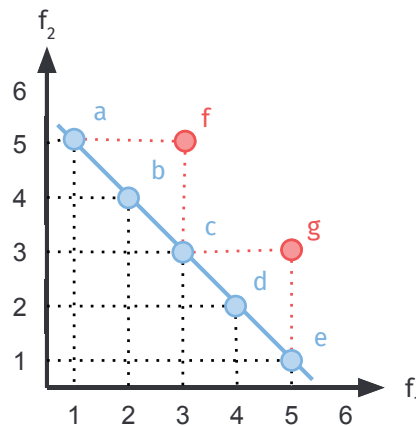


Figure 2.1: Example of a MOP with two objectives to be optimized.

Finding the true Pareto-front (or Pareto optimal set) is a difficult task (or even impossible) due to a large number of sub-optimal Pareto-fronts, the existing problem constraints, and the computational complexity [1]. Depending on the problem instance, there may be too many solutions to evaluate in a feasible time, or even infinite solutions.

Different types of algorithms to solve MOPs have been proposed in the literature [10]. Among them, it is possible to mention the Evolutionary Algorithms (EAs) [89]. These algorithms can find reasonably good approximations of the true Pareto-front in a reasonable time (as known as  $PF_{known}$  fronts). These ones are described in the next section.

## 2.2 EVOLUTIONARY ALGORITHMS

Evolutionary algorithm (EA) is a subclass of Evolutionary Computation (EC) and belongs to a set of general stochastic search algorithm [79]. These algorithms suggest to tackle complex



problems by using techniques inspired by Darwinian natural evolution, that is, mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Among the existing EAs, it is possible to cite the Multi- and Many-objective EAs. All of them are described in the next subsections.

### 2.2.1 Multi-objective Evolutionary Algorithms

Multi-objective Evolutionary Algorithms (MOEAs) [89] are those ones based on Genetic Algorithm (GA) [45] in which, by performing a stochastic optimization method, simulate the natural evolution process aiming to find solutions for MOP. Once they are stochastic [36], the randomness is present, that is, different from exact algorithms (in which every execution returns the same result), these ones can return different results for distinct executions.

MOEAs have as most prominent characteristics the population search strategy and the information exchange between the individuals. So, by using the natural evolution mechanisms, these algorithms can solve traditional problems very quickly [87].

These algorithms use the concept of a population composed of individuals in which each one means a candidate solution. The number of individuals inside a population is a user-specified parameter and its value can impact the scalability and performance of these algorithms. So, once the addressed problem and the solution are encoded (to be used by the algorithms) and the objective functions are defined, the basic procedure used by MOEAs for designing solutions is basically described using the following steps [10]:

1. **Initialization:** An initial population is created with candidate solutions defined, in general, by using some random selection process. However, in this process it is also possible to incorporate some specific information from the domain (or addressed problem);
2. **Evaluation:** After the initialization, the initial population or an offspring one is then evaluated by the objective functions (all solutions inside them are evaluated);
3. **Selection:** In this step, the best solutions are preferred and selected. The idea of this procedure is to designate more copies to a solution with better objective values and, thus, imposes the survival-of-the-fittest mechanism on the solutions. It is possible to find many selection procedures (or selection operators) in the literature [10] that incorporate this idea such as roulette-wheel selection, tournament selection and so on;
4. **Recombination:** The recombination is executed by combining pieces of two or more solutions selected in the previous step (possibly the best ones) aiming to generate a new solution probably, with better objective values. Again, there are several recombination procedures (or crossover operators), and some of them depend on the problem or solution encoding;
5. **Mutation:** The recombination process described in the previous step operates over two or more solutions. However, in the mutation procedure, this operation is performed locally in a single solution. As with recombination, it is possible to find in the literature some examples of mutation procedures (or mutation operators) that involve one or more changes in a solution. As also stated in [10], the mutation performs a random walking surrounding each candidate solution;
6. **Replacement:** After the application of selection, recombination and mutation procedures, an offspring population is created by replacing the initial population (or

the previous one). Many replacement techniques such as elitism, and steady-state replacement methods are used in MOEAs;

7. Repeat the steps 2-6 until reaching the stopping criteria.

It is possible to find several MOEAs in the literature that apply the above steps in their conception. A classification for such algorithms is shown in Figure 2.2.

### 2.2.2 Many-objective Evolutionary Algorithms

Currently, many MOPs are successfully solved by using MOEAs with two or three objective functions. However, their performance tend to decrease when the number of objectives to be optimized increases. Thus, the optimization algorithms have to deal with the following issues [19]:

- The Dominance resistance (DR) phenomenon, that is, the process of selecting a solution from the population becomes basically random once most of them are almost incomparable. Besides, most of the generated solutions are non-dominated ones becoming harder for selecting those ones for keeping in the population;
- Limited solution set size: as described by Deb et al. [19], under non-degenerated scenarios, the Pareto-front of a  $m$ -objective problem is a  $(m - 1)$ -dimensional manifold. In order to design such front, it is necessary to increase the number of solutions exponentially;
- Finally, the visualization of found solutions needs special techniques, such as projection to a lower dimension space, parallel coordinates and so on [80].

For solving these many-objective problems (MaOPs) with more than three objectives to be optimized, several Many-objective Evolutionary Algorithms (MaOEA) are proposed in the literature. The classification of these algorithms can be seen in Figure 2.2.

Basically, they are separated in some categories based on their strategies and some of them are described as follows:

- Pareto-dominance: the algorithms in this category use the concept of Pareto-dominance for comparing the solutions. However, the results reported in the literature, although good ones for some specific problems [46], in general, face the worst results comparing to other strategies;
- Indicator-based: such as IBEA [51], such algorithms do not use the Pareto-dominance concept. On the contrary, they try to maximize a given indicator, but they can deal with increasing computation cost if the indicator used is very slow to calculate;
- Preference-based: in this category, the user preferences are taken into account during the search process and then by reducing the problem complexity;
- Dimensionality Reduction: the algorithms in this category try to reduce the number of objectives to be optimized. However, these algorithms are limited to the problems in which have the possibility to reduce the number of objectives, that is, problems with non-conflicting objectives;

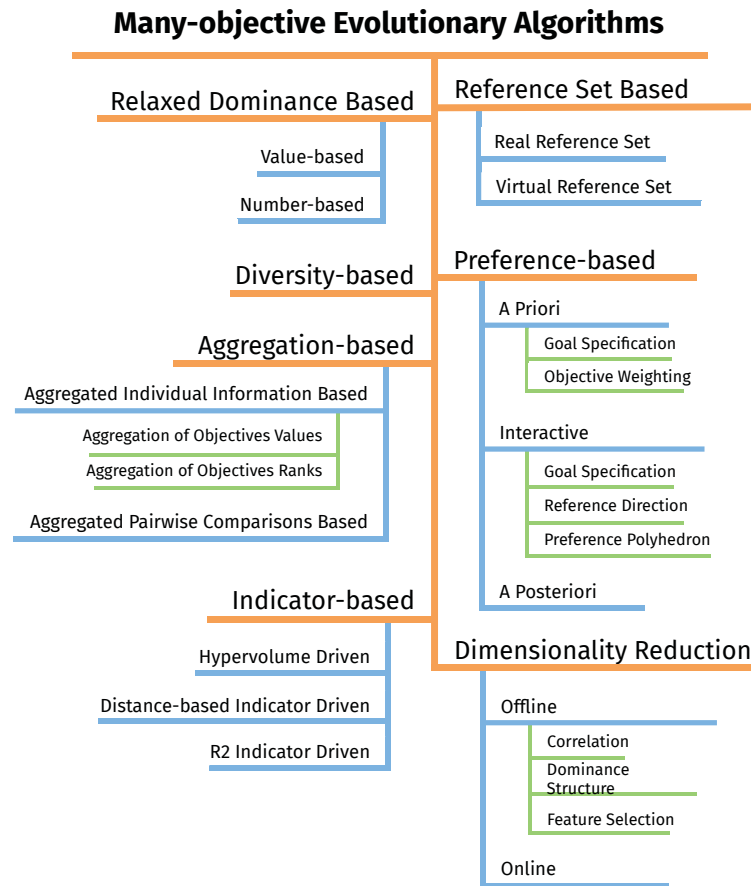


Figure 2.2: Classification of Many-objective Evolutionary Algorithms. Adapted from [51].

- Relaxed dominance-based: the algorithms in this category use a different Pareto-front concept aiming to increase the selection. As difficult it is possible to mention the parameter settings that control the relaxation;
- Hybrid-strategies: the algorithms described in this category are able to implement two or more the aforementioned techniques to solve the MaOPs.

Next, we describe the categories of algorithms for dimensionality reduction, and preference-based algorithms, which are the focus of our work.

### 2.2.2.1 Dimensionality Reduction

MOP involves multiple conflicting objectives, and, because of this, it ideally demands search a multi-dimensional Pareto-optimal front. To find the latter, some MOEAs have been used as a method to find the best representative set of solutions.

Although several papers report good results when MOEAs are applied to problems with two or three objectives, the use of these algorithms raises some discussions about their found results for solving problems with a large number of conflicting objectives, for example, more than ten objectives to be optimized [21].

Such discussions gain credence for various practical reasons, as described as follows. The visualization of a large-dimensional Pareto-front is very difficult, and an exponential large number of solutions would be necessary to represent this large Pareto-front. Besides, it is very

tedious and requires a very burden for the decision makers to analyze this large number of solutions to, at the final, pick a solution up based on his/her preferences for the problem [21].

Most optimization problems involve a large number of objective functions. However, in some problems, even though apparently there exists a conflicting scenario among the objective functions, it is possible to find that, for some ones, there may not exist any conflict (also known as redundant objective functions). As stated by Deb and Saxena [21], in such a case, the optimal Pareto-front will be of a dimension lower than the number of objectives.

Hence, the situation makes necessary the application of techniques responsible for reducing the number of objectives to be optimized by removing the redundant ones. The algorithms that apply these techniques are known as dimensionality reduction algorithms (or objective reduction).

Li et al. [51] categorize the techniques in this subject according to the time for incorporating the dimensionality reduction into MaOEA into two classes: offline and online methods.

For offline methods, the dimensionality reduction process is carried out after obtaining a set of Pareto optimal solutions. For instance, in this class Sinha et al. [73] propose NL-MVU-PCA based on Maximum Variance Unfolding.

For online methods, the number of objectives can be reduced gradually during the search process by iteratively obtaining solution sets and invoking the dimensionality reduction techniques. In the literature, it is possible to find algorithms in this class such as MVU-PCA-NSGA-II, C-PCA-NSGA-II [69], and PCA-NSGA-II [21].

### 2.2.2.2 Preference-based Algorithms

In the literature, approaches based on optimization algorithms have been proposed for solving optimization problems, as described in Section 2.1. The studies have their relevance and show encouraging results. However, the efficacy of these approaches may be questioned once they do not take into account the user participation, that is, these approaches do not consider some subjective aspects of the problem due to the difficulty of incorporating or mathematically model the user preferences. So, to reach better results regarding the reliability, it is important to deal with this issue.

Based on that, the use of preference-based algorithms emerged, allowing the incorporation of human preferences, intuition, emotion or psychological in the optimization process [75]. The user preferences are provided by a Decision Maker (DM) who plays an important role. The DM can be a person or (a group of persons), and it is supposed that s(he) has better insights for the problem. Besides, it is supposed that the user is able to express the preference relations among several solutions [9]. Figure 2.3 presents a basic framework of algorithms that consider the user preferences during the search process.

We can usually define preference-based algorithms in two cycles of executions: an inner and outer cycle. The inner cycle is responsible for generating candidate solutions that posteriorly will be evaluated by an intermediate fitness function. The outer cycle is responsible for selecting some items for the user evaluation through an interaction handler. The user visualizes these items and provides his/her preferences about them. After the user preferences are sent to the algorithm, it incorporates this information in some way into the search process, and the search continues.

According to Miettinen [57], these algorithms can be classified in many ways according to different criteria. However, the classification commonly used defines that the DM can express or provide his/her preferences before (*a priori*), during (interactively or interactive), or after (*a posteriori*) the algorithm run. They are not exclusive and can be combined.

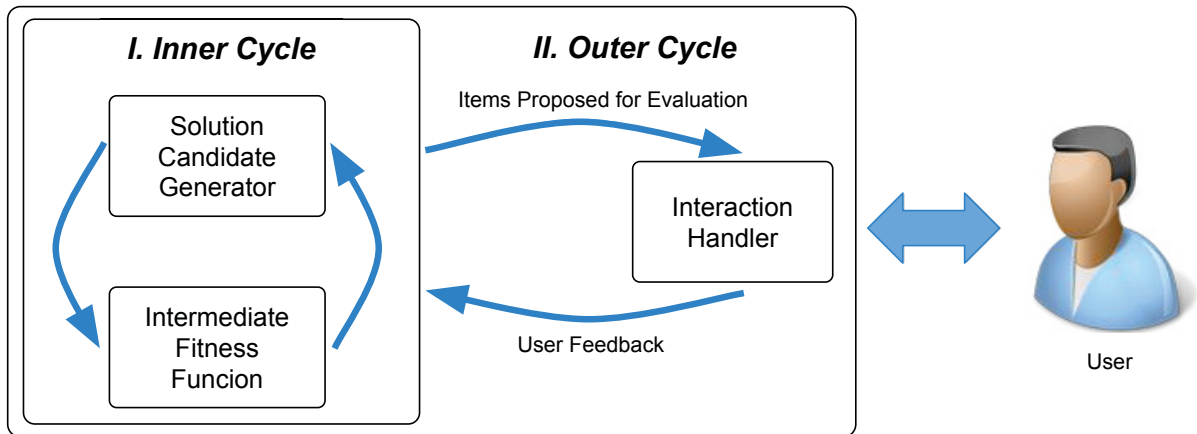


Figure 2.3: Preference-based algorithm framework [27].

One of the most crucial problems in this research area is human fatigue. The latter is a direct consequence of the excessive request for user evaluations. This causes a physiological state of reduction related to physical or mental performance capability. As a consequence, the quality of the evaluations is affected [75].

Based on that, many studies are conducted attempting to mitigate the human fatigue. It is possible to cite as an example speeding up the convergence of the algorithm with a small population and a few number of generations [11]. Nevertheless, it is possible to discrete continuous fitness values into five or seven levels to facilitate the decision making, without compromising the convergence [60]. Finally, Miller [58] discusses in his work the limit on the human capacity for processing information. So, the author also suggests it is possible to extend the fatigue limit organizing the information into “slices” sequence.

Thus, it is very important to define how the user preferences will be provided and which moment they will be incorporated during the search process. As described by Jakubovski Filho et al. [33], the best way to incorporate depends on several conditions, such as the user’s personality, the context, the characteristics of the addressed problem, and so on.

In the literature, it is possible to find some papers that incorporate such user preferences by using the concept of Reference Point (RP), or as known as aspiration level vectors. The RP means to points in the search space in which the user would like the objectives to be concentrated. So, it is possible to assume this one is a natural way to express user preferences [33].

The use of the RP can guide the search toward a Region of Interest (ROI) without demanding effort from the user, even if the number of objectives increases. Figure 2.4 shows an example of the ROI’s representation.

In this figure, the big dotted circle (red) means the ROI generated by the RP represented by a diamond (green). The filled circles (blue) mean the solutions inside the ROI, that is, solutions that are good from the user’s point of view.

In the next sections, we describe all algorithms used in this work. They were selected because these algorithms are widely used in the literature, as well as some of them have never been used in the variability testing of SPL.

### 2.3 NSGA-II

Non-dominated Sorting Genetic Algorithm II (NSGA-II) proposed by Deb et al. [20] is a strong elitist algorithm based on GA, following the concept of elitism, and crowding distance during

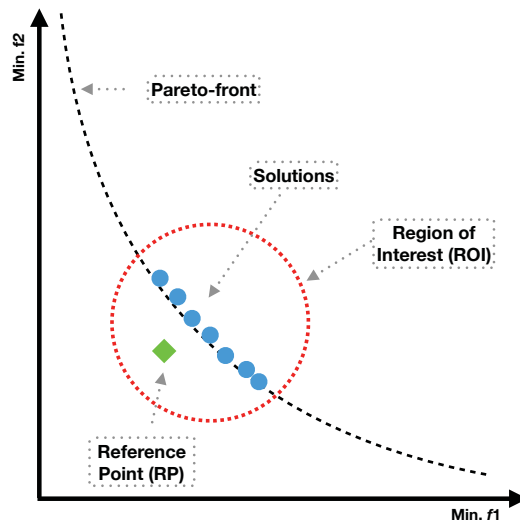


Figure 2.4: ROI's Representation. Adapted from [52]).

the search process. The algorithm is shown in Algorithm 1 and this one requires as input the population size  $N'$ , the number of generations  $g$ , and the objective functions to be optimized  $f_k(X)$ .

---

**Algorithm 1** NSGA-II Algorithm. Adapted from [21]

---

**Input:**  $N'$ ,  $g$ ,  $f_k(X)$

- 1: Initialize Population  $\mathbb{P}'$
  - 2: Generate a random population - size  $N'$
  - 3: Evaluate Objective Values
  - 4: Assign Rank (level) Based on Pareto-dominance - *sort*
  - 5: Generate Child Population with Binary Tournament Selection, and Recombination and Mutation
  - 6: **for**  $i = 1$  **to**  $g$  **do**
  - 7:   **for** each Parent and Child in Population **do**
  - 8:     Assign Rank (level) based on Pareto - *sort*
  - 9:     Generate sets of non-dominated vectors along  $PF_{known}$
  - 10:    Loop (inside) by adding solutions to next generation starting from the first front until  $N'$  individuals found determine crowding distance between points on each front
  - 11:   **end for**
  - 12:   Select points (elitist) on the lower front (with lower rank) and are outside a crowding distance
  - 13:   Create next generation with Binary Tournament Selection, and Recombination and Mutation
  - 14: **end for**
- 

An initial population is generated and, by using binary tournament selection, and crossover and mutation operators, an offspring population is created. With these two populations, basically, the NSGA-II algorithm sorts the population in several non-dominated fronts, according to the non-dominated level. The algorithm combines the parents and offspring before splitting the combined pool into fronts. Then, NSGA-II conducts the niching process by adding a crowding distance to each member.

The crowding distance is a metric used for calculating how far a solution is from the other ones on the same front. This metric is used by NSGA-II in its selection operator, trying to keep a diverse front by making sure each member stays a crowding distance far. At the same time, this procedure keeps the diversity of the population, and it helps the algorithm to explore the fitness landscape [20]. Figure 2.5 shows this procedure.

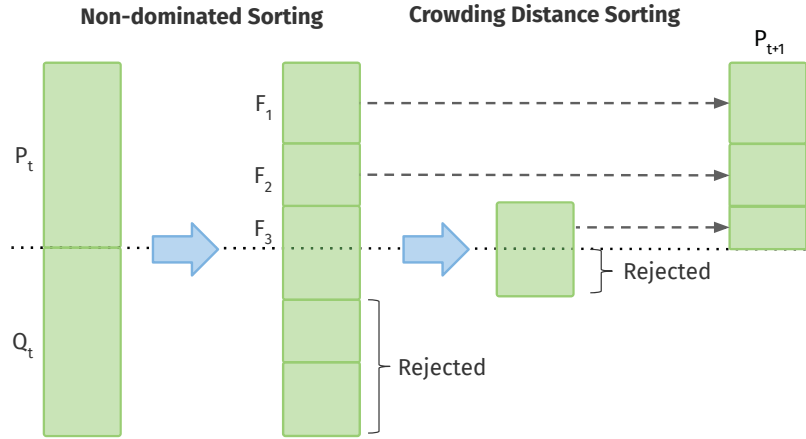


Figure 2.5: Non-dominated sorting process for NSGA-II. Adapted from [20].

In this figure,  $P_t$  is the parent population and  $Q_t$  is the offspring one at the generation  $t$ . So  $F_1, F_2$ , and  $F_3$  are fronts already sorted by the union of  $P_t$  and  $Q_t$  in which  $F_1$  are the best solutions from this combination (parent and offspring),  $F_2$  are the second best ones and so on. However, it is not possible to include the whole  $F_3$  inside  $P_{t+1}$ . So, the crowding distance is used and the best ones (far solutions) are selected from  $F_3$  and added in  $P_{t+1}$ . The process remains running until the number of generation is reached. After that, the best non-dominated front is returned

NSGA-II is one of the most traditional MOEAs and is widely used in the optimization field [87, 90].

## 2.4 NSGA-III

Non-dominated Sorting Genetic Algorithm III (NSGA-III) is a more recent MOEA proposed by Deb et al. [19, 47], similar to NSGA-II described in the previous subsection but, with significant changes in its selection mechanism. Also, this algorithm is focused on MaOPs, i.e., multi-objective ones but with a high number of objectives  $M$  to be optimized ( $M > 3$ ).

NSGA-III basically replaces the crowding distance used by NSGA-II to a different one focused on a set of reference points  $Z^r$  (see Figure 2.6). This mechanism helps to maintain the diversity among the solutions.

The algorithm is shown in Algorithm 2, meaning a generation  $t$  of NSGA-III.

After the recombination, mutation and, non-dominated sorting, all acceptable fronts and the last front  $F_l$  that could not be completely included in  $P_{t+1}$  are included in a set  $S_t$ . After that, the objective values and reference points are first normalized to be an identical range.

An orthogonal distance is computed between a member in  $S_t$  and each of the reference lines (joining the ideal point and a reference point). After that, the solution is then associated with the reference point having the smallest orthogonal distance.

The niche count  $\rho$  (defined as the number of solutions in  $S_t/F_l$  that are associated with the reference point) is computed for each reference point. Then, the reference point having the

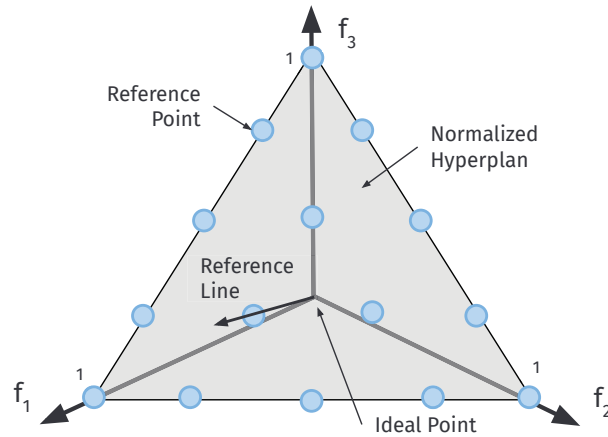


Figure 2.6: Example of a normalized reference plane for a three-objective problem. Adapted from [47].

---

**Algorithm 2** NSGA-III Algorithm [47]

---

**Input:**  $H$  structured reference points  $Z^s$  or supplied aspiration points  $Z^a$

A parent population  $P_t$

**Output:**  $P_{t+1}$

- 1:  $S_t = \emptyset, i = 1$
  - 2:  $Q_t = \text{Recombination+Mutation}(P_t)$
  - 3:  $Rt = P_t \cup Q_t$
  - 4:  $(F_1, F_2, \dots) = \text{Non-dominated-sort}(Rt)$
  - 5: **repeat**
  - 6:  $S_t = S_t \cup F_i$  and  $i = i + 1$
  - 7: **until**  $|S_t| \geq N$
  - 8: Last front to be included:  $F_l = F_i$
  - 9: **if**  $|S_t| = N$  **then**
  - 10:  $P_{t+1} = S_t$ , break
  - 11: **else**
  - 12:  $P_{t+1} = \bigcup_{j=1}^{l-1} F_j$
  - 13: Points to be chosen from  $F_l : K = N - |P_{t+1}|$
  - 14: Normalize objectives and create reference set  $Z_r : \text{Normalize}(f^n, S_t, Z^r, Z^s, Z^a)$
  - 15: Associate each member  $s$  of  $S_t$  with a reference point:  $[\pi(s), d(s)] = \text{Associate}(S_t, Z^r) | \pi(s) : \text{closest reference points}, d : \text{distance between } s \text{ and } \pi(s)$
  - 16: Compute niche count of reference point  $j \in Z^r : \rho_j = \sum_{s \in S_t / F_l} ((\pi(s) = j ? 1 : 0))$
  - 17: Choose  $K$  members one at a time from  $F_l$  to construct  $P_{t+1} : \text{Niching}(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$
  - 18: **end if**
- 

minimum niche count is identified, and the solutions from the last front  $F_l$  that is associated with it, are included in the next population. At the final, the niche count of the identified reference point is increased by one and the procedure is repeated to fill up population  $P_{t+1}$ .

## 2.5 R-NSGA-II

The Reference Point-based NSGA-II (R-NSGA-II) algorithm, proposed by Deb et al. [22] has as goal guiding the search process according to DM preferences provided by a RP. R-NSGA-II has a similar behavior when compared to NSGA-II (described in Section 2.3), but it differs in some



points explained as the following. Firstly, R-NSGA-II requires from DM one or more RPs, and secondly, the crowding distance metric (used in NSGA-II for sorting a front) is modified. In this algorithm, this new crowding distance is called “preferred distance” because it represents how closer the solutions are to the RPs. Thus, the use of this new distance implies in giving a greater emphasis to the solutions that are closer to the RP provided by the user.

Besides, the algorithm has a mechanism to maintain the diversity of selected solutions close to the RPs. This one is a selection strategy called  $\epsilon$ -clearing in which, by using a parameter named  $\epsilon$ , gives special importance to the closest solutions to the RP.

Thus, based on the aforementioned concepts, the niching strategy of NSGA-II is then updated to incorporate the idea in which the solutions closer to the RP should to be more emphasized, and the solutions within a  $\epsilon$ -neighborhood to a near RP should de-emphasized in order to keep a diverse set of solutions near each RP [22]. So, the updated steps are described as follows:

- Step 1: For each solution of the front, the normalized Euclidean distance is calculated for each RP. The solutions are ranked in ascending order of distance, in which the solution that has the smallest distance from RP is in the top of the rank;
- Step 2: After the previous step, there will exist different rankings, one for each RP. The preferred distance of a given solution will be the minimum one assigned to it, considering all the rankings. Thus, the solutions with the smallest preferred distance values are preferred in the tournament selection and in the composition of the new population (from the combined parent and offspring population);
- Step 3: Aiming to control the extent of selected solutions, the  $\epsilon$ -clearing procedure is executed in this step. To perform it, a random solution is selected from each group. Thus, all solutions that have a sum of normalized difference in objective values of  $\epsilon$  or less are discouraged by assigning an artificial large preference distance (aiming to remove them from the search process). So, only one solution within  $\epsilon$ -neighborhood is emphasized. Next, another solution is chosen from the set of non-dominated solutions (excluding the one previously chosen), and the procedure is performed again.

The value of  $\epsilon$  is chosen according to the application and it consists of a user-defined parameter [22]. Figure 2.7 shows the effect of  $\epsilon$  when this one has its value increased. It is possible to notice that the greater the values, the greater is the number of selected solutions in the Pareto-front.

## 2.6 PCA-NSGA-II

In this context of dimensionality reduction, Deb and Saxena [21] proposed PCA-NSGA-II, an elitist non-dominated sorting genetic algorithm with principal component analysis coupled. The idea of this algorithm is to identify redundant objective functions from the solutions found by NSGA-II (described in Section 2.3).

The authors describe the use of the Principal Component Analysis (PCA) method in the context of MOEAs as follows. Supposing we have  $M$ -objective functions to be optimized and  $N$  population members, the initial data matrix  $X$  will be of size  $M \times N$ . So, the procedure converts this matrix into the standardized form, and the correlation matrix (R) is calculated as follows:

$$R_{ij} = \frac{V_{ij}}{\sqrt{V_{ii} * V_{jj}}} \quad (2.4)$$

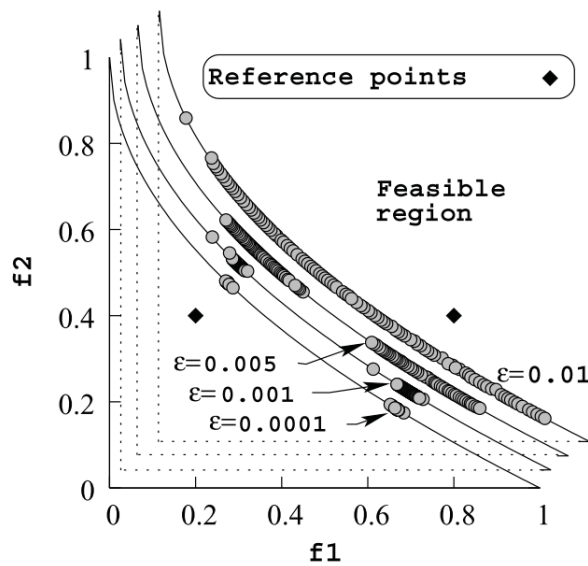


Figure 2.7: The effect of  $\epsilon$  and its impact in the Pareto-front [22].

where  $V$  is the covariance matrix described as:

$$V_{ij} = \frac{X_i X_j^T}{M - 1} \quad (2.5)$$

where  $X_i$  is the  $i$ -th row of  $X$ . This covariance matrix contains values between -1 and 1 in which negative numbers mean the objectives are negatively correlated (there is a conflict among them) and the positive ones mean that the objective are positively correlated (redundant ones).

Given three objectives  $A$ ,  $B$ , and  $C$  as example, consider that  $A$  and  $C$ , and  $A$  and  $B$  are negatively correlated. Thus, for this example, the objective functions  $A$  or  $B$  are redundant ones. In this example, it is easier to identify the redundant objectives because we have just three ones. However, this task becomes harder if the number of objectives increases. So, the authors of PCA-NSGA-II propose a procedure to reduce this burden. This procedure is described in three steps:

1. **Eigenvalue Analysis for Dimensionality Reduction:** The eigenvalues of the correlation matrix  $R$  are calculated and shown ranked in the decreasing order of their magnitudes. The first principal component (PC), corresponding to the largest eigenvalue is designed as  $PC_{A1}$ . So the first component of this vector stands for the contribution of the first objective function towards this vector, the second for the second objective, and so on. A positive value means an increase in the objective value moving along this PC, and a negative denotes a decrease. Thus, by picking the most-negative and the most positive elements from a PC, we can get the two most import conflicting objectives;
2. **Effect of Multiple Principal Components:** In the second step, each PC is analyzed for the two main objectives that are causing a conflict and the information about the other conflicting objectives are collected. However, the authors suggest a procedure in which an analysis is performed from the first PC, the second one, and so on until the significant components are considered. This procedure consists of defining a threshold cut (TC), and when the contribution of all previously PC exceeds this threshold, the analysis is ended. If the TC is too high, many redundant objectives may be chosen. However, if too small, important objectives may be ignored causing an error in the whole study. So,

the authors suggest to use a value of 95% for TC may be better. More details can be found in [21];

3. **Final Reduction Using the Correlation Matrix:** If the previous steps run successfully, the most redundant objectives will be identified. However, if it is possible to perform more reductions, this step is applied. A reduced correlation matrix (only columns and rows corresponding to non-redundant objectives or conflicting ones) is used for identifying if there still exist redundant objectives to be removed. The idea of this step is to establish that any objective is enough to define the conflicting relationship with the remaining objectives. So, this step retains the one which was chosen the earliest by the PCA analysis, but if the objectives come from the same PCA, then the one that has the most significant contribution along next PCA is picked up.

The authors also claim that once PCA-NSGA-II is run for sufficiently large number of generations, the correlation matrix gets stabilized and correlation patterns turn invariant over the number of generations. So, based on the steps previously described, we can state the PCA-NSGA-II algorithm is described in Algorithm 3.

---

**Algorithm 3** PCA-NSGA-II Algorithm [21]

---

- Step 1: Set an iteration counter  $t = 0$  and initial set of objectives  $\mathbb{I}_0 = \{1, 2, \dots, M\}$ ;
- Step 2: Initialize a random population for all objectives in the set  $\mathbb{I}_t$ , run an MOEA, and obtain a population  $P_t$ ;
- Step 3: Perform a PCA analysis on  $P_t$  using  $\mathbb{I}_t$  to choose a reduced set of objectives  $\mathbb{I}_{t+1}$  using the predefined TC. Steps of the PCA analysis are as follows:
- 1) Compute the correlation matrix using Equation 2.4;
  - 2) Compute eigenvalues and eigenvectors and choose non-redundant objectives using the Procedures 1 and 2 previously described;
  - 3) Reduce the number of objectives further, if possible, by using the correlation coefficients of the non-redundant objectives found in item 2 above, using the procedure 3 previously discussed.
- Step 4: :If  $\mathbb{I}_{t+1} = \mathbb{I}_t$ , stop and declare the obtained front. Else set  $t = t + 1$  and go to Step 2.
- 

## 2.7 QUALITY INDICATORS

There are some quality indicators that are used in the literature for evaluating the performance of MOEAs and MaOEAs [52, 90]. They allow comparing the results obtained by different algorithms. These indicators are usually based on the non-dominated solutions set generated by the algorithms. Three sets are usually used:

- $PF_{approx}$ : set of non-dominated solutions obtained by one algorithm execution;
- $PF_{known}$ : set of non-dominated solutions of an algorithm obtained by the union of all the  $PF_{approx}$  from all the executions, removing the non-dominated and repeated solutions;

- $PF_{true}$ : represents the Optimal Pareto-front to the problem. In our case this set is unknown. Due to this, and following the literature [28, 33, 74, 91], this set was formed by all sets  $PF_{known}$  obtained from different algorithms by removing dominated solutions and repeated ones. The set  $PF_{true}$  is, in fact, an approximation to the real front.

The quality indicators that are relevant to the scope of this work and used for answering the research questions are described in the next sub-sections.

### 2.7.1 Hypervolume with R-Metric (R-HV)

Traditional performance evaluation metrics do not take into account the RP informed by the DM during the evaluation process. Considering the great importance of this information when applying a preference-based algorithm, we decided to use the Hypervolume indicator with R-Metric (R-HV) proposed by Li et al. [51]. It provides a way to adapt quality indicators, such as the hypervolume (HV), to quantitatively evaluate the performance of preference-based algorithms by using RP.

The general idea of R-metric is to pre-process the preferred solutions according to a multi-criterion decision-making approach before using a regular metric to evaluate the performance of the obtained solutions. Figure 2.8 presents an illustration of the steps applied by R-Metric calculation principle.



Figure 2.8: R-Metric Steps (Adapted from [52]).

The first step is to filter the solutions by keeping only the non-dominated and no-repeated ones (Prescreening). In the second step (Pivot Point Identification), a representative point is identified, which reflects the general satisfaction of the solutions with respect to the RP. In the third step (Trimming), only solutions located in the ROI are of interest to the user. The R-Metric defines the ROI as a set of solutions that is centered at the pivot point and with length  $\delta$ . Only solutions located in this approximated ROI are valid for performance assessment. After this, in the fourth step (Solution Transfer), the trimmed points are transferred to a virtual position to be evaluated its proximity to the RP. Finally, the last step (R-Metric Calculation) applies the quality indicator in the solutions processed by R-Metric. In our case, the Hypervolume (HV) quality indicator [90].

Figure 2.9 shows an example of the application of the R-Metric for five Pareto-fronts, in which Figure 2.9 (a) shows the original Pareto-fronts and Figure 2.9 (b) the virtual ones after the application of the R-Metric. It is possible to see that, for example, for the Pareto-front  $S3$ , the solutions in this one are closest to the RP, then the solutions remain almost in the same position in the search space. However, for Pareto-front  $S5$ , its virtual position is more distant from RP provided by the user. This will impact the quality attributes for this Pareto-front once the values will be calculated taking into consideration the virtual position.

The objective of R-Metric is to evaluate the dissemination of solutions in the ROI and, at the same time, the proximity of these solutions to the RP. In this work, R-HV was calculated considering the sets  $PF_{approx}$  generated by all algorithms. At the end, the average of the results obtained by calculating the R-HV in each set is returned. For this, the objectives values are normalized between [0.0; 1.01]. Thus, higher values of R-HV present the best results, that is,

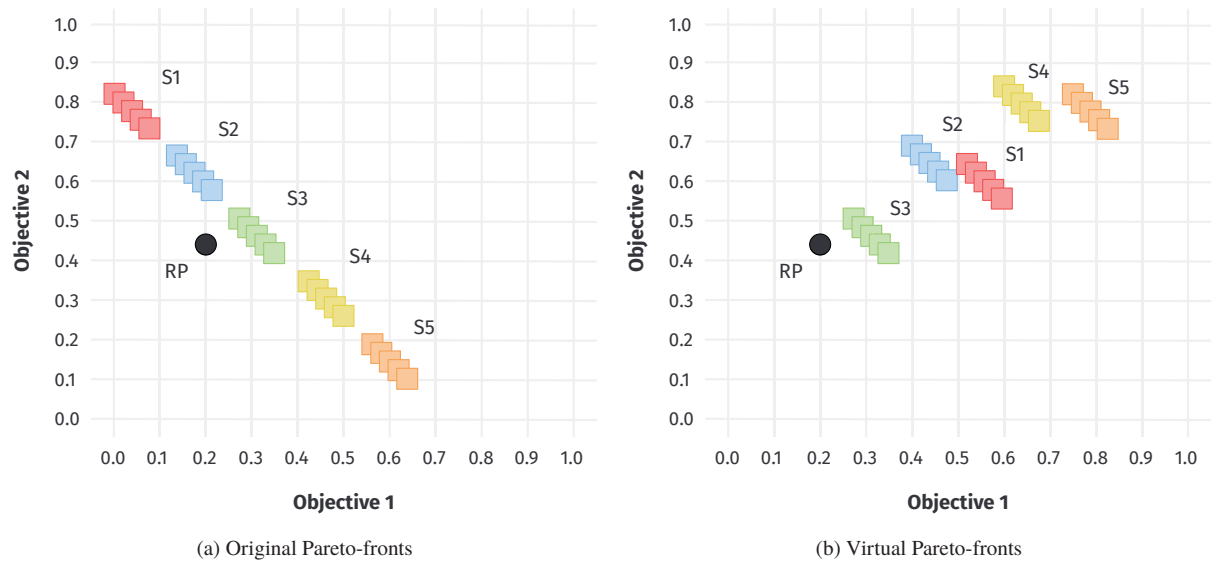


Figure 2.9: R-Metric example.

results that contain a set of solutions that are closer to the RP and also contain a greater number of solutions with good diversity within the ROI.

### 2.7.2 Inverted Generational Distance with R-Metric (R-IGD)

Inverted Generational Distance (IGD) is a convergence measure that corresponds to the average Euclidean distance between the Pareto-front approximation provided by an algorithm and a reference Pareto-front [51]. Again, this indicator does not take into consideration the user preferences. Thus, the Inverted Generational Distance with R-Metric (R-IGD) [51] follows the general way defined in the previous sub-section and it is also proposed in [52]. However, the main difference is that to calculate this quality indicator, IGD is used instead of HV.

IGD needs an approximated (or real) Pareto-front to be calculated. Thus, some steps of R-Metric are performed on  $PF_{approx}$  such as the steps *pivot identification* and *trimming procedure*. The remaining solutions are considered as trimmed  $PF_{approx}$  and they are used to calculate the R-IGD quality attribute. So, the lower R-IGD, the better the results, that is, results that contain a set of solutions that are closer to the trimmed  $PF_{approx}$ .

## 2.8 WORK ON DIMENSIONALITY REDUCTION BASED USER PREFERENCES

In this section, we aim to describe the related work on Dimensionality Reduction Based on User Preferences. In order to find these works, we conducted a search and screening of papers following some steps from the mapping process proposed by Petersen et al. [65].

Based on the goal of this work and the addressed subjects, a set of keywords was defined to form the search terms. They were categorized into three groups, as presented in Table 2.1. The first group is regarding the dimensionality reduction techniques. In this group, the keywords were based on the terms used in related surveys of literature. The second one is regarding the preference-based techniques. The latter were extracted from surveys [5, 31], which contain a classification list for preference-based multi-objective optimization algorithms.

All groups were combined by using the boolean operator “AND”. So the search string is formed by “Dimensionality Reduction” group AND “Preference-based Techniques” group AND “Search Techniques” group.

Table 2.1: Search Terms.

Group	Keywords	Ref.
Dimensionality Reduction	dimensionality reduction OR dimension reduction OR objective reduction	[34, 51]
Preference-based Techniques	Interactive OR preference OR decision-maker OR user-specified OR region of interest OR in-the-loop OR decision support OR human-evaluated OR decision-making	[5, 31]
Search Techniques	search based OR search-based OR MOEA OR MOA OR multi-objective optimization OR multiobjective optimization OR multiobjective algorithm OR multi-objective algorithm OR metaheuristic OR meta-heuristic OR search algorithm OR genetic algorithm OR genetic programming OR GP OR evolutionary algorithm OR evolutionary computation OR evolutionary optimization OR ant colony optimization OR ACO OR particle swarm optimization OR PSO OR integer programming OR exact optimization OR branch-and-bound OR hill climbing OR simulated annealing OR local search OR IGA OR R-NSGA-II OR PBEA	[31]

The search and the selection of the relevant studies were conducted in four steps as described in Figure 2.10.

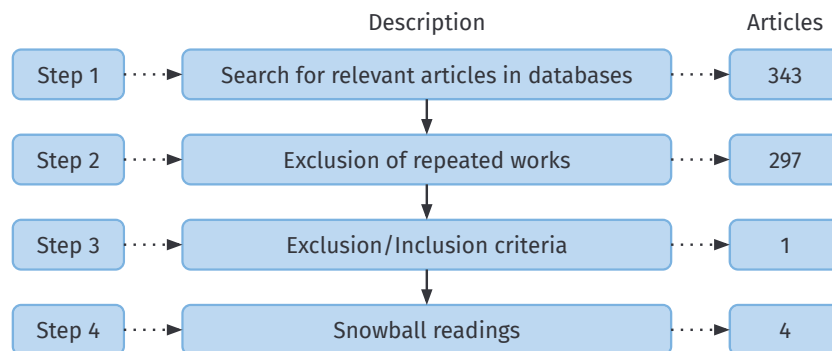


Figure 2.10: Steps of the study selection process.

In Step 1, a query was generated by search string and it was executed in the most relevant electronic databases by considering the title, abstract and keywords. The search finished at October 10th, 2019. The used databases were IEEE Xplore Digital Library, ACM Digital Library, Scopus, Springer, and Science Direct. These databases were chosen due to their importance in Computer Science and SBSE. Thus, the number of found studies in each database is described in Table 2.2.

In some cases, the query was implemented using a specific strategy for each database as, for instance, to split the search string into small pieces. It was due to the different features of the search engines. At the end of this step, a set of 343 papers was obtained.

In Step 2, 46 repeated papers were discarded, remaining 297. Thus, in Step 3, the inclusion/exclusion criteria presented in Table 2.3 were applied in the remaining papers. In this step, a paper was included or excluded by reading it in the following order: title, abstract, introduction, conclusion and the entire paper if necessary. This procedure was applied until no doubts were left about its selection. At the end of this, just one paper was obtained.

Table 2.2: Number of found studies in each electronic database.

Database	Website	#
Scopus	<a href="http://www.scopus.com">http://www.scopus.com</a>	107
Science direct	<a href="http://www.sciencedirect.com">http://www.sciencedirect.com</a>	142
IEEE Xplore Digital Library	<a href="http://ieeexplore.ieee.org">http://ieeexplore.ieee.org</a>	26
ACM Digital Library	<a href="http://dl.acm.org">http://dl.acm.org</a>	49
Springer	<a href="http://www.springerlink.com">http://www.springerlink.com</a>	19
<b>Total</b>		<b>343</b>

Table 2.3: Inclusion and exclusion criteria applied to the studies.

Inclusion criteria	<ul style="list-style-type: none"> <li>• English papers;</li> <li>• Publications in journals, conferences and workshops; tutorials, short papers, tool demonstration, entire thesis, book chapter, technical reports;</li> <li>• Available in an electronic format: HTML, etc;</li> <li>• Mapping studies, surveys, state-of-art and literature review;</li> <li>• With focus on dimensionality reduction and user preferences.</li> </ul>
Exclusion criteria	<ul style="list-style-type: none"> <li>• Position papers and doctoral symposium;</li> <li>• Abstracts;</li> <li>• Papers not available online;</li> <li>• Without focus on dimensionality reduction and user preferences.</li> </ul>

In the last step (Step 4), a snowball sampling was performed following the instructions in [85] by considering both forward and backward snowballing procedures. Then, citations and the reference list of found publications were used to identify other relevant studies and we identified 3 additional papers. Hence, the final set was composed of 4 studies. The found ones are described in the next.

### 2.8.1 Dimensionality Reduction and User Preferences

Sinha et al. [73] propose a framework composed of an algorithm and a procedure executed sequentially. The first algorithm is called NL-MVU-PCA and it is responsible for simplifying the number of objectives by using a machine learning based objective reduction algorithm. It uses the Maximum Variance Unfolding (MVU) and Principal Component Analysis (PCA) nonlinear objective reduction algorithm. As output, this generates a Pareto-front with a reduced and

non-redundant set of objectives. The second one, called PI-EMO-VF, is a procedure responsible for providing to the user the solutions found by NL-MVU-PCA and, by capturing the user preferences a posteriori, constructing an implicit value function for, consequently, making decisions. In the experiments, the authors observed that the approach reduced the cognitive load in the task of selecting a solution based on the user preferences.

### 2.8.2 Dimensionality Reduction in Software Engineering Problems

Dea [16] presents a new software refactoring approach by using PCA-NSGA-II aiming to reduce the set of objectives that represents the quality metrics of interest to the domain expert. The published work has been in a formulating stage and has not been evaluated.

In their work, Dea [17] applied the same algorithm from the previous study in Software Refactoring problem. The author conducted a human study on a set of software developers who evaluated the approach and compared it with the state-of-the-art refactoring techniques. The results presented that the proposed approach outperformed several of existing multi-objective refactoring techniques in some metrics such as execution time, and number of fixed anti-patterns.

Wang and Kessentini [81] introduce a dimensionality reduction approach based on PCA-NSGA-II to address the Web services modularization problem. In this work, the algorithm starts with a large number of Web service quality metrics as objectives that are reduced based on the correlation among them. The authors evaluated their approach in a set of 22 real-world web services and the results show that the algorithm performed significantly better than the state-of-the-art modularization techniques.

## 2.9 FINAL REMARKS

This chapter presented the topics related to this work including the main concepts on Optimization Problems and the characteristics of MOEAs. We described the algorithms NSGA-II and NSGA-III, the Preference-based algorithm R-NSGA-II, and the Dimensionality Reduction algorithm PCA-NSGA-II, as well as the quality indicators used for evaluating them.

As seen in this chapter, MOEAs are useful optimization algorithms when the addressed problems have at most three objectives to be optimized. When the number of objectives increases, the results found by such algorithms tend to deteriorate, once it is hard to differentiate and select the solutions.

In the user's point of view, this generates another problem, that is, the user have to visualize a lot of solutions and the process of picking a solution up becomes burden and, at the same time, the user can reject this task (because it is harder to select a solution) or can reject the solutions generated once the algorithms do not take into account his/her preferences.

With that in mind, in the context where all objectives should be optimized but some of them are preferred, this work has as goal the investigation of a dimensionality reduction approach based on the user preferences for solving MaOPs. For this end, the proposed approach in this work merges the concept of objective reduction and user feedback provided interactively (in-the-loop) during the search, trying to remove unimportant objective functions from the user's point of view aiming to reduce the number of solutions s(he) will visualize.

As far as we know, there not exist an approach or algorithm that incorporates both manners at the same time to find the best solutions in MaOPs. The only study most related to this work proposes to use dimensionality reduction and user preferences in a separated way, not combined, as well as the preferences are not provided in-the-loop.

Besides, dimensionality reduction applied in SE problems does not cover the addressed problem in this work, even the use of the user preferences during the reduction of the number



of objectives. However, Dea [17] suggests as future work researches on direction to integrate the users in-the-loop when reducing the number of objectives to either select which objective to eliminate or to revise the fitness function formulation (for example, aggregating some objectives).

Hence, as far as we are aware, the approach proposed in this work is the first effort on performing the dimensionality reduction based on the user preferences expressed during the search process.

The proposed approach is applied to select the products for the variability testing of SPL, a problem in the SBSE field that is addressed in the next chapter.

### 3 VARIABILITY TESTING OF SOFTWARE PRODUCT LINE

This chapter describes the problem addressed in this dissertation, the variability testing of Software Product Line (SPL). Such a problem belongs to the area of Search-based Software Engineering (SBSE), addressed in Section 3.1. The SPL testing is described in Section 3.2, followed by related work to this topic presented in Section 3.3. Section 3.4 shows the search-based approach for the variability testing of SPL with the proposed objective functions and solution representation, and, finally, Section 3.5 highlights some final remarks.

#### 3.1 PREFERENCE AND SEARCH BASED SOFTWARE ENGINEERING

The field known as Search-based Software Engineering (SBSE) [38] is devoted to the application of optimization algorithms for solving different optimization problems from the Software Engineering (SE) area. In this context, we introduced a sub-research field of SBSE called Preference and Search based Software Engineering (PSBSE) devoted to the application of preference and search-based algorithms to solve SE problems [31].

We defined that to apply PSBSE, it is necessary to define four ingredients (the first three are defined by Harman and Jones [38]):

1. a representation to the problem, to allow its manipulation by the search-algorithm;
2. a set of manipulation operators;
3. a fitness function to evaluate the quality of the solutions, which generally rely on software metrics;
4. a way to incorporate the user preferences.

These ones are required by the process that generates one or more solutions to the user. In our work, we distinguished two kinds of process generally involved in such area: a) the process that generates solutions to the problem, and b) the decision-making process. Based on that, our focus is on the solution generation process. In addition to this, we divided this process into three main phases:

1. Pre-processing – related to the operations that are executed before the search begins, such as the population initialization;
2. Intermediate Solutions Generation – related to the generation of the solution through a search-based process;
3. Post-processing – related to the operations that are executed after the search ends such as ranking the solutions or defining a region of interest based on the user preference.

Based on that, the works were classified regarding the phases in three categories: before (*a priori*), during (interactively) or after (*a posteriori*). They are not exclusive and can be combined. In an *a priori* moment, the user preferences are provided in a pre-processing phase such as weights for the objective function. In an interactive moment, the preferences are provided during the solutions generation after repeated (and finite) interactions. Finally, in an *a posteriori*

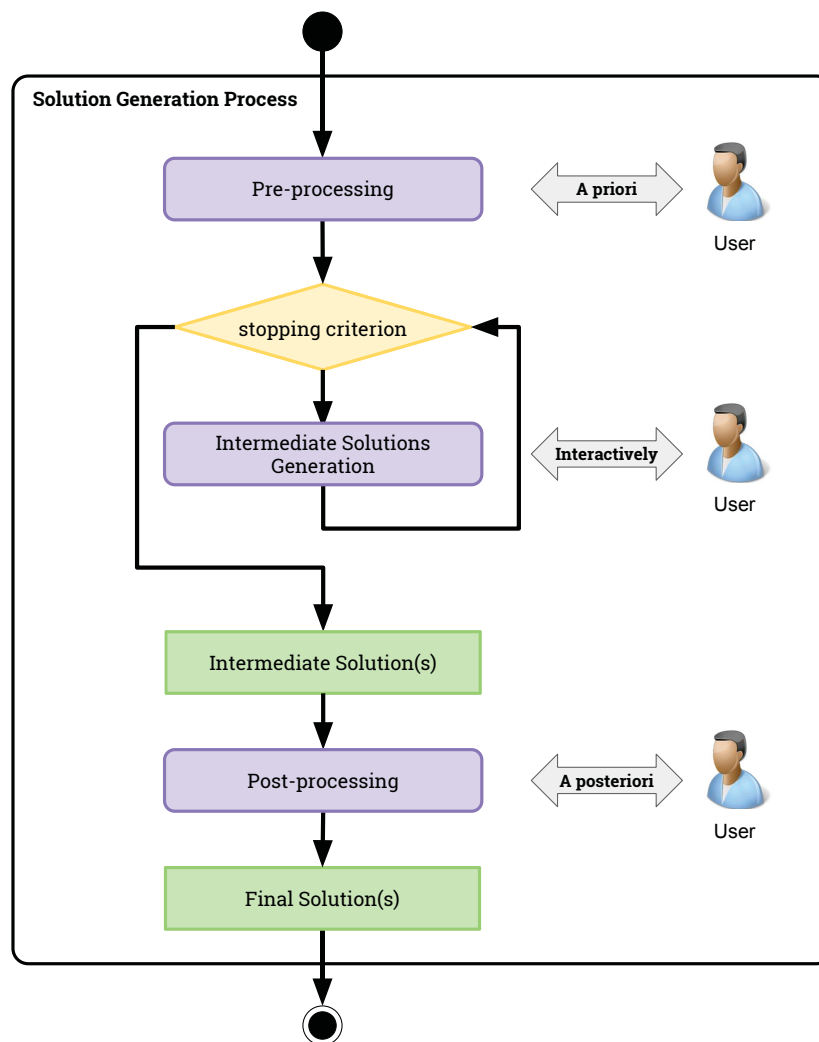


Figure 3.1: PSBSE framework [31].

moment, the preferences are provided after the intermediate solution generation phase, usually used in the multi-objective context aiming to reduce the number of solutions generated by the algorithm. A PSBSE framework is illustrated in Figure 3.1.

The approach proposed in this work is included in the PSBSE research field as an interactive (in-the-loop) approach once the user preferences are provided during the intermediate solutions generation.

Several SBSE problems can be addressed by using user preferences, such as to find the best refactoring sequence for a program, to allocate the task resources in a best way, to structure the architecture of a system satisfying factors such as cohesion and coupling, and so on. One of these problems is the Variability Testing of Software Product Line (SPL), the problem addressed in this work, described in the sequence.

### 3.2 SOFTWARE PRODUCT LINE TESTING

A Software Product Line (SPL) can be defined as a set of common products from a particular market segment or domain [78]. Such products share some features, which represent a functionality, or a system capability that is relevant and visible to the end user [74].

The features can be common to all products derived from the SPL, but they can also be variable being found in only some of them. Thus, the Feature Model (FM) diagram is used for easing feature management in most SPL methodologies. This diagram is represented as a hierarchical arrangement through a tree, and it is used for representing all the SPL commonalities and variabilities, as shown in Figure 3.2, that contains the SPL for the domain of Mobile Phone.

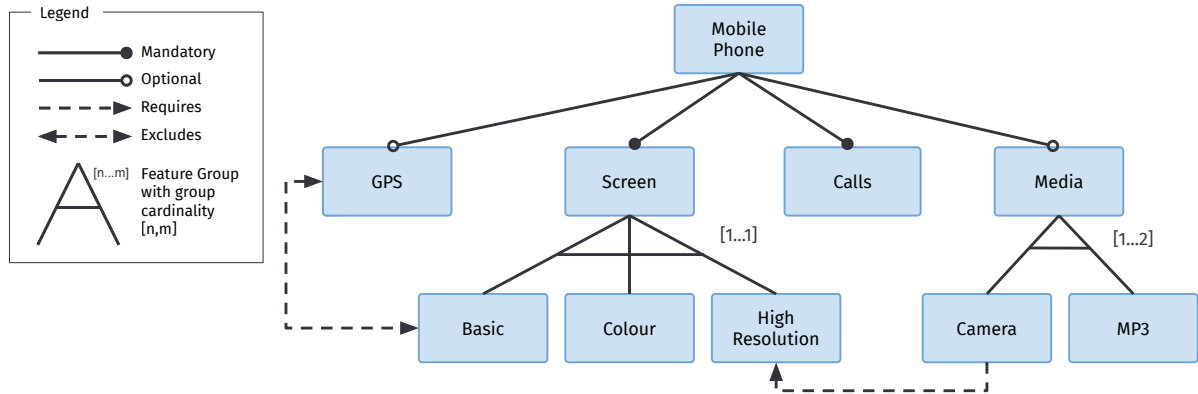


Figure 3.2: Feature diagram of Mobile Phone. Adapted from [29].

In this figure, the features **SCREEN** and **CALLS** in the sub-tree below the feature **MOBILE PHONE** are mandatory ones, that is, all products derived from this FM should implement these ones. On the contrary, the optional features are represented by an empty circle and may not be present in a product, such as the feature **GPS** and **MEDIA**. The group of alternative features is represented by interconnected edges. From this group, only a subset of features can be selected to compose a product. For instance, only one feature must be selected among **BASIC**, **COLOR**, and **HIGH RESOLUTION** below the feature **SCREEN**. A requires relation exists between **CAMERA** and **HIGH RESOLUTION** features. It implies that if a feature *A* is present in a product *p*, then a feature *B* should also be present. The excludes relation implies that both features cannot be present in the same product, such as **GPS** and **BASIC**.

A product is given by a combination of features. Figure 3.3 (a) shows an example of a valid product that can be derived from the FM of Figure 3.2, and Figure 3.3 (b) an invalid one.



Figure 3.3: Example of products generated from the FM in Figure 3.2.

The valid product satisfies all the constraints established in the FM. On the contrary, the product in Figure 3.3 (b) is invalid because it does not include the mandatory features such as **CALLS**.

The growing adoption of SPLs in the industry demanding specific testing techniques. In this context, the variability testing of SPL arises as one important activity. This one tests if the

products that can be derived from an FM match their requirements. To ensure this, all products should be tested [74].

However, the increasing size and complexity of applications can make testing of all products almost impossible in practice in term of resources and execution time [13]. Hence, it makes necessary the application of a technique in order to select the most representative set of products from an FM. In other words, testing criteria should be used. The most popular ones used in the FM-based testing are described in the sequence.

### 3.2.1 Pairwise Testing in the FM Context

In order to derive a set of products for the variability testing of SPLs, some studies in the literature are based on combinatorial testing [41, 61, 64, 77]. Pairwise testing is one of the most popular kind of combinatorial testing, therefore, it is also applied in our work.

The goal of this testing criterion is to generate a set of products that include all the valid pairs of features from the FM. Thus, the number of covered pairs can also be used for evaluating a set of products that were generated.

For instance, consider again the FM shown in Figure 3.2. The pair (GPS, BASIC) is invalid, and should not be required. Considering only the variabilities, we see that the product in Figure 3.3 (a) includes the pair (HIGH RESOLUTION, CAMERA) and does not include the pair (GPS, MP3). Thus, to derive the pairs, we use the Combinatorial tool<sup>1</sup> that implements the Automatic Efficient Test Generator (AETG) algorithm, introduced by Cohen et al. [12].

### 3.2.2 Mutation Testing in the FM Context

Another testing criterion that has been recently explored in the FM context [3, 25, 42, 66] is mutation testing, a fault-based testing criterion. In the FM context, mutant FMs are generated with operators that described possible faults that can be present in an FM. Hence, the goal of this testing criterion is to generate a product that is capable of distinguishing the behavior of the FM being testing from its mutant version.

Essentially, the product  $p$  is checked by using an FM analyzer. The mutant is considered *dead* in two situations: i) if  $p$  is *valid* according to the original FM and *invalid* fo the mutant; and ii)  $p$  is *invalid* for the original FM and *valid* for the mutant. When both FMs, original and mutant ones, validate the same set of products, they are considered as *equivalent*.

At the end of this process, a mutation score is calculated, given by the number of dead mutants over the total of non-equivalent generated mutants. Similarly to the pairwise coverage described in the previous section, the score can be used for evaluating the adequacy of a set of products, or it can be used to improve an existing one.

To illustrate this testing criterion, consider Figure 3.4.

The figure shows that the operator changes a requires relation to an excludes one, such as the one between HIGH RESOLUTION and CAMERA. In this sense, the product in Figure 3.3 (a) kills the mutant, since it is *valid* for the original FM and it is *invalid* for the mutant.

In our work, we use the set of mutation operators and the FMTS (Feature Mutation-based Test Suite) tool proposed by Ferreira et al. [25, 26], which considers FM extensions [15] using UML-like multiplicities. FMTS works with the framework Feature Model Analyser (FaMa) [76], which is responsible for validating the FM being tested and its mutants. Finally, it supports the FODA notation [49], and only valid mutants (which are validated by FaMa), are generated by FMTS.

<sup>1</sup><http://161.67.140.42/CombTestWeb>

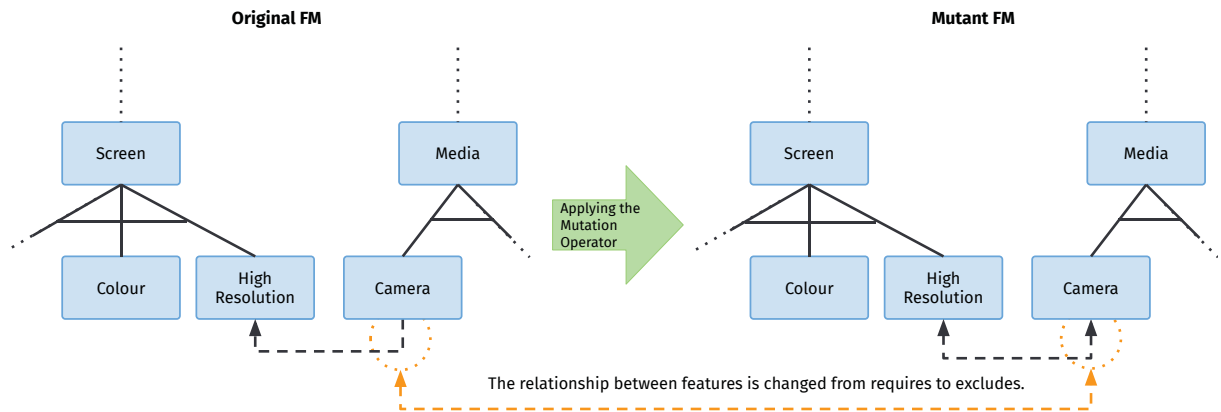


Figure 3.4: Example of a mutant generated for FM in Figure 3.2.

### 3.3 WORK ON SEARCH-BASED VARIABILITY TESTING OF SPL

We performed in 2017 [31] a systematic mapping aiming to find works that take into account the user preferences in SBSE context. The search string used in this study was again executed in order to see whether other researchers published works in this subject since then. As a result of this search, there were not found studies that use dimensionality reduction based on user preferences in SBSE. Thus, in this section, we present works that use optimization algorithms and some of them based on user preferences for the variability testing of SPL.

Wang et al. [82] propose an approach for minimization of test case sets. The authors use a GA and an aggregation function of the following factors: the number of test cases, pairwise coverage, and capability to reveal faults. Besides, other authors [83] also address prioritization of test cases, by using another aggregation function including cost measures, and comparing GA with (1+1) EA and random search. In this study, other factors, such as execution cost and resources, are also considered. In addition to this, Ensan et al. [24] also use a simple GA with an aggregation function comprised of cost and error rate factors.

The work of Henard et al. [43] also uses a GA with an aggregation function to handle the costs, pairwise coverage, and the number of products, all of them conflicting objectives in the selection of test products. Regarding a multi-objective and Pareto approach, Lopez-Herrejon et al. [53] propose a study considering pairwise coverage and the size of the test suites.

Mutation testing has been addressed for test data generation in the works of Henard et al. [40] and Matnei Filho and Vergilio [54]. The former considers mutation operators defined to generate dissimilar products, that is, products that include different features. The latter proposes a multi-objective approach by using two objectives related to the number of dead mutants and products. The authors performed experiments using NSGA-II, SPEA2, and IBEA algorithms.

More recently, Ferreira et al. [30] proposes an approach based on Ant Colony Optimization (ACO) for a single-objective formulation of this problem, besides a mathematical formulation considering the mutation score and the dissimilarity among products. In addition to this, in the context of Hyper-heuristic, several studies propose the use of a Adaptive Operator Selection (AOS) for tackling this problem [28, 29, 74]. For instance, Ferreira et al. [29] introduce a comparative study among four MOEAs based on hyper-heuristics: HH-NSGA-II, HH-SPEA2, HH-IBEA, and HH-MOEA/D-DRA. These algorithms were evaluated in this subject with three- and four-objective formulations, as well as presenting a new objective function regarding to the number of similar products.

Focusing in the application of preference-based algorithms, Jakubovski Filho et al. [48] propose the use of r-NSGA-II (Reference Solution-based NSGA-II). Besides, a hyper-heuristic

version (called r-NSGA-II-HH) is also introduced for this one in which the algorithm works with a random and FRRMAB selection methods. As quality attribute, the work uses R-Hypervolume, a Hypervolume with R-metric [52] in which its value is calculated based on a RP provided by the user.

Still in the same context, Jakubovski Filho et al.[32] propose the use of R-NSGA-II (described in Section 2.5) and a hyper-heuristic version of it (called R-NSGA-II-HH) to solve the same problem. Also, in this work a four-objective formulation is used and the results are compared to traditional algorithms (such as NSGA-II and r-NSGA-II) by showing that, in some metrics, the found results are equivalent.

Finally, Jakubovski Filho et al. [33] perform a deeper evaluation in the same problem but now by using large instances with more than 11k products to be selected. In this study, NSGA-II, a random algorithm, r-NSGA-II and R-NSGA-II are compared by using R-HV, Euclidean Distance and execution time. Three- and four-objective formulations are used and the results show that the r-NSGA-II and R-NSGA-II algorithms outperformed NSGA-II by considering R-HV.

### 3.4 SELECTING PRODUCTS WITH A SEARCH-BASED APPROACH

We can observe that deriving a set of products for the variability test of FMs (the most representative one) is an optimization problem, impacted by many factors such as number of products, coverage of testing criteria such as mutation testing and pairwise, dissimilarity of products, importance or cost of the implemented features, and so on.

As mentioned in the previous section, there are distinct formulations to the problem, as well as many objective functions considering those factors. To validate our approach, we use the formulation proposed in [33], and the set of objective functions are derived considering different approaches [28, 30, 32, 44, 55, 74]. Both, the solution representation and the objective functions are described as follows.

#### 3.4.1 Solution Representation

An individual (or possible solution) in the population is based on a binary encoding, where each gene represents a product derived for a given FM under test. When the  $i$ -th bit is equal to 1 the product  $p_i$  belongs to the solution. Otherwise, the  $i$ -th bit is equal to 0.

We are using the same convention to represent the features in a product, that is, 1 means the corresponding feature is selected for the product, otherwise, 0 the feature is not selected. Figure 3.5 shows an example of an individual for the FM of Figure 3.

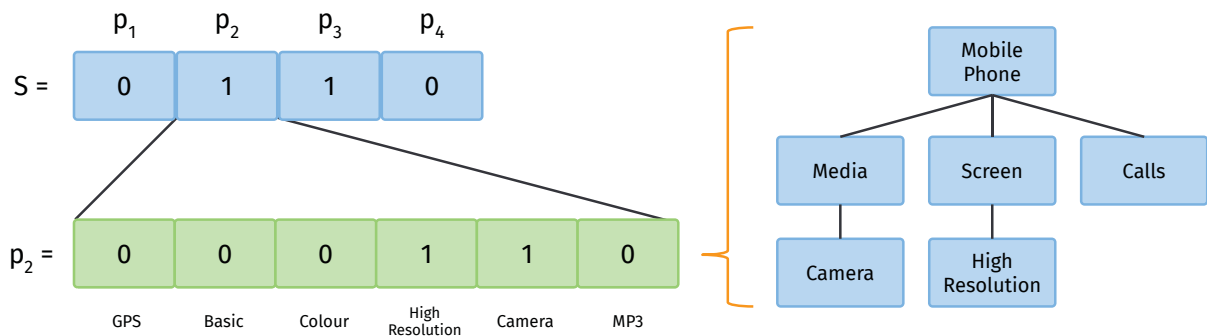


Figure 3.5: Individual representation [33].

In the example, the addressed FM has four valid products being considered. The individual  $S$ , represented in the figure, includes the products  $p_2$  and  $p_3$ , that is,  $S = \{p_2, p_3\}$ . Then, the number of selected products of  $S$ , or  $|S|$ , is 2. The product  $p_2$  in the right side of the figure is represented in terms of its variabilities, and contains the features HIGH RESOLUTION and CAMERA.

### 3.4.2 Objective Functions

Let  $P = \{p_1, p_2, p_3, \dots, p_n\}$  be a set of valid products being considered for the addressed FM, and  $S \subseteq P$  be an individual generated by an algorithm with  $|S|$  selected products.

The first objective (Equation 3.1) corresponds to the total number of products in  $S$ . This number is expressed as the ratio of the number of selected products and the number of valid products being considered for the FM. It is computed as follows:

$$N(S) = \frac{|S|}{|P|} \quad (3.1)$$

In our work, the products  $p_n$  are given by FaMa. However, for huge FMs, the tester can provide a desired value for  $n$ . In the empirical study, we observed that for small instances, around 3% of the generated mutants are equivalent. This percentage was used to set the value of  $n$  for the larger FMs. The number of products  $n$  was chosen ensuring a mutation score around 97%. In this way, the mutants that could not be killed by any one of the generated products were discarded.

The second objective (Equation 3.2) is related to the capacity of the product set to reveal faults (or efficacy). In our case, is given by the mutation score with respect to a set of mutation operators that represent possible faults that can be present in an FM. The objective is defined as:

$$M(S) = 1.0 - \frac{KM}{AM} \quad (3.2)$$

where  $KM$  is the number of killed mutants by the products in  $S$ , and  $AM$  is the total number of active mutants. Basically, this function returns the percentage of alive mutants. Besides, we use the set of mutation operators and the FMTS, and the  $AM$  set of active mutants is composed of only valid and non-equivalent mutants.

The third objective (Equation 3.3) corresponds to the pairwise coverage defined as follows:

$$P(S) = 1.0 - \frac{CP}{VP} \quad (3.3)$$

where  $CP$  represents the number of pairs covered by  $S$  and  $VP$  means the total number of valid pairs. Basically, this function returns the number of uncovered pair. In this work, we use the above-mentioned Combinatorial tool to derive the pairs.

The fourth objective (Equation 3.4) corresponds to the variability (products similarity). This function takes into account the similarity between the products regarding the features they have. It is computed in the next:

$$V(S) = \frac{RF}{OF} \quad (3.4)$$

where  $RF$  means the number of features that appears more than once in  $S$  and  $OF$  is the number of non-mandatory (optional) features in the instance.

The fifth objective (Equation 3.5) takes into account the cost of the selected products, e.g., implementation time of the products or cost of setting up correct resources for developing a



specific product. The cost of a given product is calculated based on the included features. In this work, the cost of leaf and non-leaf features are considered, that is, we assume that the leaf and non-leaf features in a given FM can have concrete implementations, which was already addressed by Pereira et al. [63].

So, this objective is a ratio of the cost of the selected products by the cost of all valid products being considered for the FM. This objective function is calculated as follows:

$$C(S) = \frac{\sum_{p_i \in S} cost(p_i)}{\sum_{p_j \in P} cost(p_j)} \quad (3.5)$$

where  $cost(p_i)$  returns the cost of a product  $p_i$ , estimated by summing of the cost of the features included in  $p_i$ . In this study, the cost assigned for each feature was randomly defined before the search process, once this information is not available in the addressed instances.

The sixth objective (Equation 3.6) considers the richness of features, that is, how many features were included in  $S$ . It is calculated as follows:

$$F(S) = 1.0 - \frac{NF}{TF} \quad (3.6)$$

where  $NF$  means the number of features included in  $S$  and  $TF$  represents the number of features being considered for the FM. Basically, this function returns a percentage of unselected features.

Finally, the seventh objective (Equation 3.7) takes into consideration the feature importance for the stakeholders. As described in the fifth objective, the leaf and non-leaf features in an FM can have concrete implementations. Consequently, in our proposal, the stakeholders are able to express their degree of preference for any feature in the FM. If there are more than one of them or criteria involved, the Analytic Hierarchy Process (AHP) [67] or another Multi-Criteria Decision Making (MCDM) technique could be considered. So, it is calculated as follows:

$$I(S) = 1.0 - \frac{\sum_{p_i \in S} importance(p_i)}{\sum_{p_j \in P} importance(p_j)} \quad (3.7)$$

where  $importance(p_i)$  returns the importance of the product  $p_i$  calculated based on the sum of the importance of the features included in  $p_i$ . Basically, this function returns the percentage of irrelevant features from the user's point of view. Again, in this study, the importance defined for each feature was randomly set before the search process, once this information is not available in the addressed instances.

In this work, all objectives functions are normalized in the range  $[0, 1]$  where 0 is the best value and 1 the worst one, that is, all of them should be minimized. To sum up, Table 3.1 shows a summary of all objective functions used in this work by showing the function, the goal, a short description and the references.

To clarify how the objective functions are computed, we consider an instance example with illustrative values for three features ( $F1$ ,  $F2$ , and  $F3$ ), in which all of them are non-mandatory ones, their cost and importance are described in Table 3.2, and a set of five possible products that should be selected, shown in Table 3.3.

Now, consider that a solution  $S = \{p_3, p_4\}$  was selected to be evaluated in which Products #3 and #4 were selected, the objective functions are calculated as follows:

Table 3.1: Objective Functions used in this work.

#	Function	Goal	Short Description	References
1	N(S)	Minimize	Number of Products	
2	M(S)	Minimize	Alive Mutants	[28, 55, 74]
3	P(S)	Minimize	Uncovered Pairs	
4	V(S)	Minimize	Similarity	[30, 32]
5	C(S)	Minimize	Cost	[83]
6	F(S)	Minimize	Unselected Features	[44]
7	I(S)	Minimize	Unimportant Features	[4]

Table 3.2: Features from the instance example.

Features	Cost	Importance
F1	2	1
F2	4	2
F3	6	3

Table 3.3: Products from the instance example.

#	Features	Killed Mutants	Covered Pairs	Cost	Importance
$p_1$	[F1]	[M1]	[P2, P3]	2	1
$p_2$	[F1, F3]	[M1, M5]	[P2]	8	4
$p_3$	[F1, F2, F3]	[M3]	[P1, P2, P3]	12	6
$p_4$	[F2, F3]	[M1, M2, M4]	[P3]	10	5
$p_5$	[F2]	[M1, M3]	[P1]	4	2
<b>Sum</b>				<b>36</b>	<b>18</b>

$$N(S) = \frac{2}{5} = 0.4$$

$$M(S) = 1.0 - \frac{4}{5} = 1.0 - 0.8 = 0.2$$

$$P(S) = 1.0 - \frac{3}{3} = 1.0 - 1.0 = 0.0$$

$$V(S) = \frac{2}{3} = 0.6$$

$$C(S) = \frac{22}{36} = 0.6$$

$$F(S) = 1.0 - \frac{3}{3} = 1.0 - 1.0 = 0.0$$

$$I(S) = 1.0 - \frac{11}{18} = 1.0 - 0.6 = 0.4$$

Therefore, in this instance example, the objective values for the solution  $S = \{p_3, p_4\}$  is (0.4, 0.2, 0.0, 0.6, 0.6, 0.0, 0.4).

### 3.5 FINAL REMARKS

This chapter introduced the PSBSE research field, as well as the variability testing of SPL. In addition to this, this chapter described the most related studies to this topic addressing the use of MOEAs and preference-based optimization algorithms for SPL testing. Based on that, we introduced a search-based approach including the solution representation and objective functions for tackling this problem. Such an approach is the same one used in [33], and the objective functions were derived considering the objectives which are most commonly used in the literature.

On the one hand, the use of preference-based algorithms in variability testing of SPL reaches good results by using R-NSGA-II, for example. On the other hand, there are no studies that use dimensionality reduction mechanisms for addressing the same problem, even by using both techniques.

Although the use of a preference-based algorithm can reduce the number of solutions to be selected, in some specific problems this reduction can not be successfully reached. So it is necessary to find a way to reduce the number of solutions by guiding the search towards the user preferences. By using dimensionality reduction based on the user preferences expressed during the search process can support this task.

In conclusion, the problem stated in this chapter was used for evaluating the approach proposed in the next chapter.

## 4 PROPOSED APPROACH

This chapter presents MaDRUP (Many-objective Optimization with Dimensionality Reduction based on User Preferences), the approach proposed in this work. Thus, Section 4.1 presents the overview of MaDRUP. Section 4.2 introduces COR-NSGA-II, the algorithm derived by the approach proposed with NSGA-II. Section 4.3 concludes this chapter.

### 4.1 MANY-OBJECTIVE OPTIMIZATION WITH DIMENSIONALITY REDUCTION BASED ON USER PREFERENCES

MaDRUP is an approach that generates solutions for many optimization problems by reducing the number of objectives to be optimized based on the user preferences stated during the solution generation process, that is, in-the-loop.

MaDRUP encompasses three main activities: problem encoding, optimization, and interaction. Figure 4.1 shows the organization of the activities, sub-activities, elements, and identifies briefly the flow of the dimensionality reduction process.

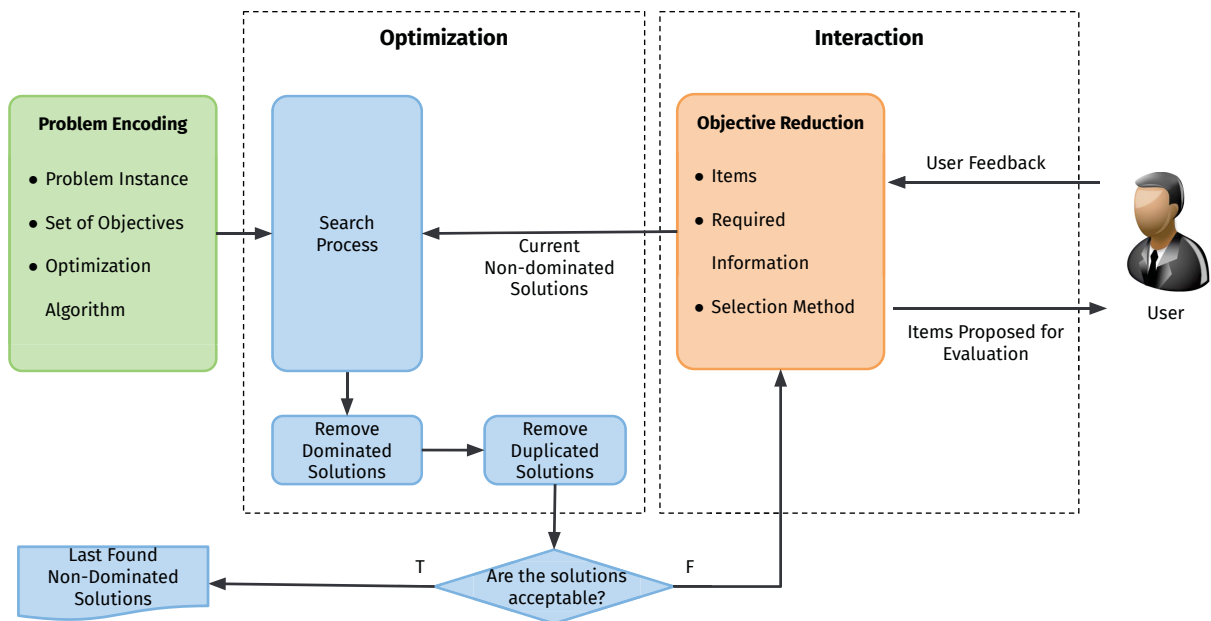


Figure 4.1: MaDRUP Overview.

The Problem Encoding activity is responsible for the definition of three basic elements: a set of objectives to be optimized, an optimization algorithm (used in the search process), and a problem instance. A problem instance contains information about the solution representation, and required information for calculating the fitness functions used by the optimization algorithms.

Regarding the Optimization activity, this is responsible for searching the non-dominated solutions. To reach this, the Search Process sub-activity is responsible for executing the optimization algorithms (such as NSGA-II, SPEA2, and so on) aiming to find solutions for the addressed problem.

The Search Process sub-activity requires an initial population (a set of non-dominated solutions). If the initial population is not provided, a random population is generated by the

optimization algorithm. Once Search Process ends, the dominated and repeated solutions are removed. Then, the stopping criterion is tested.

If the criterion is satisfied, the last found non-dominated solutions are returned as the best ones for the addressed problem. Otherwise, the search process continues but with a new subset of objectives to be optimized. For example, in some approaches such as PCA-NSGA-II, the stopping criterion is defined based on the non-conflicting objectives set found. If the set contains the same objectives to be optimized of the last algorithm execution, the search process ends, and the non-dominated solutions are returned.

In this work, the stopping criterion is defined based on the user preferences. That is, if the found non-dominated solutions are good from the user's point of view, the search ends. Otherwise, the Interaction activity is launched, aiming to define the next subset of objectives.

Regarding the Interaction activity, it aims to provide an interactive way in which the user can provide his/her preferences about the generated solutions. In this activity, a subset or all non-dominated and non-repeated solutions are shown to the user and s(he) is invited to provide his/her preferences. This activity has a sub-activity called Objective Reduction based on three elements. They are:

- **Items:** These ones are elements (usually displayed to the user) in which the user is required to provide his/her preferences. As examples of these items it is possible to cite solutions, variables, objectives, and so on;
- **Required Information:** It is the information (or preference) the user needs to provide about the Items. For example, a ranking of solutions or objectives, or a number meaning his/her preferences about the visualized items and so on. More examples can be found in the work of Ferreira et al. [31];
- **Selection Method:** The main component of this sub-activity, this one is the algorithm used for selecting or choosing the next subset of objectives to be optimized. This one takes into account the Items and the Required Information provided by the user.

Specifically for the Selection Methods, it is possible to categorize the proposed ones into three classes based on the concept used for defining the next set of objectives to be optimized. The classes are described below:

- **Stay In:** In this class, the chosen methods aim to define which objectives should stay in the next subset of objectives based on the user feedback;
- **Stay Out:** In an opposite way, in this class, the goal is to define which objectives should stay out of the next subset of objectives;
- **Composition:** Different from the above ways, in this approach, the goal is to merge some objectives and generate new ones to be optimized in the next algorithm execution.

Thus, once the next subset of objectives is selected in this activity, the search process starts again, but now taking into consideration the new subset of objectives and the non-dominated solutions from the last algorithm execution as initial population. The process keeps running until the last found non-dominated solutions are acceptable from the user's point of view (stopping criteria).

So, it is possible to summarize MaDRUP in some generic steps shown in Algorithm 4. MaDRUP can be instantiated with several preference-based objective reduction algorithms taking

---

**Algorithm 4** MaDRUP Algorithm
 

---

**Input:** A problem instance  
 A set of objectives to be optimized,  
 An optimization algorithm and its parameter settings

**Output:** A set of non-dominated solutions

- 1: Execute the search process;
- 2: Remove the dominated and repeated solutions from population;
- 3: Show to the user the found solutions;
- 4: **while** the user does not accept the found solutions **do**
- 5: User provides his/her feedback for a set of Items;
- 6: The selection method is performed taking into account the user feedback and a new subset of objective is defined;
- 7: Execute the search process again but now with the new subset of objectives and the non-dominated solutions as initial population;
- 8: Remove the dominated and repeated solutions from population;
- 9: Show to the user the found solutions;
- 10: **end while**
- 11: **return** the last found non-dominated solutions;

---

into account the previously described activities. To this end, it is necessary to define the problem instance, the optimization algorithm, and the elements of the Objective Reduction activity.

Then, to investigate the applicability of MaDRUP, we instantiate the proposed approach with NSGA-II. As a result, we derived a new algorithm described in the next section.

#### 4.2 CONFIDENCE-BASED OBJECTIVE REDUCTION NSGA-II

Confidence-based Objective Reduction NSGA-II (or simply COR-NSGA-II) is an instantiation of MaDRUP described in the previous section and it uses the concept of a confidence level for removing an objective from the next algorithm execution. Basically, a confidence level is defined based on the user preferences for each objective to be optimized. These preferences are provided for values closest to the lowest and the highest values for each objective.

To illustrate this, consider that the user is visualizing the non-dominated solutions shown in Figure 4.2

In this figure, considering 0.0 as the best value and 1.0 the worst one, Solution #4 has the best value for Objective 1, Solution #5 has the best value for Objective 2, and Solution #3 the best value for Objective 3. On the contrary, Solution #5 has the worst value for Objective 1, Solution #3 has the worst one for Objective 2, and Solution #1 has the worst value for Objective 3.

COR-NSGA-II needs the user feedback about the objective values found by the search process (circles numbered from 1 to 15). However, the user does not need to provide all of them but just those most important ones from his/her point of view. For instance, the user can provide his/her preferences just for the circles 4, 5, 11, and 12, or, the user can provide his/her feedback just for the extreme values (objectives values equals to 0.0 or 1.0) such as the circles 1, 2, 3, 13, 14, and 15. This is a user decision.

Firstly, COR-NSGA-II requires all non-dominated and non-repeated solutions must be normalized in [0:1] before showing them to the user, and it assumes, initially, all objectives should be selected for the next search process, but some of them should be removed (or not included). This initial assumption is important because if no user preferences are provided, the same objectives should be selected for the next search process and no one should be removed.

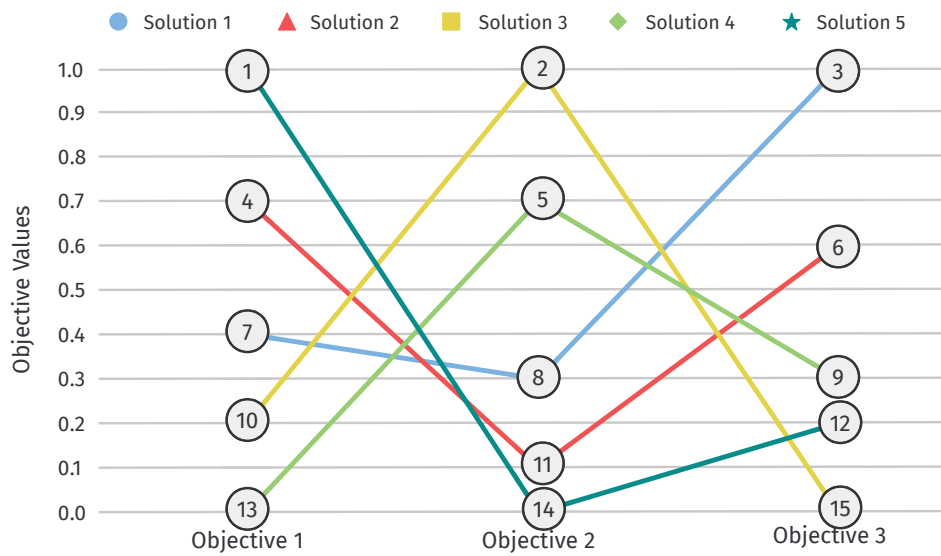


Figure 4.2: Example of numbered objective values.

Thus, once the user feedback is provided, COR-NSGA-II selects those ones closest to the lowest and highest values for each objective and defines a confidence level for removing the objective from the next subset. An overview of the algorithm is shown in Figure 4.3.

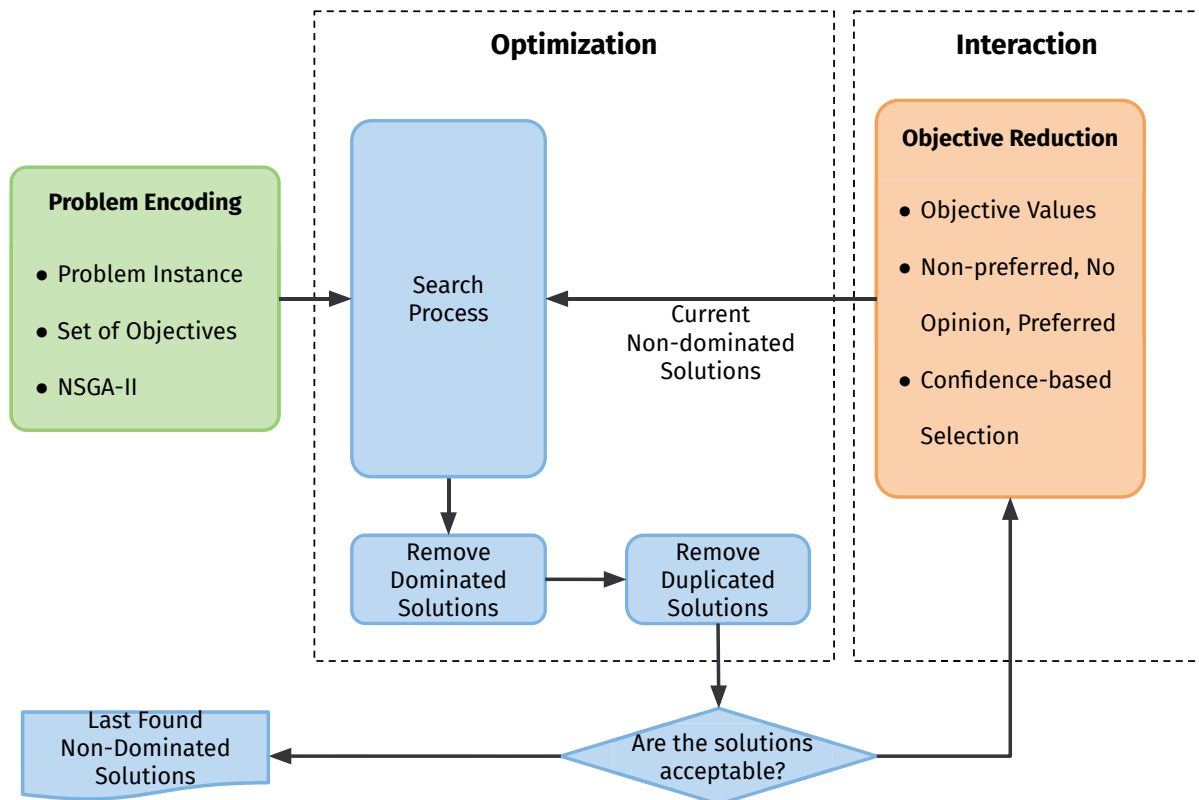


Figure 4.3: COR-NSGA-II Overview.

The figure shows that the algorithm considers NSGA-II as an optimization algorithm, and concerning to the Objective Reduction activity, the elements used are:

- **Items:** Objective Values;

- **Required Information:** *Non-preferred, No Opinion, Preferred*;
- **Selection Method:** Confidence-based Selection.

Regarding Items, the user is required to provide his/her preferences about the objective values in the population. As aforementioned, the user does not need to provide his/her preferences for all of them. The preferences in the Required Information component that the user must provide are, in a high-level, as *Non-preferred, No Opinion, Preferred*. If no preferences are provided, the default preference is *No Opinion*.

Thus, the user feedback required by COR-NSGA-II must be composed of:

**User Feedback** = [solution index | objective index | objective value | required information]

For instance, the user feedback [#1, #2, 0.0, *Preferred*] provided by the user means the user provided for Objective #2 with 0.0 from Solution #2 as *Preferred*, while [#2, #2, 0.7, *Non-preferred*] means the user provided for Objective #1 with 0.7 from Solution #2 a preference as *Non-preferred*.

Concluding, for the Selection Method component, this algorithm uses the Confidence-based Selection. This one is described in more details in the next subsection.

#### 4.2.1 Confidence-based Selection Method

This selection method is based on Stay Out category as described in Section 4.1. Basically, this one calculates a confidence level for all objectives based on the user feedback. Hence, based on a minimum confidence level (a number provided by the user in [0:100] corresponding a percentage of confidence), the objectives with confidence levels greater or equals to the minimum confidence level are removed from the next algorithm execution.

In this method, the Required Information is translated to an ordinal scale described in Table 4.1.

Table 4.1: Ordinal Scale for the Required Information.

<b>Required Information</b>	<b>Value</b>
<i>Non-preferred</i>	-1
<i>No Opinion</i>	0
<i>Preferred</i>	1

The correlation between the Required Information and a number is important once we have to define a priority among them. So, in the case of more than one preference provided for the same Item, we can choose that one with the lowest priority. Thus, the algorithm for selecting the next objectives is described in Algorithm 5.

According to the algorithm, it requires as input a minimum confidence level *minConf* provided by the user, the current population *P*, a set of optimized objectives, *O*, and a set of user feedback *F*.

In the first step (Lines 6-34), the algorithm goes through all user feedback trying to select the minimum and maximum feedback for each objective. Whether two feedback values



---

**Algorithm 5** Confidence-based Selection Method Algorithm
 

---

**Input:** A minimum confidence level ( $minConf$ ) value desired to remove an objective

The current population  $P$

A set of optimized objectives  $O = (o_1, o_2, \dots, o_m)$

A set of user feedback  $F = (f_1, f_2, \dots, f_i)$

**Output:** A subset  $N$  of objectives to be optimized

```

1: Let  $N \subset O$  be the next subset of objectives to be optimized for  $N = \emptyset$ 
2: Let  $MaxF = (maxF_1, maxF_2, \dots, maxF_m)$  be the maximum feedback found by the algorithm for the
   optimized objectives, in which  $\forall maxF_i \in MaxF, maxF_i \leftarrow NIL$ 
3: Let  $MinF = (minF_1, minF_2, \dots, minF_m)$  be the minimum feedback found by the algorithm for the
   optimized objectives, in which  $\forall minF_i \in MinF, minF_i \leftarrow NIL$ 
4: Let  $MinV = (minV_1, minV_2, \dots, minV_m)$  be the minimum values for the population  $P$ .
5: Let  $MaxV = (maxV_1, maxV_2, \dots, maxV_m)$  be the maximum values for the population  $P$ .
6: for all  $f$  in  $F$  do
7:    $i \leftarrow objective\_index(f)$ 
8:    $distToMaxValue \leftarrow |maxV_i - objective\_value(f)|$ 
9:    $distToMinValue \leftarrow |minV_i - objective\_value(f)|$ 
10:  if  $distToMaxValue < distToMinValue$  then
11:    if  $maxF_i$  is  $NIL$  or  $objective\_value(f) > objective\_value(maxF_i)$  then
12:       $maxF_i = f$ 
13:    else if  $objective\_value(f) = objective\_value(maxF_i)$  then
14:      if  $objective\_feedback(f) < objective\_feedback(maxF_i)$  then
15:         $maxF_i = f$ 
16:      end if
17:    end if
18:  else if  $distToMinValue < distToMaxValue$  then
19:    if  $minF_i$  is  $NIL$  or  $objective\_value(f) < objective\_value(minF_i)$  then
20:       $minF_i = f$ 
21:    else if  $objective\_value(f) = objective\_value(minF_i)$  then
22:      if  $objective\_feedback(f) < objective\_feedback(minF_i)$  then
23:         $minF_i = f$ 
24:      end if
25:    end if
26:  else if  $distToMinValue = distToMaxValue$  then
27:    if  $minF_i$  is  $NIL$  or  $objective\_feedback(f) < objective\_feedback(minF_i)$  then
28:       $MinF_i = f$ 
29:    end if
30:    if  $maxF_i$  is  $NIL$  or  $objective\_feedback(f) < objective\_feedback(maxF_i)$  then
31:       $maxF_i = f$ 
32:    end if
33:  end if
34: end for
35: return the objectives selected by Algorithm 6, given  $minConf, O, MaxF, MinF$ 

```

---

were provided for the same objective, the algorithm selects that one with a lower value. As soon as the maximum and minimum are selected, the algorithm calls Algorithm 6.

In this algorithm, for each objective, the preferences provided by the minimum and maximum feedback are defined to the lowest value (*best*) and the highest one (*worst*), respectively (Lines 4-9). After this step, the confidence level for removing this objective is calculated in Line 10 based on the information described in Table 4.2.

**Algorithm 6** Objective Selection Algorithm

**Input:** A minimum confidence level ( $minConf$ ) value desired to remove an objective  
 A set of optimized objectives  $O = (o_1, o_2, \dots, o_m)$   
 A maximum feedback  $MaxF = (maxF_1, maxF_2, \dots, maxF_m)$  for all objectives  
 A minimum feedback  $MinF = (minF_1, minF_2, \dots, minF_m)$  for all objectives

**Output:** A subset  $N$  of objectives to be optimized

```

1: Let  $Best = (best_1, best_2, \dots, best_m)$  be the confidence level for the solutions in the best
   objective values in which  $\forall best_i \in Best, best_i \leftarrow 0$ 
2: Let  $Worst = (worst_1, worst_2, \dots, worst_m)$  be the confidence level for the solutions in the
   worst objective values in which  $\forall worst_i \in Worst, worst_i \leftarrow 0$ 
3: for  $o_i$  to  $O$  do
4:   if  $maxF_i$  is not  $NIL$  then
5:      $worst_i \leftarrow objective\_feedback(maxF_i)$ 
6:   else
7:      $worst_i \leftarrow 0$ 
8:   end if
9:   if  $minF_i$  is not  $NIL$  then
10:     $best_i \leftarrow objective\_feedback(minF_i)$ 
11:  else
12:     $best_i \leftarrow 0$ 
13:  end if
14:  if  $confidence(best_i, worst_i) < minConf$  then
15:     $N = N \cup o_i$ 
16:  end if
17: end for
18: if  $N$  is  $\emptyset$  then
19:   return a random objective from  $O$ 
20: end if
21: return  $N$ 

```

Table 4.2: Confidence Level for removing an objective.

		Highest Value			
		Feedback	Preferred	No Opinion	Non-preferred
Lowest Value	Preferred		0%	20%	0%
	No Opinion		80%	50%	20%
	Non-preferred		100%	80%	100%

The values in this table are proposed in this work and they were generated based on the highest confidence level to the lowest one. For example, as the optimization algorithms tend to optimize towards the lowest values (usually best ones for minimization problems), consider that the user provided *Non-preferred* for both lowest and highest values for a given objective. So, we assume with 100% of confidence level this objective must be removed because the solutions generated with this objective tends not to be good. In the contrary, if both extreme values are

*Preferred*, we have to keep the objectives once the algorithm with them may generate new good solutions. In another example, if the lowest value is *Non-preferred* and the highest is *Preferred*, we assume the algorithm tend to keep generating non-preferred solutions so we have to remove the corresponding objective with 100% of confidence level.

Finally, we have to remove just the objectives in which the confidence level is greater than or equal to the minimum confidence level that was previously defined by the user. However, if this method is carried out and all objectives should be removed (for example, when the user defines that the minimum confidence level is 0%), a random objective must be picked up for the next algorithm execution.

To illustrate the method described in this section, let consider the example shown in Figure 4.4 where the addressed problem has three objectives to be optimized and the non-repeated and non-dominated solutions are composed of five solutions.

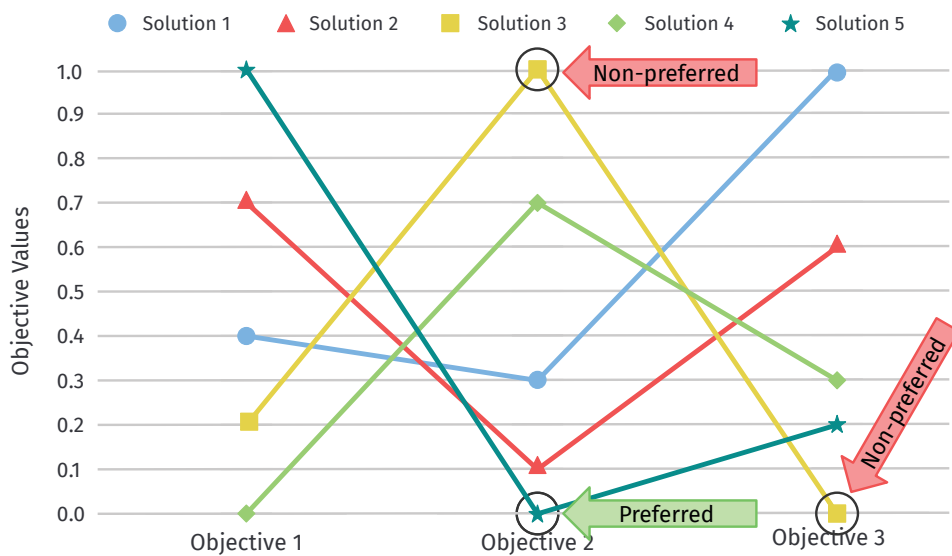


Figure 4.4: Example of application of the confidence level.

In this example, the user provided three feedback. The first feedback was *Preferred* for Solution #5 in Objective 2. The second feedback was *Non-preferred* for Solution #3 in Objective 3. The last feedback was *Non-preferred* for Solution #3 in Objective 3. The selection method aforementioned is performed aiming to find the maximum feedback provided by each objective. In this example, there is no more than one feedback for the lowest and highest objective values. Then, the *Best* and *Worst* sets, and the confidence level (defined by the values in Table 4.2) are show in Table 4.3.

Table 4.3: Confidence Level Example.

Objectives	Objective 1	Objective 2	Objective 3
<b>Worst Value</b>	No Opinion	Non-preferred	No Opinion
<b>Best Value</b>	No Opinion	Preferred	Non-preferred
<b>Confidence Level</b>	50%	0%	80%

Supposing that the minimum confidence level defined by the user for removing an objective for the next execution is, at least, 80%, Objective 3 must be removed once it is associated with a confidence level of 80%. However, if the minimum confidence level is defined as 50%, Objectives 1 and 3 must be removed from the next execution.

This example shows another important property about this selection method. If the minimum confidence level is 100%, the user is required to provide that a given objective has to be *Non-preferred* and *Preferred*, or *Non-preferred* for both to be removed. In an opposite way, if the minimum confidence level is 50% or less, if no user preferences are provided, the selection method considers that this objective is not good (once the user does not express his/her preferences about it), and it must be removed.

### 4.3 FINAL REMARKS

In this chapter we introduced MaDRUP, an approach centered on preference-based dimensionality reduction.

MaDRUP presents some activities and elements to be instantiated to derive algorithms for dimensionality reduction based on user preferences to reduce the number of objectives to be optimized in the next algorithm execution. As an advantage, MaDRUP can use any optimization algorithm in Optimization activity, and it can be applied to any optimization problem once it is not domain-dependent.

By instantiating MaDRUP we derived the algorithm COR-NSGA-II. The main advantage of COR-NSGA-II is that the users do not need to visualize the whole Pareto-front to take a good solution for his/her problem. The user only needs to visualize some solutions and provide his/her feedback for them. After that, the algorithm reduces the number of objectives to be optimized.

MaDRUP and COR-NSGA-II are original since, as far as we are aware, there are no other works or algorithm in the literature that reduce the problem dimensionality based on the user preferences. Besides, to the best of our knowledge, this is the first work actually reducing the number of objectives based on the confidence level defined by the user feedback.

The next chapter introduces Nautilus, a web-based tool developed to evaluate MaDRUP and COR-NSGA-II. Besides, this tool also implements traditional mono-, multi-, and many-optimization algorithms and the preference-based ones.

## 5 NAUTILUS

Nautilus is a free, extendable, and open source Java web platform tool for user feedback capturing, developing and experimenting with several mono-, multi-, and many-objective evolutionary algorithms, optimization algorithms based on reference points, including COR-NSGA-II and other ones with dimensionality reduction based on the user preferences, that can be derived by instantiating MaDRUP (described in the previous chapter). To reach this, Nautilus works with jMetal framework [23] (that is, it is possible to use the algorithms developed by jMetal in Nautilus).

So, this chapter is organized as follows. Section 5.1 presents the motivation about development of Nautilus. Section 5.2 introduces the design goals which guided the implementation of the tool. Section 5.3 presents how Nautilus was built, describing information about its architecture and implementation aspects. Section 5.4 shows how to use Nautilus and some aspects of its user interface. Section 5.5 presents a summary of the Nautilus's key features. Finally, Section 5.6 concludes this chapter.

### 5.1 MOTIVATION

In the literature, we can find some software systems responsible for generating solutions for a given optimization problem. Among them, we can cite the most famous and used, such as PISA framework [8], jMetal [23], and MOEA Framework [35], the last one is also based on jMetal.

Concerning PISA, this software system is a text-based interface and is mainly known by offering some “official” support to some optimization algorithms such as SPEA2, but it is not restricted to that by offering other optimization algorithms. In the same line of thought, jMetal is a Java framework in which several optimization algorithms can also be implemented and adapted for the given optimization problem, widely used in the literature. Additionally, MOEA Framework is a tool based on jMetal in which this one provides the tools necessary to rapidly design, develop, execute and statistically test optimization algorithms.

In the context of this work, the dimensionality reduction based on the user preferences demands a user-friendly interface in which the user can explore the whole Pareto-front (by visualizing the solutions and their variables, and objective values in an uncomplicated manner) and, at the same time, can express his/her preferences about the solutions found by the algorithms.

However, no tools previously described contain these demands. Some of them do not have even an official user interface in which the user can interact. Another important fact is that, although the tools found in the literature are platform-independent ones, no tool is integrated with cloud computing (which could allow scalability) or even available online as web application (supporting reports, user customization of some interface aspects, and so on).

With these considerations in mind, Nautilus was developed aiming to be a tool in which the users can easily implement their optimization algorithms and problems (in an easier way by just importing a plugin to the tool), and visualize the solutions generated by the algorithms with a user-friendly interface. Also, by using Java as programming language and some web technologies, we assume that the tool is portable, it is, the object-oriented features of Java will allow the user to facilitate the code-reuse in the algorithms and problems or extends new ones, and it will also allow scalability once the tool can run in the cloud computing by supporting huge optimization problems, or huge number of objectives, problem instances and so on.

## 5.2 DESIGN GOALS

The purpose of Nautilus is to be a tool that can be used by many researchers to develop their own optimization algorithms with or without objective reduction based on user preferences, and to adapt it to solve his/her optimization problems. Based on that, the main design goals driving Nautilus are described as follows:

- **Simplicity and easy-to-use:** This goal means the user should be able of just selecting the problem and run it with his/her own problem instances. As Nautilus is based on jMetal, some optimization algorithms are already available for the user. Also, this one includes some pre-defined optimization problems and problem instances to be used by the user in which, by using a user-friendly interface and just a few steps, the users are able to just select a optimization problem, open a problem instance and optimize it;
- **Customizability:** This goal aims to provide the ability of some parts of the tool to be customized to suit a particular need from a user. In this context, Nautilus provides a multi-user system in which each one can customize some information and upload to the tool his/her own problem instances to be optimized. Also, the users are able to change some information they visualize about the non-dominated solutions;
- **Portability.** The tool should be executed in machines with different architectures and/or running in a distinct operating system. Nautilus is developed in Java, so it allows to reach this goal;
- **Extensibility:** New optimization algorithms, search operators, and problems should be easily added to the tool. To reach this goal, Nautilus supports plugins in which the users can adapt their needs or context to the tool. Also, the plugin can be written out-of-the-box by just importing some libraries specifically designed to this task;
- **Performance:** Supporting huge problem instances is required to the developed system. Knowing that, Nautilus is a web platform application that allows running it in cloud computing. As advantages, the latter allows automatic software updates, mobility, performance, and so on.

## 5.3 ARCHITECTURE AND IMPLEMENTATION

Figure 5.1 shows all modules and external tools involved in Nautilus. The figure is a simplified version in order to make it understandable.

The figure shows that Nautilus uses three main third-party libraries such as the jMetal framework (previously described) for the optimization algorithms, MongoDB (a general purpose and document-based database) for database, and Spring Boot (application framework and inversion of control container for the Java platform) as web application framework. Regarding to the main modules, Nautilus has basically three ones: *nautilus-core*, *nautilus-plugin*, and *nautilus-web*. Specifically, the *nautilus-plugin-spl* module is the implementation of the problem addressed in this work and it uses *nautilus-plugin* to support it. Thus, the main modules are described in the next sub-sections.

### 5.3.1 Nautilus Core

This is one of the most important modules from Nautilus because it contains the base classes in which the other modules require to work. Figure 5.2 shows the packages inside this module.

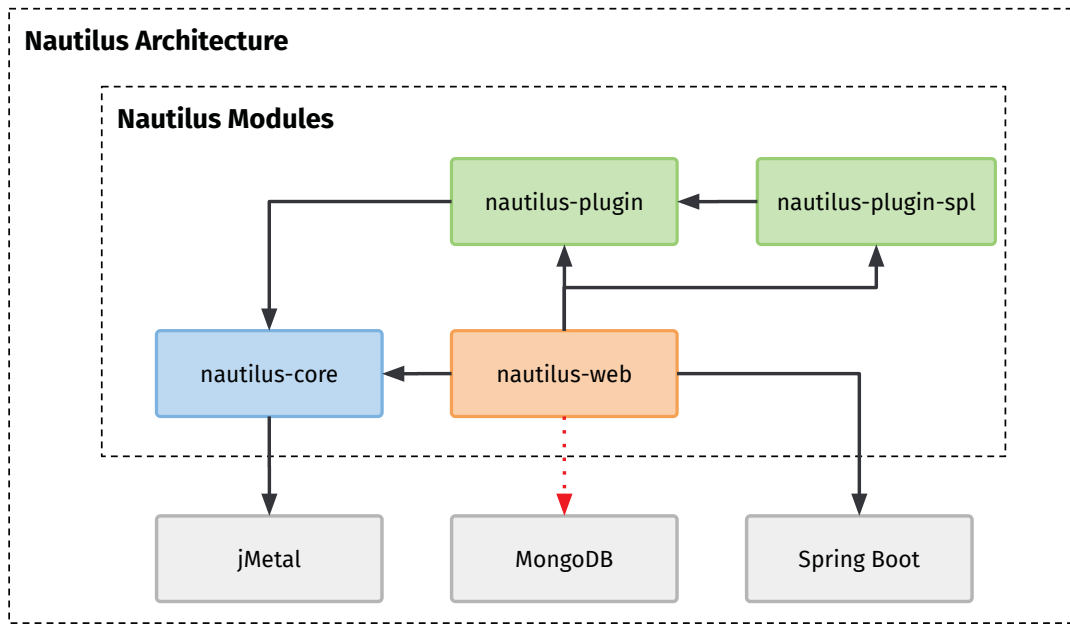


Figure 5.1: Nautilus Architecture.

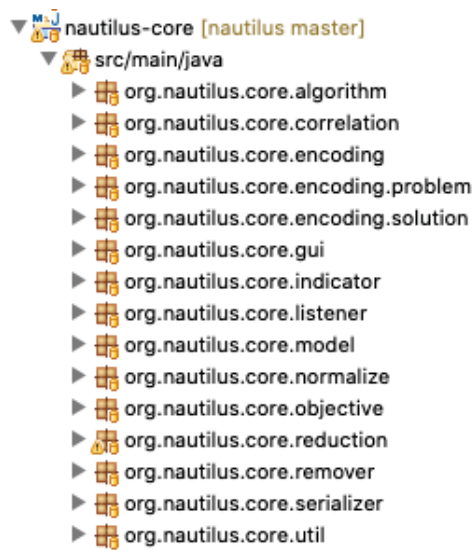


Figure 5.2: Nautilus Core's packages.

The figure shows that this module contains 14 packages. Inside of them, there are some classes that extend other ones from jMetal, mainly to support MaDRUP. For example, in the *org.nautilus.core.encoding* package it is possible to find classes that extend jMetal's classes and add support to dynamic number of objectives (currently, jMetal does not support it).

Moreover, this module provides the classes responsible for defining the encoding type of the problems supported. Currently, Nautilus supports Integer, Double and Binary encoding problems. At the following are described some examples of classes found in this module and a short description about them.

- **AbstractObjective:** Provides the methods in which the objective to be optimized must extend to be used by Nautilus;
- **SolutionListUtils:** An utility class with several methods to manipulate a list of solutions;

- **AbstractReduction:** Provides the methods in which the new objective reduction method must extend.

### 5.3.2 Nautilus Plugin

This module is responsible for providing extensible classes in which the user can create his/her own plugin for Nautilus and adapt his/her needs to the tool. This one uses the classes from *nautilus-core* and Figure 5.3 shows the packages inside this module.

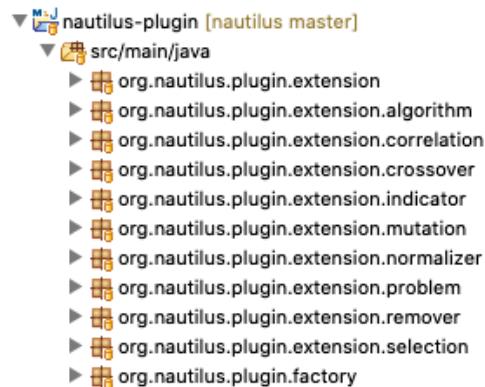


Figure 5.3: Nautilus Plugin's packages.

This module has 9 packages and their names represent basically the piece of Nautilus the user can extend and create his/her plugins. Next, it is shown the most important extensible classes that can be used to extend the tool and incorporate new features, and a short description about them.

- **AbstractAlgorithmExtension:** Provides the methods in which a new optimization algorithm must extend;
- **AbstractCorrelationExtension:** Provides the methods in which a new correlation method must extend. ;
- **AbstractCrossoverExtension:** Provides the methods in which crossover operators must extend;
- **AbstractMutationExtension:** Provides the methods in which mutation operators must extend;
- **AbstractNormalizerExtension:** Provides the methods in which normalization method must extend;
- **AbstractProblemExtension:** Provides the methods in which optimization problem must extend to be addressed in Nautilus;
- **AbstractRemoverExtension:** Provides the methods in which a class responsible for removing the duplicated solutions must extend;
- **AbstractSelectionExtension:** Provides the methods in which selection operators must extend.



For example, the user can extend the *AbstractAlgorithmExtension* class and create his/her own optimization algorithm or even extend the *AbstractProblemExtension* class and create a new optimization problem. Algorithm 7 shows an example of how to extend and create a new algorithm to be used in Nautilus.

---

#### Algorithm 7 Example of an algorithm extension

---

```

1 @Extension
2 public class NSGAIAlgorithmExtension extends AbstractAlgorithmExtension {
3
4     @Override
5     public Algorithm<? extends Solution<?>> getAlgorithm(Builder builder) {
6         return new NSGAI(builder);
7     }
8
9     @Override
10    public String getName() {
11        return "NSGA-II";
12    }
13 }

```

---

### 5.3.3 Nautilus Web

This module provides a user interface based on a web platform to ease the use of the algorithms, visualize the found solutions, and interact with the tool. This module uses *nautilus-core* and *nautilus-plugin* and is developed in Spring Boot by using MongoDB for saving in a database all generated solutions. Figure 5.4 shows the packages inside this module.

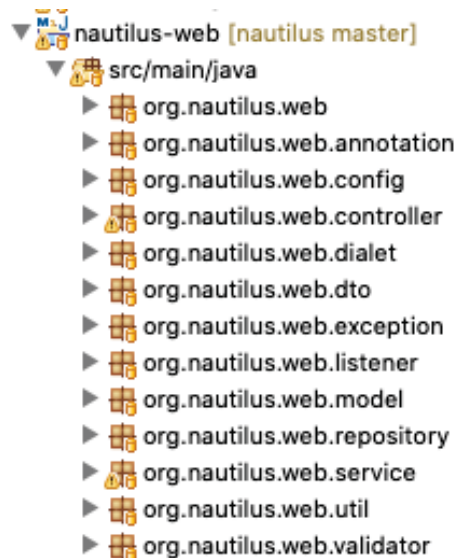


Figure 5.4: Nautilus Web's packages.

The figure shows that this module has 13 packages and most of them extend classes from Spring Boot, such as those ones inside of *org.nautilus.web.repository*, *org.nautilus.web.controller*, and *org.nautilus.web.service* packages.

## 5.4 USING NAUTILUS

In this section, we introduce Nautilus for the variability testing of SPL, the problem addressed in this work and described in Chapter 3.

First of all, Nautilus is a multi-user tool, that is, the users can sign up and create his/her own executions by using the available optimization problems and problem instances. However, Nautilus has, as default, an *admin* user for uploading new plugins and, with this, to support other optimization problems, such as Software Refactoring, Next Release Problem, and so on. In this section, we simulate a default user (non-admin one) that uses the tool.

Figure 5.5 shows the Nautilus' home page. This one is available when the user signs up and successfully logs in the system. This interface contains information about all executions the user has already performed. In Nautilus, an execution is composed of a Pareto-front with non-dominated solutions and the parameters used to find them such as algorithm used, number of evaluations, crossover operators and so on.

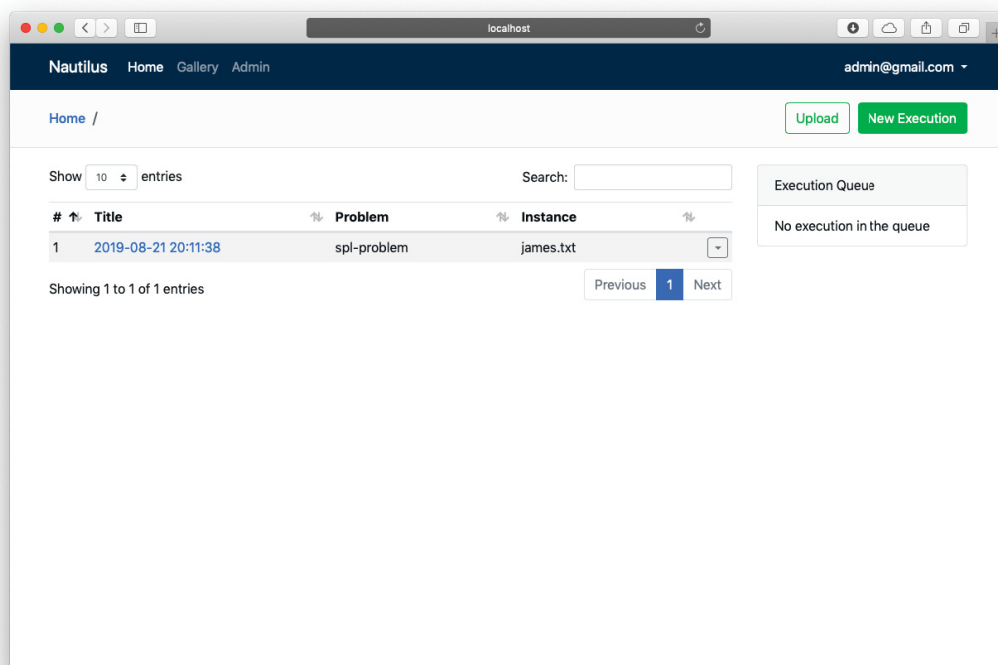


Figure 5.5: Nautilus's screenshot from home page

Besides, this page contains information about all running executions the user launched. Thus, the user can launch several executions at the same time and monitor all of them in this page while they are running. When the execution is done, it appears in the table and the user can open and visualize it.

Nautilus still allows the user to import and export a given execution in some pre-defined file formats such as JSON (a simple data structures and objects in JavaScript Object Notation format [14]), FUN (a file with only the objective values), and VAR (a file with only variables). Once the user is ready to start a new execution, s(he) can just click on the button "New Execution" to select the problem instance to be optimized.

Figure 5.6 shows the page in which all problem instances the user can select are displayed. In this page, all problem instances are grouped by the problem and it is shown information about the problem instance size and the last modified date. If the user is interested in

the optimization of a new problem instance, s(he) can just click on “Upload Instance” button and send it to Nautilus.

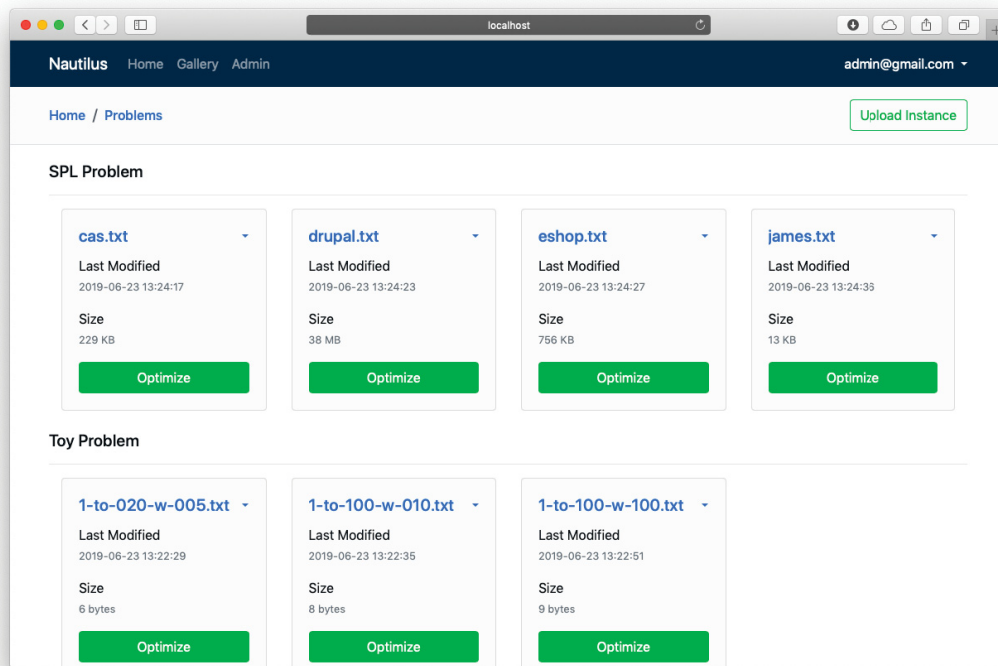


Figure 5.6: Nautilus’s screenshot from problem page.

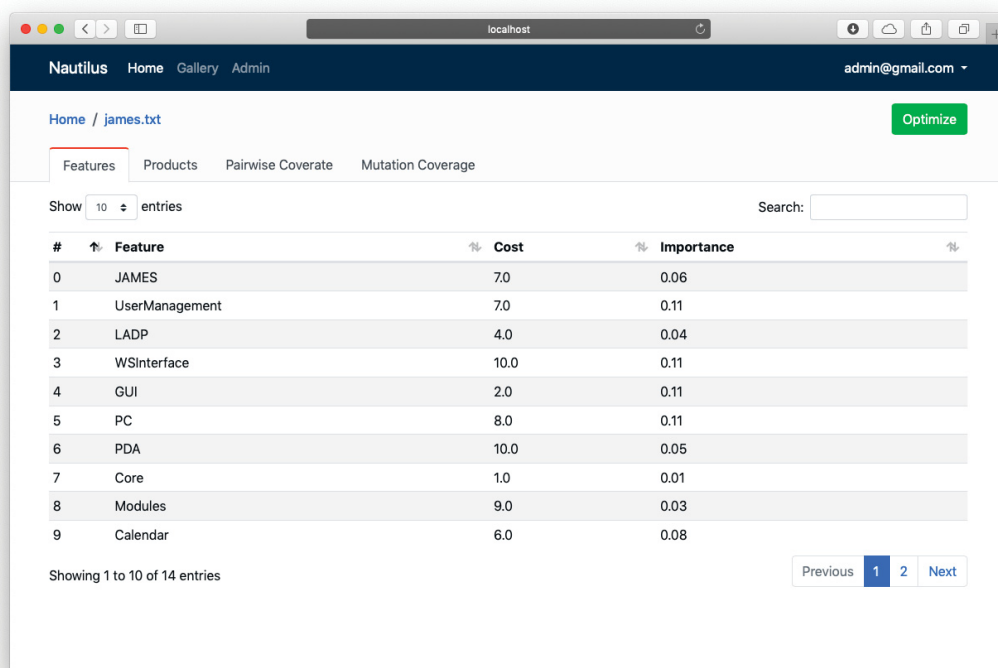
In addition, in this page the user can click on the problem instance name for visualizing the specific information about the problem instance or just skip it and go to the optimization page by clicking on the “Optimize” button.

Next, the interface shows the problem instance page with information about the problem instance to be optimized (Figure 5.7). This one can be customized for each problem by the Nautilus Plugin (see Subsection 5.3.2). In this figure, James was selected and information about the features, products, pairwise and mutation coverage are shown. After visualizing the information, the user can click on "Optimize" button to go to the “optimize” page to set the parameter settings.

Figure 5.8 shows the optimize page. In this page the user is able to select the parameter settings used for optimizing the problem instance. Besides, other information can be set, such as the mating operators, number of evaluations, the population size, the objectives to be optimized and so on. It is important to notice that the parameter settings are different according to the chosen optimization algorithm.

The current version of Nautilus implements the following optimization algorithms and mating operators:

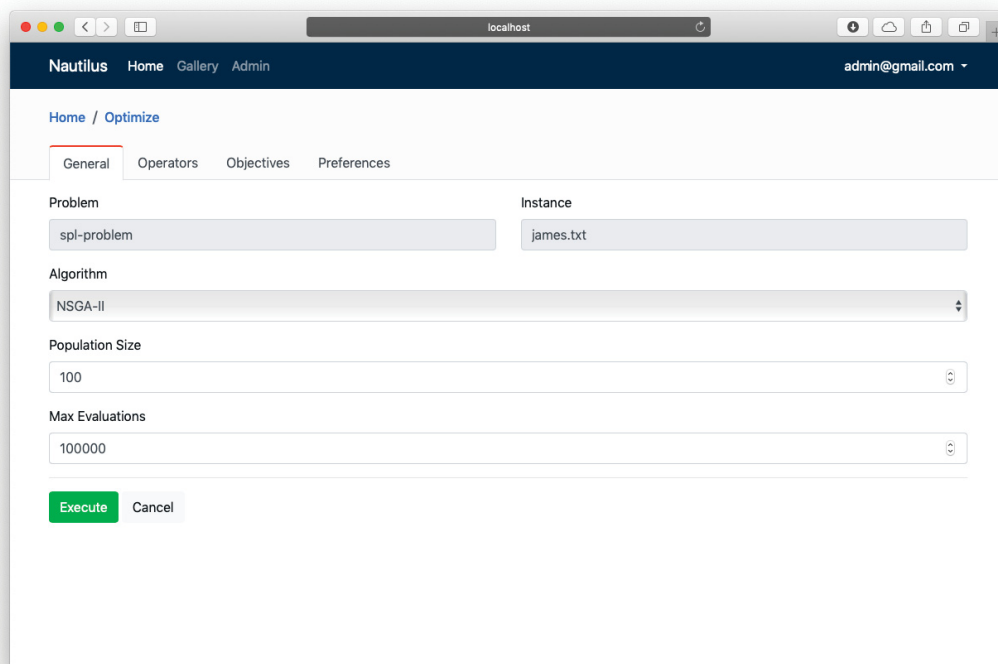
- **Optimization algorithms:** NSGA-II, NSGA-III, GA, R-NSGA-II, COR-NSGA-II, SPEA2 and Random Search as optimization algorithms. There is a Greedy Algorithm but it is limited for some problems and a very small problem instance size;
- **Selection operator:** Binary Tournament with Ranking and Crowding Distance selection;
- **Crossover operator:** Single Point, SBX and Integer SBX crossovers;



The screenshot shows the Nautilus web interface for a problem instance named 'james.txt'. The 'Features' tab is active, displaying a table with 10 entries. The table has columns for '#', 'Feature', 'Cost', and 'Importance'. A search bar and a 'Show 10 entries' dropdown are at the top. A green 'Optimize' button is in the top right. The bottom of the table shows 'Showing 1 to 10 of 14 entries' and pagination controls for 'Previous', '1', '2', and 'Next'.

#	Feature	Cost	Importance
0	JAMES	7.0	0.06
1	UserManagement	7.0	0.11
2	LADP	4.0	0.04
3	WSInterface	10.0	0.11
4	GUI	2.0	0.11
5	PC	8.0	0.11
6	PDA	10.0	0.05
7	Core	1.0	0.01
8	Modules	9.0	0.03
9	Calendar	6.0	0.08

Figure 5.7: Nautilus's screenshot from problem instance page.



The screenshot shows the Nautilus web interface for the 'Optimize' page. The 'General' tab is active. The 'Problem' field is set to 'spl-problem' and the 'Instance' field is set to 'james.txt'. The 'Algorithm' dropdown is set to 'NSGA-II'. The 'Population Size' is set to 100 and the 'Max Evaluations' is set to 100000. There are 'Execute' and 'Cancel' buttons at the bottom.

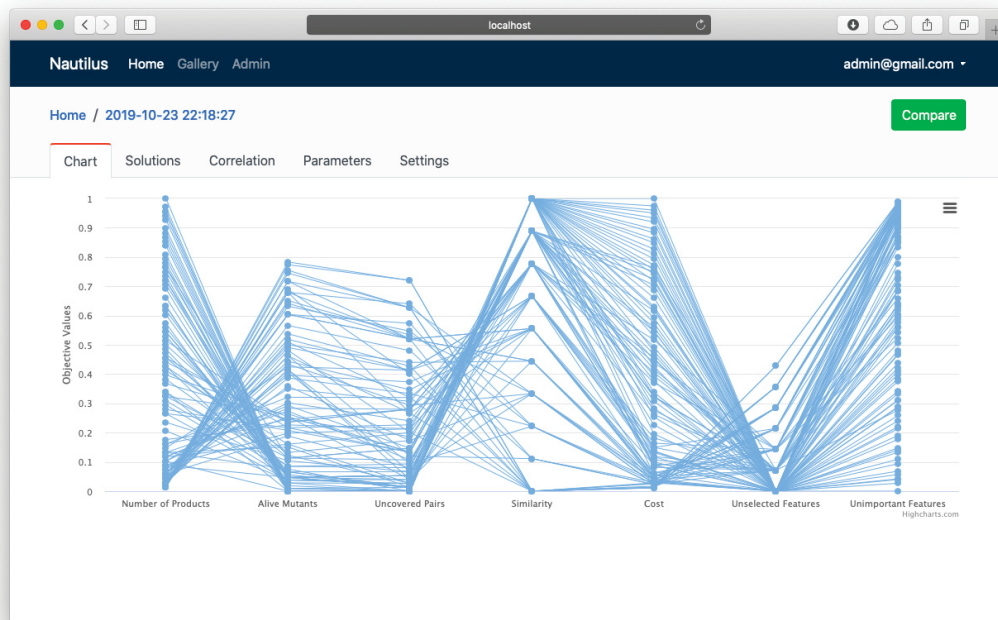
Figure 5.8: Nautilus's screenshot from optimize page.

- **Mutation operator:** Bit Flip, Integer Polynomial and Polynomial mutations.

Nautilus still supports R-NSGA-II and, in this page, it is also possible to add the reference points used by the algorithm. Once the parameter settings are set, the optimization starts and the

page redirects to the home page (see Figure 5.5) where the user can monitor the progress of this execution.

In the next step, Figure 5.9 the non-dominated solutions are presented. In this page the user can visualize the solutions either by using a chart or a table with the solutions and their objective values.



(a) Line chart representation

The screenshot shows the Nautilus application interface with the 'Solutions' tab selected. The table displays 11 entries of solutions. The columns are: #, Number of Products, Alive Mutants, Uncovered Pairs, Similarity, Cost, Unselected Features, Unimportant Features, and Selected. The 'Selected' column contains 'False' for all entries. The table also includes a search bar, a 'Show 10 entries' dropdown, and a pagination control at the bottom showing 'Showing 1 to 10 of 106 entries' and a 'Previous 1 2 3 4 5 ... 11 Next' navigation bar.

#	Number of Products	Alive Mutants	Uncovered Pairs	Similarity	Cost	Unselected Features	Unimportant Features	Selected
0	0.0147	0.7736	0.7200	0.0000	0.0112	0.4286	0.9879	False
1	1.0000	0.0000	0.0000	1.0000	1.0000	0.0000	0.0000	False
2	0.0294	0.4811	0.2000	0.0000	0.0287	0.0000	0.9707	False
3	0.1029	0.0000	0.0000	1.0000	0.0997	0.0000	0.9011	False
5	0.3382	0.0849	0.0800	1.0000	0.3248	0.0000	0.6838	False
7	0.0147	0.7830	0.7200	0.0000	0.0115	0.4286	0.9877	False
8	0.0294	0.7170	0.6267	0.3333	0.0235	0.3571	0.9725	False
9	0.0294	0.6038	0.5733	0.1111	0.0230	0.2857	0.9764	False
10	0.0294	0.6792	0.6400	0.2222	0.0227	0.3571	0.9756	False
11	0.0147	0.6887	0.4000	0.0000	0.0144	0.2143	0.9822	False

(b) Table representation

Figure 5.9: Nautilus's screenshot from execution page.

Furthermore, Nautilus is able to perform the correlation among the objectives. Currently, the tool supports Kendall, Pearson and Spearman correlations [86]. Another important feature in this page is the ability to change some displayed information such as chart's color, remove the duplicated solutions from Pareto-front before show it, normalize the objective values and change de correlation type. So, to open the solution and visualize it, it is necessary just to click in the circle on the chart.

As a result, the tool presents information about the selected solution as illustrated in Figure 5.10. The figure shows the variables from the selected solution and its objective values (raw and normalized ones). Also, the user can provide his/her preferences about that by just sliding left or right the component below the objective values.

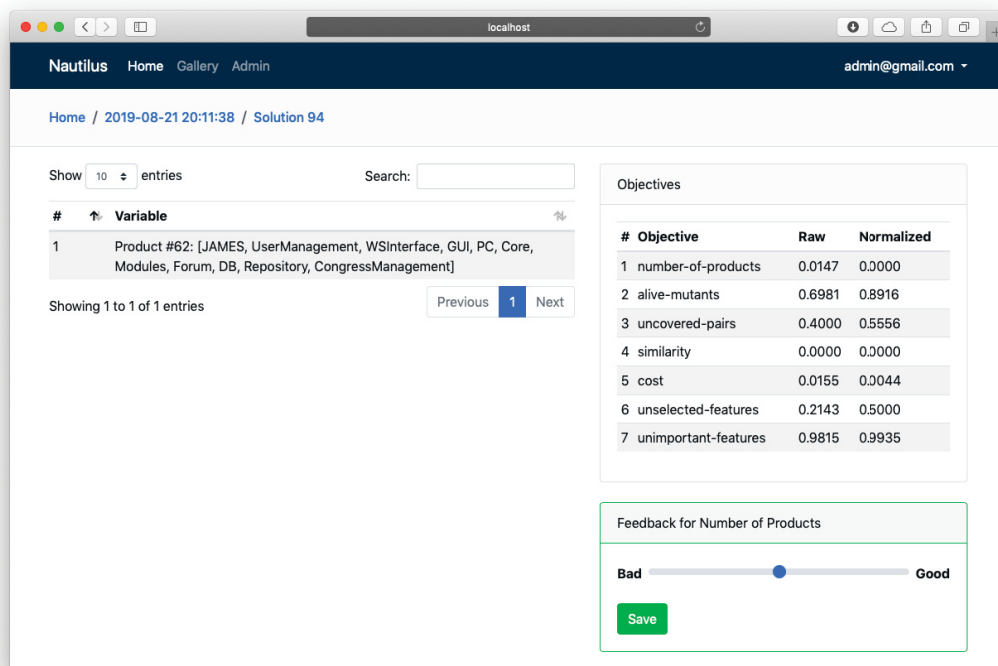


Figure 5.10: Nautilus's screenshot from solution page.

After the process of optimizing and picking a solution up as the best one, the user can compare all of the selected solutions in a single chart. The page is available in Figure 5.11 and, in this example, it is shown four solutions selected by the user as the best ones for NSGA-II, COR-NSGA-II, Manual selection and R-NSGA-II. Besides, in this screen it is possible to visualize some information about the quality attributes for the selected solutions such as R-Hypervolume, R-IGD and so on.

Finally, Figure 5.12 shows the customization page. In this one the user is able to change information about the decimal places, decimal separator, the language used by the tool and the time zone.

## 5.5 AVAILABLE FEATURES

In this section, we describe a summary of the most important features previously described and other minor ones that Nautilus implements.

1. A web application platform;

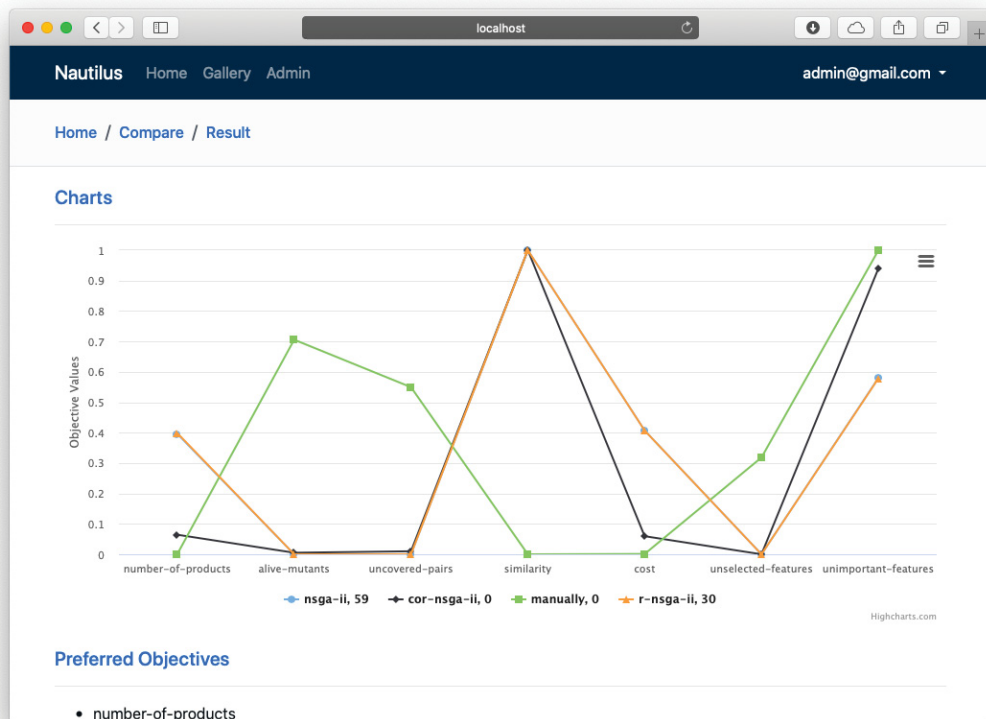


Figure 5.11: Nautilus's screenshot from compare page.

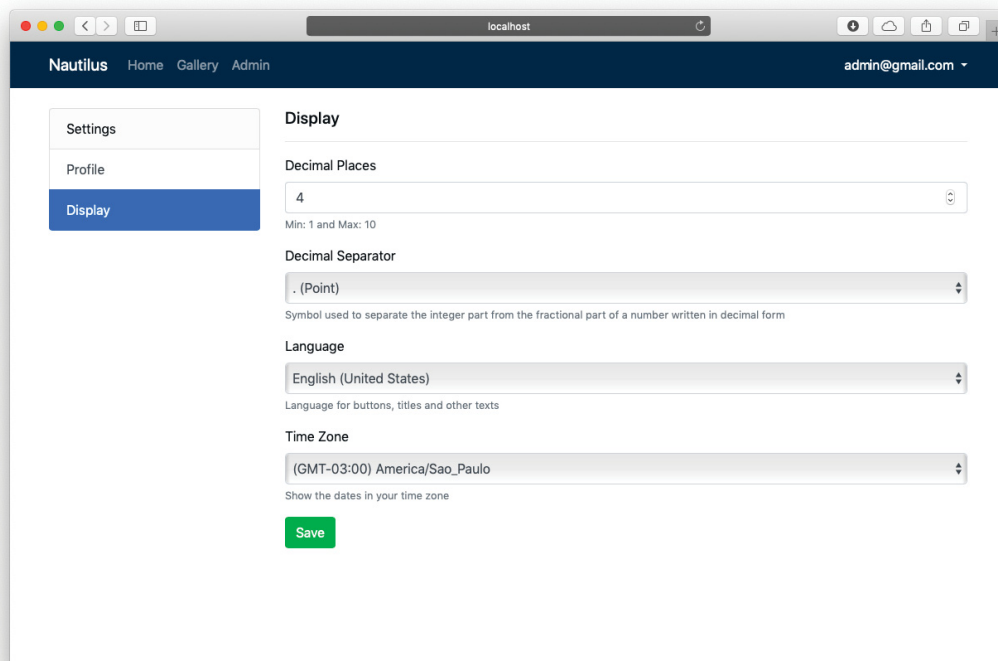


Figure 5.12: Nautilus's screenshot from customization page.

2. Extensible through plugins;
3. Several optimization algorithms and mating operators are available;

4. Support to multi-users with roles and permissions for each one;
5. Gallery support to share the execution with other users;
6. Multi-language support (currently Portuguese and English ones);
7. Customization (for example decimal separator and places);
8. Support to Integer, Double and Binary encoding types;
9. Mobility to see the executions from anywhere;
10. Calculate some quality indicators such as R-HV and R-IGD;
11. Support both MaDRUP and COR-NSGA-II proposed in this work.

## 5.6 FINAL REMARKS

This chapter presented Nautilus and its main implementation aspects and features, such as its modules and pages.

Firstly, the motivation for the Nautilus development was presented by showing that the found tools in the literature do not provide the features required by this work. So, based on that, some design goals were described in which all development was guided by them.

Next, the main modules of Nautilus were detailed and the responsibility of each one and how they are integrated were presented. The packages inside of them were also displayed and the main classes were shown and briefly described.

Last but not least, the user interface was presented, explaining how the user can interact by setting the parameters, visualizing the solutions and providing his/her preferences. Details about each one was provided and the key features were described.

Nautilus was developed aiming to support MaDRUP and the COR-NSGA-II algorithm described in the previous chapter and to offer a user-friendly interface in which, by using a few steps, the users are able to select solutions very quickly which meet their expectations.

Besides, the tool and COR-NSGA-II are publicly available and its source code can be found at <https://github.com/thiagodnf/nautilus>.

Based on that, Nautilus was used for evaluating MaDRUP and COR-NSGA-II with real users in an empirical study on the problem of selecting products for SPL testing introduced in Chapter 3. The experiment and results are described in the next chapter.



## 6 EMPIRICAL STUDY

The hypothesis of this work is “*a preference-based dimensionality reduction approach is capable of taking less execution time and generating a reduced set of solutions that takes into account the user preferences. In addition to this, the solutions are as good as those ones generated by multi- and many-objective algorithms with respect to quality indicators from the literature.*”. To investigate this hypothesis, we conducted an evaluation by using six real-world FMs and compared the results found by COR-NSGA-II to those ones obtained by using four multi- and many-objective evolutionary algorithms found in the literature.

This chapter describes the evaluation conducted and is organized as follows. Section 6.1 presents the Research Questions (RQs) derived and how we designed the experiments to answer them. Section 6.2 presents the target FMs and their characteristics. Section 6.3 describes the kind of users considered to determine the preferences used for answering the RQs. Section 6.4 introduces the quality indicators used for evaluating the RQs. Section 6.5 describes how the reference points used in R-Metric and R-NSGA-II algorithm were chosen. Section 6.6 shows the parameters used by all algorithms. The results and discussion are presented in Section 6.7 and Section 6.8, respectively. The threats to the validity of the obtained results are shown in Section 6.9. Finally, 6.10 concludes this chapter.

### 6.1 RESEARCH QUESTIONS

Considering the goal of this work, the empirical study was guided by the following research questions:

- RQ 1:** *Is COR-NSGA-II capable of reducing the problem dimensionality towards the user preferences?* The goal of this RQ is to evaluate if the Confidence-based selection method of COR-NSGA-II is better than a random selection method (sanity check). To support this analysis, the algorithm was executed asking the preferences to a simulated user (explained in more details in Section 6.3) and the results were compared concerning to Reduction Efficiency, the Number of Preferred Objectives in the Last Subset, and Reduction Capacity (see Section 6.4).
- RQ 2:** *How are the results of COR-NSGA-II compared to those ones obtained by multi- and many-objective evolutionary algorithms?* This RQ aims to compare the proposed algorithm to those ones that use reference-set based, preference-based, or, dimensionality reduction approaches for solving many-objective problems. To reach this, the algorithm was executed by using preferences provided by a simulated user, and a quantitative analysis was performed by using R-HV, R-IGD, # of solutions generated (and the number of them in the ROI), and the execution time. So, to answer this RQ the following sub-questions were considered:
- RQ 2.1:** *How are the results of COR-NSGA-II compared to NSGA-II and NSGA-III, a traditional and reference-set based algorithms, respectively?*
- RQ 2.2:** *How are the results of COR-NSGA-II compared to R-NSGA-II, a preference-based algorithm?*
- RQ 2.3:** *How are the results of COR-NSGA-II compared to PCA-NSGA-II, a dimensionality reduction based algorithm?*

**RQ 3:** *Can COR-NSGA-II help users to find useful solutions?* The goal of this research question is to evaluate if the solutions generated by COR-NSGA-II are more preferred than those ones generated by the other multi- and many-objective evolutionary algorithms. To achieve this goal, a set of potential users were invited and asked to select a good solution in their point of view. The analysis conducted is based on a qualitative questionnaire available in Appendix A.

**RQ 4:** *Can Nautilus be useful for the users in the task of selecting a good solution?* In this research question, the goal is to evaluate the proposed tool regarding its applicability as a tool to support the decision-making process. To reach this, the same set of users that participated responding the questionnaire of **RQ3** also responded another questionnaire to evaluate Nautilus. Such a questionnaire is in Appendix B.

## 6.2 TARGET FEATURE MODELS

This work uses six FMs already used in related work [25, 28, 43, 62, 74], in which five of them were extracted from the SPLOT repository [56]. Details about them can be found in Appendix C. These FMs are:

- a) **James:** SPL for collaborative web systems [6];
- b) **CAS (Car Audio System):** a SPL to manage automotive sound systems [84];
- c) **WS (Weather Station):** SPL for weather forecast systems [7];
- d) **E-Shop:** an E-commerce SPL [71];
- e) **Drupal:** a modular open source web content management framework [62];
- f) **Smarthome v2.2:** SPL for a smart residential solution [43].

Table 6.1 shows information about each FM, such as number of products ( $n_t$ ), number of used products  $n$ , alive mutants ( $AM$ ), valid pairs ( $VP$ ), and number of features (# of Features). We can observe that the last two FMs contain a larger number of features and products. Due to this, it is impractical to use all the products in the population representation. For both FMs  $n$  products were randomly selected from the total number of products  $n_t$  that can be derived.

Table 6.1: Characteristics of the FMs used in the experiments.

FM	$n_t$	$n$	$AM$	$VP$	# of Features
James	68	68	106	75	14
CAS	450	450	227	183	21
WS	504	504	357	195	22
E-Shop	1152	1152	94	202	22
Drupal	$\approx 2.09E9$	11k	2194	1081	48
Smarthome	$\approx 3.87E9$	11k	2948	1710	60

In this work, James, CAS, WS, and E-Shop are considered small instances, while Drupal and Smarthome are the larger ones.

All FMs were used for evaluating **RQ1** and **RQ2**. However, as the users were required to express their preferences, E-Shop was chosen for answering **RQ3** and **RQ4**, because it is widely used in the literature [28, 30, 32, 33], it is composed of a reasonable number of products to be selected, and it has a suitable execution time for experiments with users.

### 6.3 USERS

To evaluate the aforementioned RQs, we used real users and simulated ones. Both of them are described in the following sub-sections.

#### 6.3.1 Simulated Users

The simulated user is a user simulator developed for answering **RQ1** and **RQ2**, aiming to represent a possible evaluation profile, as explored in other work of the literature [2, 27, 72]. In this method, when a user preference is required for a given solution, COR-NSGA-II asks it for the user simulator.

It is important to notice that the main objective of this simulator is not a faithful representation of a human being, but it demonstrates the influence of a certain evaluation profile in the search process. This simulator is summarized in Figure 6.1.

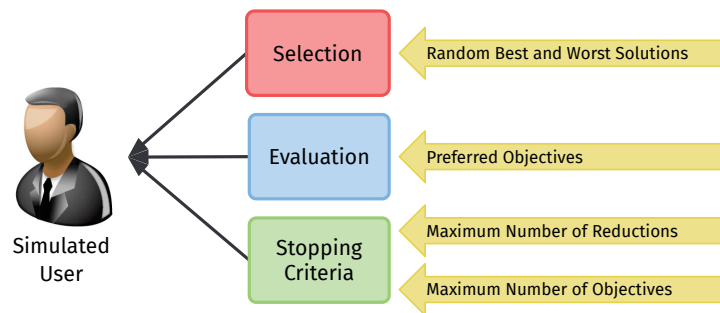


Figure 6.1: Simulated user representation.

The user simulator requires a set of preferred objectives (a subset of those ones to be optimized) and it supposes that the population is normalized in  $[0:1]$  in which 0.0 means the best value and 1.0 the worst one for every objective. So, this one is grouped in three main components: selection, evaluation, and stopping criteria briefly described below.

The first module is responsible for selecting the items for evaluation required by COR-NSGA-II and described in Section 4.2. When this one is performed, for each objective, all solutions from the non-dominated population that have the best and the worst values are selected. After that, a set of random solutions from this group (in this context, called items for evaluation) is picked up to be evaluated after by the user simulator.

In the second component, these items are evaluated. This one is responsible for evaluating the items proposed according to the preferred objectives previously defined for the user simulator. Algorithm 8 shows the algorithm used for evaluating the items.

The algorithm first verifies if the objective index of the item for evaluation is part of the preferred objectives. If it is not, this item is marked as *Non-preferred*. Otherwise, it is marked as *Preferred* if the objective value is 0.0 or the maximum and minimum objective values are the same, and as *Non-Preferred* if the objective value is 1.0. In the last case, if no previous conditions were reached, it is marked as *No Opinion*.

---

**Algorithm 8** Preferred Objectives Algorithm
 

---

**Input:** A set of preferred objectives  
Items proposed for evaluation

**Output:** The user feedback

```

1: for all item in items of evaluation do
2:   if the preferred objectives contain the objective index then
3:     if the minimum and maximum values for this objective are the same then
4:       save the feedback as Preferred
5:     else
6:       if the objective value = 0.0 then
7:         save the feedback as Preferred
8:       else if the objective value = 1.0 then
9:         save the feedback as Non-preferred
10:      else
11:        save the feedback as No Opinion
12:      end if
13:    end if
14:  else
15:    save the feedback as No-preferred
16:  end if
17: end for
  
```

---

Finally, the third module is responsible for defining the stopping criteria considered by the simulator. In this one, the used criteria are the maximum number of interactions or the maximum number of objectives is reached.

To illustrate the user simulator, consider the non-dominated population shown in Figure 6.2.

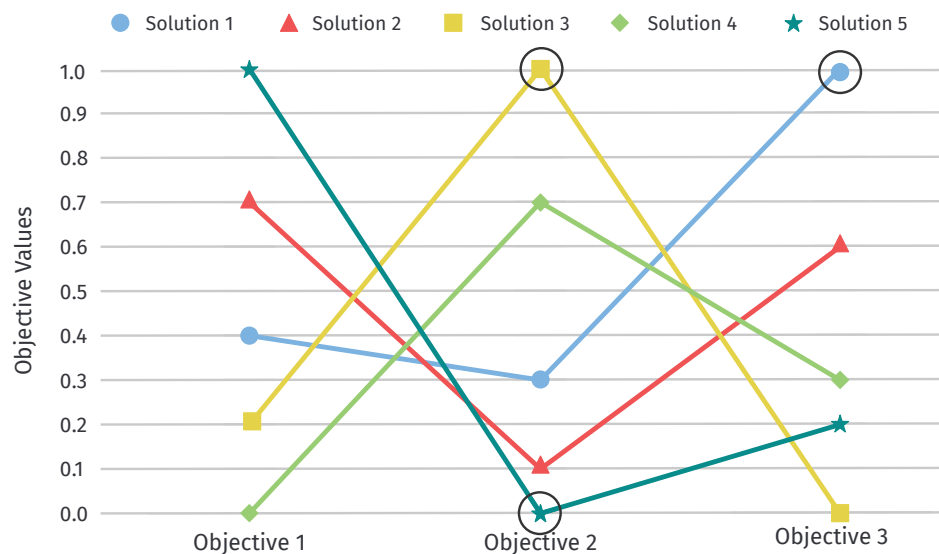


Figure 6.2: Example for user simulator evaluation.

In this example, the indexes of the preferred objectives are #1 and #2. So the items for evaluation are these ones:

$Item_1 = [\text{solution: \#3} \mid \text{objective index: \#2} \mid 1.0]$

$Item_2 = [\text{solution: \#1} \mid \text{objective index: \#3} \mid 1.0]$

$Item_3 = [\text{solution: \#5} \mid \text{objective index: \#2} \mid 0.0]$

Performing the evaluation component in this example, the user preferences provided by the user simulator for  $Item_1$ ,  $Item_2$ , and  $Item_3$  are respectively, *Non-preferred*, *Non-preferred*, and *Preferred*.

The user simulator requires a set of preferred objectives. Based on this, two scenarios were designed and evaluated in **RQ1** and **RQ2**. The first scenario (called *Scenario 2D*) is responsible for simulating a user who prefers two objectives, and the second one (called *Scenario 3D*) simulates a user who prefers three objectives. For *Scenario 2D*, *Number of Products* and *Alive Mutants* objectives are randomly defined as preferred ones. Regarding *Scenario 3D*, *Alive Mutants*, *Similarity*, and *Cost* objectives are selected as preferred ones.

The choice of these scenarios and the objectives selected for them was based on the correlation among the objectives, seeking to simulate a possible preferred set of objectives. For example, in *Scenario 2D*, all objectives are conflicting ones. Concerning to *Scenario 3D*, two of the selected objectives are redundant ones: *Similarity*, and *Cost*.

### 6.3.2 Real Users

For answering **RQ3** and **RQ4**, we asked to a group of potential users of Nautilus to run and evaluate the solutions generated by COR-NSGA-II and the other multi- and many-objective evolutionary algorithms. To reach this, Nautilus was used in all experiments involving the users and they were invited to assist the solution generation process by using the tool. So, they could state their preferences about the found solutions.

Our study involved 12 participants from the Federal University of Paraná (UFPR) and Federal University of Technology - Paraná (UTFPR) to use and evaluate our tool. Participants include 3 Master students and 7 Ph.D. students in Software Engineering and Optimization Algorithms, and 2 professors. All the participants are volunteers and familiar with the subject of this work. The experience in years of these participants on programming ranged, in general, from 2 to 10 years.

The participants were first asked to fill out a Participant questionnaire (available in Appendix D). This questionnaire helped to collect background information such as their role within the company, their programming experience, their familiarity with software testing and so on. Figure 6.3 shows some specific information about the participants.

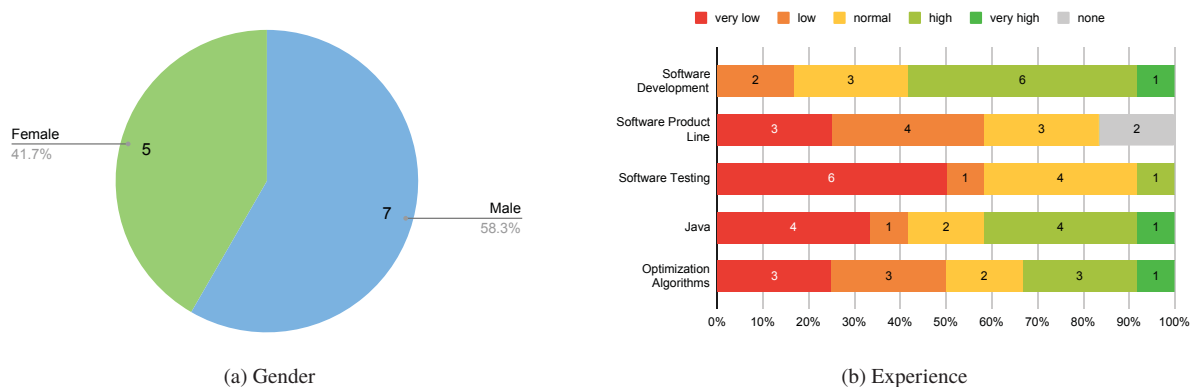


Figure 6.3: Results from the Participant Questionnaire.

As we can see in the figure, the experiment involved 7 male and 5 female participants, and around 50% of the participants have high experience in software development and optimization algorithms.

In addition, all the participants attended one lecture about the variability testing of SPL and optimization algorithms, and, at the end, a test with five questions (a pre-study questionnaire available in Appendix E) was filled out aiming to evaluate their performance and understanding of the subject for suggesting good solutions for an example of problem instance.

For **RQ4**, every participant was invited to interact with Nautilus to become familiar with it. The idea of this scenario is to perform an evaluation about the tool developed and, at the same time, to avoid that the lack of knowledge about the tool influences his/her decision about the generated solutions. In this scenario, a toy problem was used and the user was invited to fill out a questionnaire (available in Appendix B) about his/her impressions by using the tool.

For **RQ3**, we formed 4 groups in which each one is composed of 3 participants. These groups were formed based on the pre-study questionnaire and the test results to make sure that all the groups have almost the same average skills. For the test results, every question is weighted and we tried to form the group in which all users have a similar average. As a result of this pre-study, all participants reached a good performance by selecting the correct answer in almost all questions. Because of this, the participants were assigned randomly in each group, since all of them were at the same level.

We asked every participant to define based on his/her preferences a set of preferred objectives and, following the sequence described in each group, the algorithms were executed. For each algorithm, the user was required to select/provide a good solution based on his/her preferences previously defined. In this scenario, a questionnaire (available in Appendix A) was provided and the participants were asked to justify their evaluation about his/her decisions and these justifications are reviewed by the organizers of the study. Table 6.2 summarizes the groups organization including the list of algorithms evaluated by each one.

Table 6.2: Groups Organization.

Group 1	Group 2	Group 3	Group 4
Manual Selection	NSGA-II	R-NSGA-II	COR-NSGA-II
NSGA-II	R-NSGA-II	COR-NSGA-II	Manual Selection
R-NSGA-II	COR-NSGA-II	Manual Selection	NSGA-II
COR-NSGA-II	Manual Selection	NSGA-II	R-NSGA-II

The table shows the sequence for execution of the algorithms used by the participants in each group. For example, the participants of Group 2 are invited to execute the NSGA-II, R-NSGA-II and COR-NSGA-II algorithms and, at the end, to generate a solution for the problem instance by using manual selection.

Furthermore, the quality indicators used in this work to answer the research questions are described in the next sub-sections.

#### 6.4 QUALITY INDICATORS

The analysis of **RQ1** and **RQ2** was conducted by using the quality indicators described in Section 2.7 along with the execution time. However, COR-NSGA-II generates a non-dominated population with the same number of objectives used in the last reduction. So, it makes necessary

to evaluate again  $PF_{true}$  with the same objectives used at the beginning of the execution. To cope with this, when COR-NSGA-II stops the reduction process, all solutions in  $PF_{true}$  are re-evaluated with the same objectives described in Subsection 3.4.2 and, so, it is possible to compare all algorithms once all of them have solutions with the same set of objectives.

In addition to this, the quality attributes described in the next subsections were also used for evaluating **RQ1** and **RQ2**. Some of them are introduced in this work.

#### 6.4.1 Average Number of Solutions in the ROI

This quality attribute calculates how well an algorithm can generate solutions in the ROI, that is, the main idea of this quality attribute is to evaluate the algorithm ability to obtain concentrated solutions that satisfy the user preferences.

Again, in order to perform the calculation of this metric, the average was calculated considering the sets  $PF_{approx}$  formed by each algorithm and, for the determination of the ROI, we made use of R-Metric previously described. Thus, in the context of this work, lower values of this quality indicator represent the best results, once we want to reduce the number of generated solutions in the ROI.

#### 6.4.2 # of Targets in the Last Subset

This quality indicator counts the number of preferred objectives in the last reduction. Ideally, this metric should return the same number of preferred objectives defined by the user to make sure the dimensionality reduction is moving towards the user preferences.

#### 6.4.3 Reduction Capacity

This quality indicator calculates the percentage in which the reduction mechanism reached the preferred objectives. For instance, if this metric indicates that it has 100%, it means, in all executions, the algorithm reaches the preferred objectives defined by the user. Otherwise, this metric scores 0%. So, the greatest value is better.

#### 6.4.4 Reduction Efficiency

This quality indicator returns the number of reductions performed until the final set of objectives includes just the preferred ones. A less number is better, since this means the algorithm has a faster convergence, but 0 means the algorithm never reached the preferred objectives.

#### 6.4.5 Execution Time

The execution time is also used for assessing the algorithms. In this indicator, the time spent on generating the final Pareto-front set is calculated. However, for preference-based algorithms such as COR-NSGA-II, the time spent on providing the user preferences is also taken into consideration. Thus, a less number is better.

### 6.5 DEFINITION OF THE REFERENCE POINTS (RP)

The R-metric process described in Subsection 2.7.1 and the R-NSGA-II algorithm require a reference point (RP) used for calculating the ROI and finding solutions close to the RP respectively. In this case, the RP also means a preference for the objectives.

A RP is defined as a tuple  $RP = (N, M, P, V, C, F, I)$  where the sequence of elements represents either a value (or a point) in the objective space in which the ROI should be generated or a point in which the algorithm should concentrate its search mechanism. So, in this tuple,  $N$  is the value for the objective *Number of Products*,  $M$  is the value for *Alive Mutants*,  $P$  is the value for *Uncovered Pairs*,  $V$  is the value for *Similarity*,  $C$  is the value for *Cost*,  $F$  is the value for *Unselected Features*, and  $I$  is the value for *Unimportant Features*.

Knowing this, two kinds of RPs were defined to be used by R-metric and R-NSGA-II algorithm in **RQ2**: i) restricted, and ii) compromised. The former is responsible for representing a preference restricted to a specific set of preferred objectives, and the other objectives (non-preferred one), are assigned “no preferences”. The latter is responsible for also representing a preference for a specific set of preferred objectives, but intermediate values for the other objectives are also defined.

Thus, two RPs were defined for each scenario described in Subsection 6.3.1 taking into consideration the preferred objectives of them. Then, the RPs used to calculate the R-metric and to run the R-NSGA-II algorithm are show in Table 6.3.

Table 6.3: Reference Points.

Scenario	Type	Reference Point
Scenario 2D	Restricted	(0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0)
	Compromised	(0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.5)
Scenario 3D	Restricted	(1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0)
	Compromised	(0.5, 0.0, 0.5, 0.0, 0.0, 0.5, 0.5)

The table shows that, for Scenario 2D, the restricted RP aims to express preferences for *Number of Products* and *Alive Mutants* with 0.0 value, and no preferences (a value of 1.0) for the other objectives, while in the compromised RP, a 0.5 value is defined for the other non-preferred objectives. Concerning Scenario 3D, the RP aims to express the preference for *Alive Mutants*, *Similarity*, and *Cost* objectives with 0.0 value for the restricted RP and 1.0 for the non-preferred ones, while in the compromised RP, 0.5 is defined for the non-preferred objectives.

## 6.6 PARAMETER SETTINGS

The type and values for crossover and mutation probabilities already defined in related work for the same problem and FMs [28, 33, 54, 74]. Then, we adopt the same probability rates for all algorithms being 90% for crossover probability and 0.5% for mutation one.

Regarding the population size and the maximum number of evaluations (considered as a stopping criterion), a tuning phase was performed and we tested two settings for these values: 112 and 238 for population size and 134,400 (or each solution is going to be evaluated 1200 times) and 238k (or each solution is going to be evaluated 1000 times) for maximum number of evaluations.

The population size was defined based on the mechanism used by NSGA-III. The latter uses a set of reference points on a hyper-plan where, by using  $p$  divisions along each objective, the number of reference points (and consequently the population size) is calculated by  $H = \binom{M+p-1}{p}$ , where  $M$  is the number of objectives [19].

Other specific parameters were also tuned. For example,  $\epsilon$  for R-NSGA-II, was evaluated with 0.0001 and 0.001 (the same in [33]) where this one controls the number of solutions inside



the ROI. Concerning COR-NSGA-II, the minimum confidence level (80% and 100%), the number of items for evaluation (5 and 10 items) and the number of reductions (5 and 10 ones) were also evaluated. In addition to this, R-Metric requires a  $\delta$ , a parameter that specifies the ROI's size. For this work, the value of 0.3 was used. This value is the same used in the previous work for the addressed problem [32, 33].

It is also important to notice that for **RQ1** and **RQ2**, COR-NSGA-II divides the maximum number of evaluations by the number of reductions. It means COR-NSGA-II is going to perform the same number of evaluations of other algorithms no matter the number of reductions. For example, suppose we have 900 as a maximum number of evaluations and COR-NSGA-II performs 3 reductions, each one runs the optimization algorithm for 300 evaluations.

Thus, 30 independent runs were performed using the combination of the parameters. After tuning, the best parameter settings were selected based on the best average values of R-HV and R-IGD. At the end, the values chosen are displayed in Table 6.4.

Table 6.4: Parameter Settings.

Parameter	Algorithm	Value
Population Size	All	112
Max Evaluations	All	134,400
Crossover Operator	All	Single Point Crossover
Crossover Probability	All	0.9
Mutation Operator	All	Bit Flip Crossover
Mutation Probability	All	0.005
$\epsilon$	R-NSGA-II	0.001
# of Items for evaluation	COR-NSGA-II	5
# of Reductions	COR-NSGA-II	10

With the best configuration of parameters chosen, 30 independent runs of each algorithm were performed for answering **RQ1** and **RQ2**. At the end, the set of non-repeated and non-dominated solutions was obtained.

As a statistical test, Kruskal-Wallis [50] with 95% significance level was considered where the bold values in the tables represent the best ones, and light gray cells represent values that are statistically equivalent.

Finally, the algorithms were executed in a machine with an Intel(R) Core(TM) i7-5930K CPU 3.50GHz with 40Gb RAM.

## 6.7 RESULTS

This section summarizes and discusses the results obtained in the experimentation. Subsection 6.7.1 presents results regarding **RQ1**, where COR-NSGA-II is compared to an algorithm with a random objective reduction method. Subsection 6.7.2 presents the results regarding **RQ2**, where the solutions found by COR-NSGA-II were compared to those ones found by MOEAs and MaOEAs. Subsection 6.7.3 shows the results for answering **RQ3** about the selected solutions by the potential users, and finally, Subsection 6.7.4 introduces an evaluation about Nautilus, the tool used during the experiments (**RQ4**).

### 6.7.1 RQ1 - Sanity Check

This RQ seeks to compare the results found by COR-NSGA-II to those ones found by a random dimensionality reduction algorithm (or simply, random algorithm). The results found by COR-NSGA-II with 80% and 100% of minimum confidence level are presented, and concerning the random algorithm, the results are shown with 5 and 10 reductions. To ease understanding, we present the results of both experiments (Scenarios 2D e 3D) in separated sections.

#### 6.7.1.1 Scenario 2D

Table 6.5 shows the mean values and standard deviations for Reduction Efficiency, Size of the Last Subset, # of Targets in the Last Subset, and the Reduction Capacity for COR-NSGA-II and random algorithm for Scenario 2D (*Number of Product* and *Alive Mutants* objectives as preferred ones).

Table 6.5: COR-NSGA-II versus Random Algorithm in Scenario 2D.

	Algorithm	Reduction Efficiency	Size of the Last Subset	# of Targets in the Last Subset	Reduction Capacity
James	random-10	0.00 ± 0.00	1.00 ± 0.00	0.27 ± 0.45	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.03 ± 0.18	0.40 ± 0.50	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.70 ± 0.47</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	2.03 ± 0.32	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
CAS	random-10	0.00 ± 0.00	1.00 ± 0.00	0.27 ± 0.45	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.33 ± 0.48	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.53 ± 0.51</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	1.63 ± 0.49	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
WS	random-10	0.00 ± 0.0	1.00 ± 0.00	0.27 ± 0.45	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.07 ± 0.25	0.27 ± 0.45	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
E-Shop	random-10	0.00 ± 0.00	1.00 ± 0.00	0.13 ± 0.35	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.03 ± 0.18	0.40 ± 0.50	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
Drupal	random-10	0.00 ± 0.00	1.00 ± 0.00	0.40 ± 0.50	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.20 ± 0.41	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
Smarthome	random-10	0.00 ± 0.00	1.00 ± 0.00	0.27 ± 0.45	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.47 ± 0.51	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>2.00 ± 0.00</b>	<b>100.00% ± 0.00</b>

The table shows COR-NSGA-II reaches the best performance in all instances in which it can converge to the target objectives in 100% of the executions. On the contrary, the random algorithm always reaches just one objective in the last execution and, in almost cases, this one is

not a preferred objective. So, we can conclude that the random algorithm was the worst for this scenario.

Regarding the minimum confidence level used by COR-NSGA-II, the performance was similar (statistically equivalent) for most instances, except for James and CAS. In these ones, the COR-NSGA-II algorithm with 80% of minimum confidence level reached the best results, converging to the preferred objective in less reductions.

Summarizing the results, we can observe that the sanity check has passed (i.e., COR-NSGA-II outperforms the random algorithm by a large degree). Besides, we can assume that the confidence level of 80% is better for this scenario once it slightly scores a better performance compared to 100%, by converging towards the preferred objectives quickly.

### 6.7.1.2 Scenario 3D

Table 6.6 shows the mean values and standard deviations for Reduction Efficiency, Size of the Last Subset, # of Targets in the Last Subset, and the Reduction Capacity for COR-NSGA-II and random algorithm for Scenario 3D (*Alive Mutants*, *Similarity*, and *Cost* objectives as preferred ones).

Table 6.6: COR-NSGA-II versus Random Algorithm in Scenario 3D.

	Algorithm	Reduction Efficiency	Size of the Last Subset	# of Targets in the Last Subset	Reduction Capacity
James	random-10	0.00 ± 0.00	1.00 ± 0.00	0.53 ± 0.51	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.40 ± 0.50	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.83 ± 0.53</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	2.07 ± 0.25	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
CAS	random-10	0.00 ± 0.00	1.00 ± 0.00	0.43 ± 0.50	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.47 ± 0.51	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.20 ± 0.41</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	1.33 ± 0.48	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
WS	random-10	0.00 ± 0.00	1.00 ± 0.00	0.43 ± 0.50	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.40 ± 0.50	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
E-Shop	random-10	0.00 ± 0.00	1.00 ± 0.00	0.37 ± 0.49	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.43 ± 0.50	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
Drupal	random-10	0.00 ± 0.00	1.00 ± 0.00	0.37 ± 0.49	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.00 ± 0.00	0.57 ± 0.50	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
Smarthome	random-10	0.00 ± 0.00	1.00 ± 0.00	0.33 ± 0.48	0.00% ± 0.00
	random-5	0.00 ± 0.00	1.07 ± 0.25	0.50 ± 0.57	0.00% ± 0.00
	cor-nsga-ii-0.8	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>
	cor-nsga-ii-1.0	<b>1.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>3.00 ± 0.00</b>	<b>100.00% ± 0.00</b>

The table shows COR-NSGA-II also reaches the best performance in all instances in which it can converge to the preferred objectives in 100% of the executions. Regarding to the random algorithm, a similar performance found in Scenario 2D was also found in this scenario. In this one, the random algorithm presented the worst performance, by reducing the set to just one objective and this, in most cases, was not the preferred objective.

Concerning the minimum confidence level used by COR-NSGA-II in this scenario, the performance was also similar (statistically equivalent) for most instances, except again for James and CAS. In these instances, COR-NSGA-II with 80% of confidence level converged to the preferred objective in less reductions.

Summarizing the results found in this scenario, we can observe that the sanity check has also passed. Besides, we can also assume (such Scenario 2D) that the minimum confidence level of 80% is better for the addressed problem once it slightly scores a better performance compared to 100% (by converging to preferred objectives quickly, in general when compared to the other). So, answering **RQ1**, we observed that the COR-NSGA-II algorithm is capable of reducing the problem dimensionality towards the user preferences by generating in the last subset of objectives, those user-preferred ones in all evaluated scenarios.

#### 6.7.2 RQ2 - Comparing COR-NSGA-II to Multi- and Many-objective Evolutionary Algorithms

To answer this RQ the results found by COR-NSGA-II in both scenarios are compared to those ones found by MOEAs and MaOEAs. As described in Section 6.5, two types of RPs (Restricted and Compromised ones) were used for each scenario for calculating the R-Metrics (R-HV and R-IGD), and running R-NSGA-II. All results can be found in Appendix G.

##### 6.7.2.1 COR-NSGA-II Versus NSGA-II and NSGA-III

In this subsection, the results found by COR-NSGA-II are compared to NSGA-II and NSGA-III. Aiming to facilitate a better understanding of the results, we present these ones of both experiments (Scenarios 2D e 3D) in separated sub-sections.

##### 6.7.2.2 Scenario 2D

Table G.1 shows the mean values and standard deviations for R-HV, R-IGD, # of solutions, # of Solutions in the ROI and Execution Time for Scenario 2D. The table shows that, regarding Restrict RP, COR-NSGA-II reaches the best R-HV and R-IGD values for all instances with a statistical difference to the other algorithms. The same performance can be found for # of Solutions, # of solutions in the ROI and Execution Time.

On the one hand, when the Compromised RP is considered, COR-NSGA-II outperforms them just for the WS instance concerning to R-HV. For R-IGD values, this one reaches statistical equivalence for Smarthome and Drupal, the largest instances. On the other hand, COR-NSGA-II can also reach the best values concerning # of Solutions, # of solutions in the ROI and Execution Time for all instances. For the other quality attributes, it is possible to notice that NSGA-II reached the best performance.

To sum up, considering all RPs and instances, for R-HV and R-IGD, the COR-NSGA-II algorithm is the best in 7 (out of 12) instances, NSGA-II in 4 and NSGA-II in one single instance. Considering other quality attributes, COR-NSGA-II outperformed the other algorithms in all instances.

### 6.7.2.3 Scenario 3D

Regarding Scenario 3D, Table G.2 shows the mean values and standard deviations for R-HV, R-IGD, # of solutions, # of Solutions in the ROI and Execution Time for COR-NSGA-II, NSGA-II and NSGA-III.

For R-HV and R-IGD, the table shows that COR-NSGA-II can reach the best values for James, CAS, WS and E-Shop instances considering Restricted and Compromised RPs. Concerning # of Solution and # of solutions in the ROI, COR-NSGA-II performs better for the largest instances (such as E-Shop, Drupal and Smarthome) by generating less solutions inside the ROI. Besides, the results for the Execution Time show again the COR-NSGA-II algorithm can reach the lowest time for all instances and RPs.

To sum up, the results found in this RQ, considering again all RPs and instances, for R-HV and R-IGD, the COR-NSGA-II algorithm is the best in 8 (out of 12) instances, NSGA-II in 4 instances.

Summarizing the results and taking into account all instances, Table 6.7 shows the number of times that the algorithms generated the best results for each quality indicator.

Table 6.7: COR-NSGA-II Versus NSGA-II and NSGA-III in both scenarios.

Scenario	Algorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time
2D	COR-NSGA-II	<b>7</b>	<b>9</b>	<b>12</b>	<b>12</b>	<b>12</b>
	NSGA-II	4	3	0	0	0
	NSGA-III	1	4	0	0	0
3D	COR-NSGA-II	<b>8</b>	<b>7</b>	<b>6</b>	<b>10</b>	<b>12</b>
	NSGA-II	4	5	0	2	0
	NSGA-III	0	0	<b>6</b>	0	0

It is possible to notice that COR-NSGA-II can generate the best results for all scenarios and most quality indicators compared to the NSGA-II and NSGA-III algorithms. Just for Scenario 3D and the largest instances, the performance of COR-NSGA-II is slightly worst, but it takes a reduced time to execute and generates a lower number of solutions in the ROI.

### 6.7.2.4 COR-NSGA-II Versus R-NSGA-II

In this subsection, the results found by COR-NSGA-II are compared to R-NSGA-II, a preference-based algorithm. Aiming to facilitate a better understanding of the results, we present these ones of both experiments (Scenarios 2D e 3D) in separated sub-sections.

### 6.7.2.5 Scenario 2D

Table G.3 shows the mean values and standard deviations for R-HV, R-IGD, # of solutions, # of Solutions in the ROI and Execution Time for Scenario 2D.

The table shows that, regarding Restrict RP, COR-NSGA-II reaches the best R-HV and R-IGD values for all instances with a statistical difference to the other algorithms, and considering Compromised RP, the best ones for the CAS, WS, Drupal instances. In addition to this, COR-NSGA-II outperforms R-NSGA-II for # of Solutions, # of Solution in the ROI and Execution Time. Focusing in R-NSGA-II, it can reach the best values for James, E-Shop, and Smarthome instances when the Compromised RP is taken into account.

So, to sum up, considering all RPs and instances, for R-HV and R-IGD, the COR-NSGA-II algorithm is the best in 8 (out of 12) instances, R-NSGA-II in 4. For the other quality attributes, COR-NSGA-II reached the best results.

#### 6.7.2.6 Scenario 3D

Regarding Scenario 3D, Table G.4 shows the mean values and standard deviations for R-HV, R-IGD, # of solutions, # of Solutions in the ROI and Execution Time for COR-NSGA-II and R-NSGA-II.

In this scenario, COR-NSGA-II generates the best results for James, CAS, WS, and E-Shop instances for all RPs and quality attributes. Considering Drupal and Smarthome, R-NSGA-II outperforms the COR-NSGA-II algorithm regarding R-HV and R-IGD. However, when the # of solutions, # of solutions in the ROI and Execution Time are taken into account, COR-NSGA-II maintains generating the best values.

To sum up these results, the performance was similar to that one found in Scenario 2D where COR-NSGA-II is the best in 8 instances (out of 12) and R-NSGA-II in 4 instances.

Again, Table 6.8 shows the number of times that the algorithms generated the best results for each quality indicator, considering all RPs.

Table 6.8: COR-NSGA-II Versus R-NSGA-II in both scenarios.

Scenario	Algorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time
2D	COR-NSGA-II	<b>8</b>	<b>10</b>	<b>12</b>	<b>12</b>	<b>12</b>
	R-NSGA-II	4	4	0	0	0
3D	COR-NSGA-II	<b>8</b>	<b>8</b>	<b>12</b>	<b>12</b>	<b>12</b>
	R-NSGA-II	4	4	0	0	0

The table presents that COR-NSGA-II can outperform the R-NSGA-II algorithm in all scenarios. Specifically for Scenario 3D and the largest instances, the results found by COR-NSGA-II is slightly worst (similar to those ones found when it compared to NSGA-II and NSGA-III), but it maintains a less Execution Time, and generates a lower number of solutions in the ROI.

#### 6.7.2.7 COR-NSGA-II Versus PCA-NSGA-II

In this subsection, the results found by COR-NSGA-II are compared to PCA-NSGA-II, a dimensionality reduction algorithm. Scenario 2D and 3D are addressed in separated sub-sections aiming to facilitate a better understanding of the results.

#### 6.7.2.8 Scenario 2D

Table G.5 shows the mean values and standard deviations for R-HV, R-IGD, # of solutions, # of Solutions in the ROI and Execution Time for Scenario 2D.

Regarding Restrict RP, the table shows that COR-NSGA-II is the best concerning R-HV and R-IGD values for all instances with statistical difference to PCA-NSGA-II. The same performance can be found for # of Solutions, # of solutions in the ROI and Execution Time for almost instances and RP. Just for James, PCA-NSGA-II generates the best # of Solution in the ROI.

For this Scenario, PCA-NSGA-II just generates the best results for E-Shop, Drupal and Smarthome when the Compromised RP is taken into account. When R-IGD is considered, the performance of the last ones are statistically equivalent to COR-NSGA-II.

To sum up the found results in this scenario, considering all RPs and instances, for R-HV and R-IGD, the COR-NSGA-II algorithm is the best in 9 (out of 12) instances, PCA-NSGA-II in 3 instances. In all instances, in general, COR-NSGA-II generates less solutions in a lower Execution Time.

#### 6.7.2.9 Scenario 3D

Regarding Scenario 3D, Table G.6 shows the mean values and standard deviations for R-HV, R-IGD, # of solutions, # of Solutions in the ROI and Execution Time for COR-NSGA-II and PCA-NSGA-II.

Analyzing this scenario, COR-NSGA-II generates the best value for R-HV and R-IGD for James, CAS, WS, and E-Shop instances considering all RPs. For # of Solutions and # of Solutions in the ROI, and Execution time, the proposed algorithm outperforms the PCA-NSGA-II algorithm for CAS, WS, E-Shop, Drupal and Smarthome instances. However, for James, PCA-NSGA-II generates better values for # of Solutions and # of Solution in the ROI.

To sum up these results, again, the performance was similar to that one found in Scenario 2D where COR-NSGA-II is the best in 8 instances (out of 12) and PCA-NSGA-II in 4.

Summarizing the found results, Table 6.9 the number of times that the algorithms generated the best results for each quality indicator, considering all RPs.

Table 6.9: COR-NSGA-II Versus COR-NSGA-II in both scenarios.

Scenario	Algorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time
2D	COR-NSGA-II	<b>9</b>	<b>11</b>	<b>12</b>	<b>11</b>	<b>12</b>
	PCA-NSGA-II	3	3	0	1	0
3D	COR-NSGA-II	<b>8</b>	<b>8</b>	<b>10</b>	<b>10</b>	<b>12</b>
	PCA-NSGA-II	4	4	2	2	0

COR-NSGA-II reaches the best values for all scenarios when compared to PCA-NSGA-II. Considering the Execution Time, it is important to notice that the proposed algorithm reaches the lowest execution time, where in some cases, this one can be less than half of the time spent by other algorithms.

Answering **RQ2**, if we consider other quality attributes such as the # of solutions, COR-NSGA-II reaches, in general, less solutions in the ROI and these ones have a good performance (taking into account the R-HV and R-IGD) when compared to those ones found by the other algorithms. On the one hand, COR-NSGA-II outperforms the other algorithms in most instances. On the other hand, its performance is slightly decreased when the Compromised RP and 3D Scenario (not necessarily together) are considered. This is a subject discussed in Section 6.8.

#### 6.7.3 RQ3 - Evaluating the Solutions

In this RQ, the goal is to evaluate the usefulness of the solutions according to the user preferences found by COR-NSGA-II compared to those ones found by multi- and many-objective evolutionary algorithms. So, for answering this RQ, the users were required to run Nautilus and picked a

solution up from the population generated by COR-NSGA-II, R-NSGA-II, and NSGA-II, the best algorithms found in **RQ2**. Also, they were required to, manually, provide a solution (Nautilus also supports this process).

As explained in Subsection 6.3.2, before performing the experiment, the users were required to select at most 4 objectives (from 7 available to be optimized) as preferred ones. This was required seeking to guarantee COR-NSGA-II will perform some reduction. Details about the set of preferred objectives selected by the participants and the final set of objectives generated by COR-NSGA-II are shown in Appendix H. Thus, Figure 6.4 shows the number of participants by objectives.

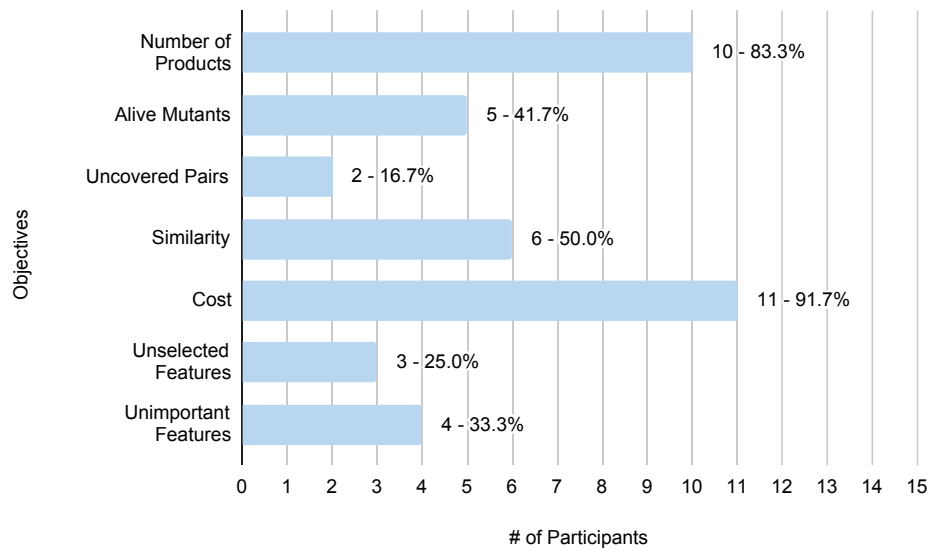


Figure 6.4: Preferred Objectives from the user’s point of view.

The figure shows that, on the one hand, 11 users (91% of the them) selected *Cost* as the preferred objective, followed by *Number of Products* selected by 10 users (or 83.3%). On the other hand, *Unselected Features* and *Uncovered Pairs* were less preferred by the users.

Table 6.10 presents detailed information for each participant about the # of Preferred Objectives, # of Reductions performed by COR-NSGA-II, Size and # of Targets in the Last Subset of objectives.

Table 6.10 shows that 3 users selected 2 objectives, 1 user selected 3 objective as preferred, and 8 ones selected 4 objectives. The participants #2, #3, #4, #5, and #12 selected a solution when in the last execution, the set of objectives optimized contained just the preferred ones (Size and # of Targets in the Last Subset are the same). However, the participants #1, #8, #9, #10, and #11 were able to pick a solution up before COR-NSGA-II reaches this convergence.

After interacting with Nautilus and selecting a good solution for each algorithm, we asked the users to describe how difficult was the selection of this solution. Figure 6.5 shows the feedback captured from the questionnaire.

This figure describes that 8 users chosen the option “Easy” and “Very Easy” for COR-NSGA-II, 4 for the NSGA-II and R-NSGA-II algorithms. In the last position, the manual selection appears as the most difficult (8 users). Analyzing the motivation, some participants claimed COR-NSGA-II generated less solutions and other ones claimed COR-NSGA-II took less execution time compared to the other algorithms. With this, we asked the users to rank the algorithm based on his/her preferences where 1 means the best one and so on. The information are displayed in Figure 6.6.



Table 6.10: COR-NSGA-II's results for each participant.

Participant	# of Preferred Objectives	# of Reductions	Size of the Last Subset	# of Targets in the Last Subset
#1	2	3	3	2
#2	2	2	2	2
#3	3	2	3	3
#4	2	4	2	2
#5	4	1	4	4
#6	4	1	3	3
#7	4	1	3	3
#8	4	1	5	4
#9	4	1	5	4
#10	4	1	6	4
#11	4	2	5	4
#12	4	2	4	4

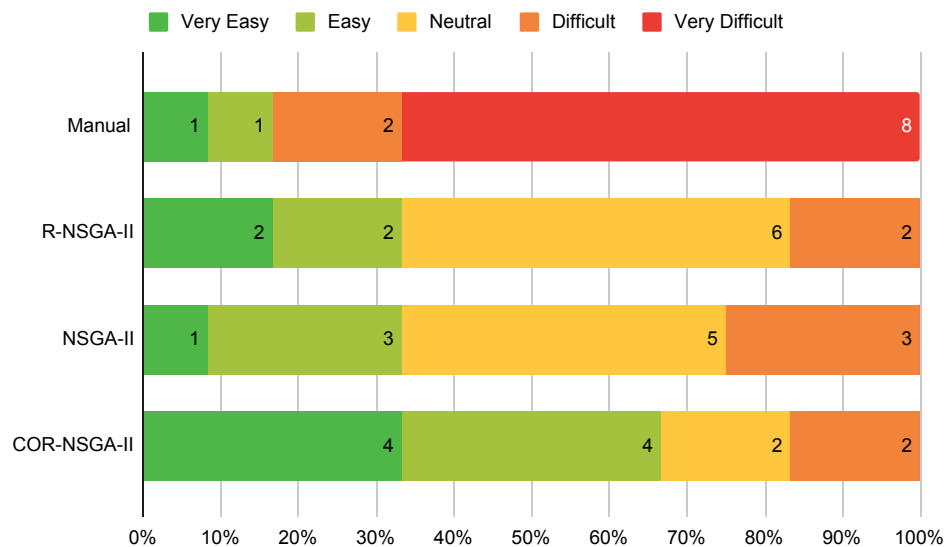


Figure 6.5: Easiest algorithms from the user's point of view.

The figure shows that COR-NSGA-II was ranked as the best algorithm by 10 users (or around 83%). NSGA-II was ranked as the best 2 times. As expected, the manual selection received the lowest score in this experiment, being ranked in the last position by 9 users (around 80%). However, aiming to verify the dependence on the execution order, Table 6.11 shows the best algorithm by group.

The users belong to Group 4 selected the NSGA-II algorithm as the best one. In such a group, COR-NSGA-II was the first algorithm evaluated. We suppose that the results found by COR-NSGA-II influenced the user decision about the solutions generated by the other algorithms, that is, they used the solution selected by applying COR-NSGA-II to guide his/her preferences in

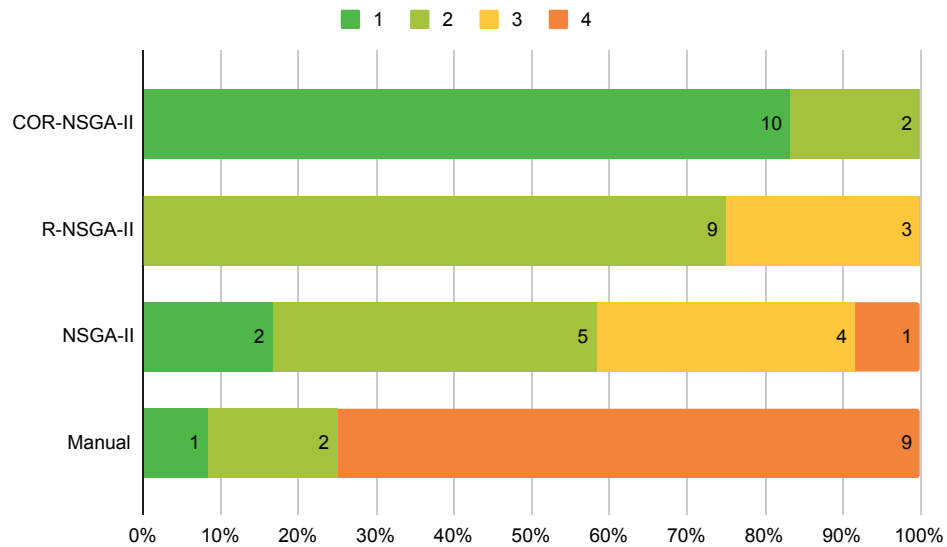


Figure 6.6: Best algorithms from the user's point of view.

Table 6.11: Best algorithm by group.

Group 1	Group 2	Group 3	Group 4
COR-NSGA-II	COR-NSGA-II	COR-NSGA-II	COR-NSGA-II
			NSGA-II

the following algorithms and this can be a motivation for the other algorithms appear as good as COR-NSGA-II. However, more experiments must be performed to validate our hypothesis.

A fact that corroborates this hypothesis is that we asked the users if the order of the execution in each group (described in Table 6.2) impacted their responses. Around 58.3% answered positively.

Summarizing the results and answering **RQ3**, 10 out of 12 users defined that the solutions found by COR-NSGA-II were the best ones compared to other algorithms and it was also easy to select them. So, and concluding the research questions, COR-NSGA-II can help the user to find useful solutions for the addressed problem.

#### 6.7.4 RQ4 - Evaluating Nautilus

Nautilus tool was evaluated and the answers provided by the users are shown in the following. The answers from the questionnaire were grouped seeking to facilitate the understanding of them.

Figure 6.7 shows how much time the users spent to get familiar and explore the Pareto-front. Regarding the time spent to get familiar, 8 participants took less than 10 minutes in which, 5 of them spent less than 5 minutes. Besides, all users claimed to spend less than 10 minutes to explore the Pareto-front by using the visualization support.

We also asked to the users how difficult was to use Nautilus. The answers about these questions are shown in Figure 6.8.

The figure presents that 58.3% of the users said it was easy to learn to operate the tool and just one claimed difficulty. Besides, around 75% of the users stated that it is easy to understand the task they were asked to do.

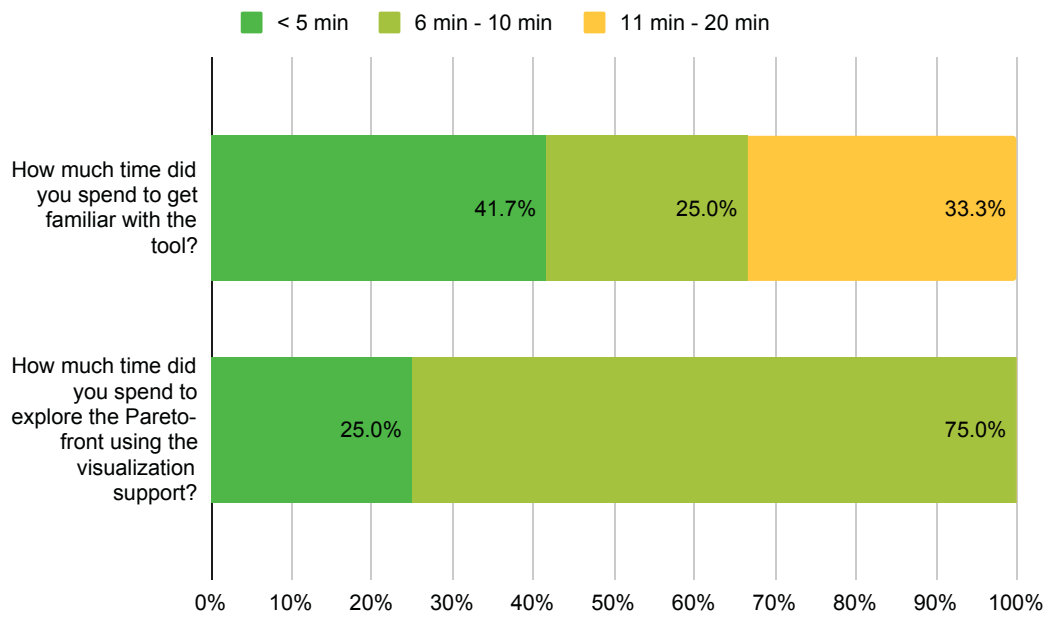


Figure 6.7: *How much time?* answers from questionnaire.

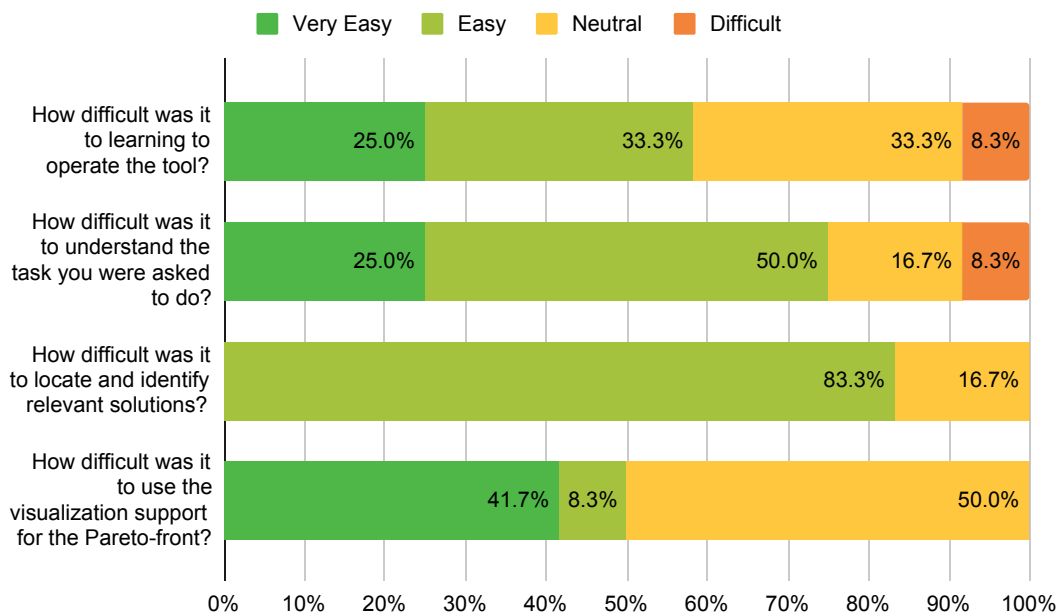


Figure 6.8: *How difficult time?* answers from questionnaire.

Still in this context, 83.3% of the asked users asserted that it was easy to locate and identify relevant solutions. Besides, 50% of the users stated it is easy to use the visualization support for the Pareto-front. Other 50% claimed it was neutral.

During the user interaction with Nautilus, we asked the users to provide a feedback if s(he) agree with the statements in Figure 6.9.

The figure presents that most users (more than 60% of them) asserted that the tool has a user-friendly interface, it is very easy to navigate and the error messages are helpful. We also asked to the users their opinions about the organization of the information in the screen. The

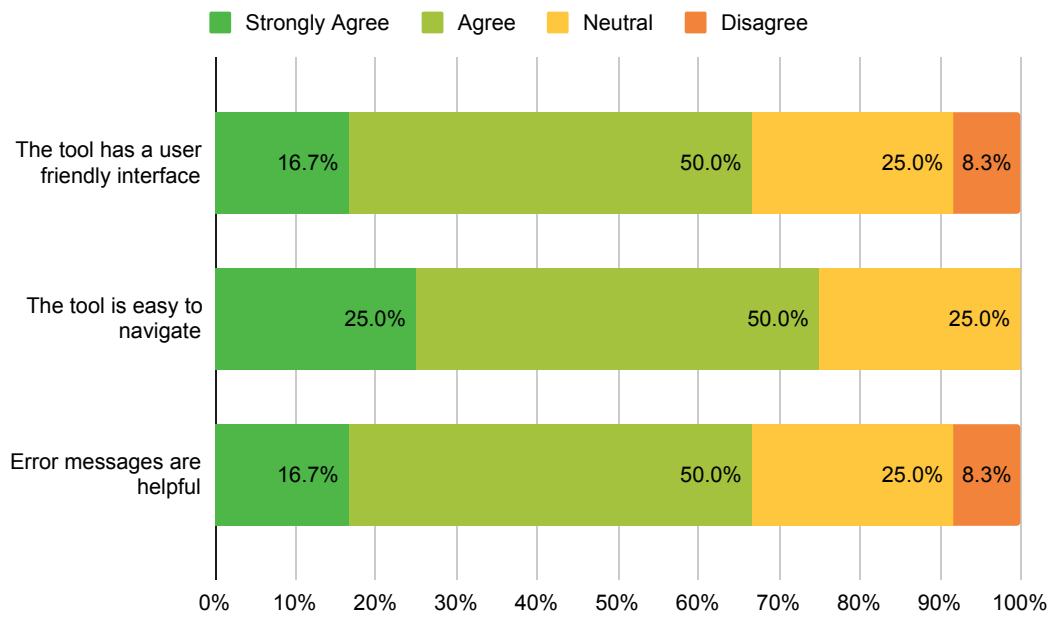


Figure 6.9: *Agree or not?* answers from questionnaire.

result is show in Figure 6.10. In this case, more than 70% of them stated that the information in the screen are very clear and just 25% said to be neutral.

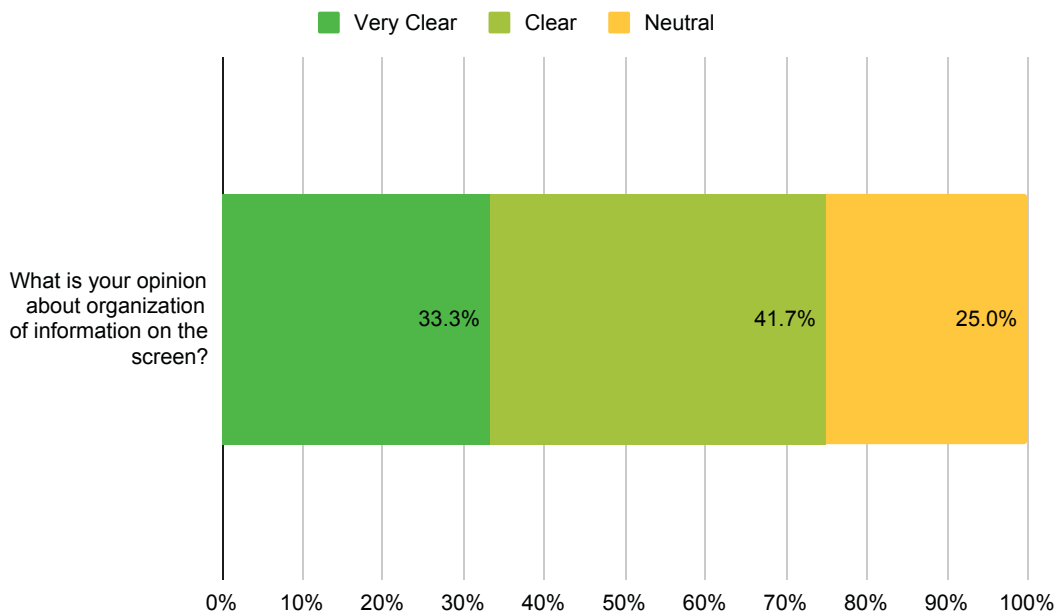


Figure 6.10: *Clear or not?* answers from questionnaire.

Finally, we required to the user to provide in their opinion the best features provided by Nautilus. The user could state 1 for the best feature, 2 for the second best feature and so one. Figure 6.11 shows the answer for this question.

The figure shows that for most users, the best feature Nautilus provides is the Pareto-front visualization followed by the interface. In third place, we have the cloud-computing support as

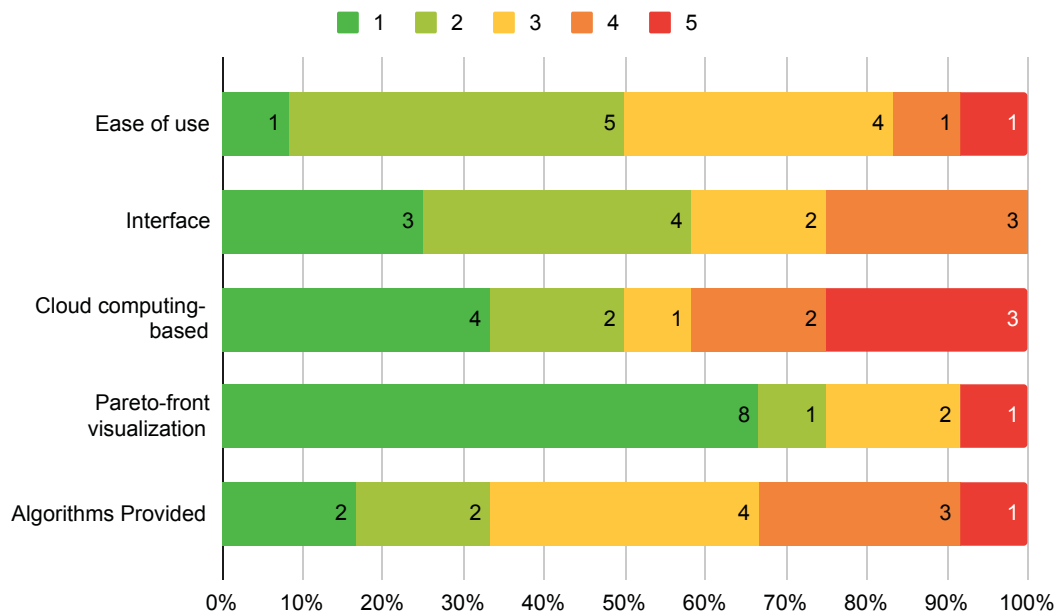


Figure 6.11: Nautilus' best features.

the best feature and ease of use in the next feature. As the last best feature, we have the algorithms provided.

So, to summarizing the answers provided by the users and answering **RQ4**, we can suppose based on the feedback collected by the questionnaire, that Nautilus is useful for the task of selecting a good solution since, for the majority of the users, it is easy to learning, to navigate, and locate relevant solutions. It has a user-friendly interface with well-organized information on the screen, and as the best feature, the users stated the Pareto-front visualization support.

## 6.8 DISCUSSION

As presented in the previous section, COR-NSGA-II can generate less solutions, taking less execution time for searching them, but maintaining the quality attributes in a competitive way. However, in this section, we discuss some important findings of the results of our experimental study.

As expected in **RQ1**, the random dimensionality reduction algorithm was the worst one for the addressed problem once it does not take into account the user preferences. In all cases, this one ends the search process simply because this reached the minimum number of objectives (in case, a single one). On the contrary, COR-NSGA-II reached the preferred objectives, on average, in the second reduction. As in this experiment a user simulator was used and this one is responsible for selecting randomly solutions in the Pareto-front, the number of reduction could be less if a kind of smart mechanism was used in this task.

In **RQ2**, in most cases COR-NSGA-II found better results for the addressed problems. However, it is possible to notice a slightly decreasing of the values regarding R-HV and R-IGD when the number of preferred objectives increases. We suppose that the presence of redundant objectives in this set can affect the algorithm performance. In addition to this, we also suppose that if the set of preferred objectives has almost the same size as the original set of objectives, the performance must not be competitive since COR-NSGA-II may not perform multiple reductions.

New experiments should be performed in the future to verify this assumption with different scenarios.

Although this can happen, the difference between COR-NSGA-II and the other algorithms remains small and it is a good trade-off. For example, considering R-NSGA-II in Scenario 3D, the performance of COR-NSGA-II was around 3% worse than the former for the Smarthome instance (the largest one). However, for the same context, R-NSGA-II took 8 hours on average for finding a solution, while COR-NSGA-II with its dimensionality reduction mechanism took 3.2 hours on average, that is, a reduction in the execution time more than 50%.

This is another important finding in the experiment performed in this work. COR-NSGA-II took less time in all instances and compared with algorithms, and the difference among the algorithms increases when the instance size increases as well. This becomes COR-NSGA-II an important algorithm in the context of where the number of products to be selected is huge.

Still in this research question, the performance of COR-NSGA-II remains good when the Compromised RP is considered. Different from Restricted RP, the former considers 0.5 in the RP for the non-preferred objectives (that is, although a small value, it still is a user preference for this objective). However, COR-NSGA-II does not take into account this concept, that is, for the algorithm, if an objective to be optimized is not preferred, it is simply discarded. So, more researches in the future should evaluate the performance of the proposed algorithm in the context where there are some small preferences for non-preferred objectives. Moreover, the use of COR-NSGA-II is recommended in the context where all objectives should be optimized but some of them are preferred. If, in the user's point of view, all objectives have the same preferences, traditional MOEAs and MaOEAs should be applied.

Concerning to **RQ3**, two unexpected situations were identified during the performance of the experiments. The first one is the fact that a user selected as preferred objectives two ones that are redundant. Because of this, the manual mechanism for selecting a solution was considered by this user better than the optimization algorithms used in this experiment (although the user took more than 20 minutes to pick a solution up in using the manual mechanism).

Another finding was that some users took more than 10 minutes exploring the Pareto-front, providing his/her preferences. Since the experiment did not set a limit to the users express his/her preferences in COR-NSGA-II, some users spent time providing his/her preferences as much as possible. Thus, seeking to improve this process, it is necessary in a future work, to study a kind of limit for the number of provided user preferences and, also, to propose a way to deal with redundant objectives in the set of preferred ones.

Finally, regarding **RQ4** where Nautilus was evaluated by the user, several further comments were collected. For example, some for them suggest that, although the tool provides a good Pareto-front visualization for the population, an automatic mechanism for suggesting solutions to be evaluated to the users should be implemented. Still in this context, some users suggested for the future version of the tool, the use of mechanisms for quickly identifying solutions that already have some preferences provided by them, such as, use different colors in the Pareto-front visualization.

## 6.9 THREATS TO VALIDITY

The threats to validity are divided into four categories according to the framework proposed by Wohlin et al. [86].

### 6.9.1 Internal Validity

With regard to the internal validity, we use the tool FMFS to calculate the objective values of our problem. FMFS makes use of the FaMa framework to deal with resource model constraints and derive products. However, FaMa has some limitations to work with large FMs such as Drupal and Smarthome. To mitigate this threat, the user can set a number  $n$  of products to be used for selection. Other representations for the problem can be used in the future, as well as other analyzers, such as SAT solver.

Regarding reference points used in **RQ2**, the results are dependent on the reference points used. These ones were selected considering our previous knowledge about the Pareto-front and the set of preferred objectives. So, aiming to mitigate this threat, two kinds of RPs were considered in which the non-preferred objectives in the first RP do not have any preferences, while for the second one compromised values (0.5 to be exact) are provided to non-preferred objectives. To better evaluate the impact of the provided RPs, we need to conduct future experiments.

The different scenarios used can also be considered a threat. In order to mitigate this threat, two scenarios were considered trying to represent a real scenario, in which the first scenario has just conflicting objectives, and the second one includes redundant objectives.

Furthermore, the experiments with users showed that the algorithm is highly dependent on the context in which it is applied and the set of preferred objectives defined by the user, especially because the participants do not work in the system on which the FMs were based. Hence, all users received the same information and, before performing the tests, all of them took part in a preliminary test to learn how to use the tool, and prevent any effects of ignorance on their usage.

Another threat to the internal validity can be verified: the *Hawthorne effect* [68]. This effect defines that the user's behavior can change according to specific situations or a special treatment. Thus, the users can change their behavior and performance knowing that they are participating in an experiment.

Finally, seeking to mitigate these problems, the experiment was conducted as consistently as possible. We attempted not to conduct the experiment with all of the participants at the same time and gave each one the option of choosing the best time to participate. Besides, the users were separated in four groups in which each one has a different sequence of execution. In this manner, we mitigate the fact of solutions found by the last executed algorithm influence his/her opinion about the new found solutions.

### 6.9.2 Construct Validity

The users were involved only in this study, which did not generate a difference between the interaction treatments. Besides, an explanation were given at the beginning of the experiments in which they were not informed about what exactly would be investigated, so the participants could not imagine that their personalities were involved in the study. Hence, this approach helped to prevent the participants from guessing the hypothesis.

Besides, we used questionnaires to assess the comprehension of the solution selection process and the participants' answers to these questionnaires were evaluated comparing the answers with the quality metrics for them. This design choice avoided as much as possible any subjective evaluation. Thus, the questionnaires were defined to be complex enough without being too obvious.

### 6.9.3 External Validity

We tested COR-NSGA-II and Nautilus in six different instances of SPL Testing. Even though these instances are evaluated in other studies of the literature, we cannot state that this is enough to generalize the results. Besides, the cost and importance (both of them were randomly defined), and the size of the instances may not reflect real-world FMs. To minimize this threat, we tried to evaluate FMs of several sizes (including two with more the 11k products to be selected) and domains.

Regarding the larger instances, we consider that 11k of products to be selected were adequate proportionally to the mutation score (around 97%) chosen for the experiments performed in this work. However, a greater number for large SPLs should be evaluated in a future experiment. It is expected similar performance, and good results with acceptable time.

Moreover, the results of the experiments performed to answer **RQ3** and **RQ4** depend on the experience of each participant, especially for the optimization algorithm. A larger number of participants would proportionate more strength to the work, but the total years of experience of the participants in this field (29 years) suggest a certain level of credibility to their participation. Also, the experiment performed with some students can be considered a threat in this study. However, the tasks considered in our experiments do not require a high level of industrial experience, but some replications of these experiments with professionals are needed in the future to confirm and contradict the achieved results.

### 6.9.4 Conclusion Validity

Regarding conclusion validity, it is not possible to compare the results with similar approaches given that there is an absence of similar studies using preference-based dimensionality reduction in the literature.

The parameter settings used by the algorithms can be a threat to the experiments. The number of evaluations is the same for every one, even COR-NSGA-II that applies dimensionality reduction. Different values, especially in the larger instances, could result in different, perhaps better, capacity of reducing towards the user preferences.

To address the stochastic nature of the evolutionary algorithms, all the algorithm were performed 30 times for each instance and reference points to answer **RQ1** and **RQ2**, while capturing the arithmetic mean and standard deviation of the metrics.

The parameter settings, RPs,  $\delta$  value used in R-Metric, and quality indicators can also be stated as a threat once different indicators could derive a divergent conclusion. So, to mitigate this, the algorithms were tuned with different parameters aiming to select the best ones, and the quality indicators used are widely used in the literature. Also, Kruskal-Wallis test with 95% significance level was used to compare the results found by the algorithms. This test is quite robust and it has been extensively used in the past to conduct analyses similar to ours.

## 6.10 FINAL REMARKS

In this chapter, we presented the experiments conducted to evaluate COR-NSGA-II and Nautilus. Four RQs were formulated and the experiments encompassed six FMs from different domains, two types of reference points, and we used four quality indicators for asserting **RQ1** and five ones for evaluating **RQ2**. The solutions generated by COR-NSGA-II were compared to four multi- and many-objectives evolutionary algorithms from literature.

After analyzing the data, we were able to positively answer all research questions proposed in this empirical study. If all objectives in a given optimization problem should be



optimized but some of them are preferred, COR-NSGA-II is the best choice for this context once the algorithm can generate less solutions taking less execution time comparing to the other ones. In most instances, COR-NSGA-II outperforms the other algorithms regarding the quality attributes analyzed.

The main observed advantage of COR-NSGA-II is that it filters the solutions generated taking into consideration the user preferences. Letting COR-NSGA-II generate the solutions, the burden of selecting a solution from the population may be reduced once less (and good) solutions were generated.

In this sense, we can accept the main hypothesis presented in the introduction of this work: “a preference-based dimensionality reduction approach is capable of taking less execution time and generating a reduced set of solutions that takes into account the user preferences. In addition to this, the solutions are as good as those ones generated by multi- and many-objective algorithms with respect to quality indicators from the literature.”.

## 7 CONCLUSION

This work presented MaDRUP, an approach for dimensionality reduction guided by the user preferences provided in an interactive way, that can be instantiated with different algorithms and different kinds of elements such as items to be visualized or information required from the user. The main motivation for proposing and implementing these ones is to reduce the number of generated solutions in many-objective problems, since such a number increases exponentially with the number of objectives, and the use of dimensionality reduction and user preferences can mitigate this problem.

Also, this work also introduces Nautilus, a clouding-computing and Java web-based tool for mono-, multi- and many-objective evolutionary algorithms. The motivation of this tool is to support MaDRUP approach and other MOEAs and MaOEAs found in the literature.

We also introduce an instantiation of MaDRUP with NSGA-II that uses a confidence-level, called COR-NSGA-II. The algorithm reduces the number of objectives to be optimized towards the user needs based on a confidence level for each objective optimized. Based on that, this algorithm is categorized in Stay Out class, that is, this one defines which objective should be removed from the next execution.

As main characteristics, COR-NSGA-II is an algorithm that requires the user preferences interactively (or in-the-loop) and reduces the number of objectives in an online approach, that is, during the solution generation process. The algorithm shows to the user the current set of non-dominated solutions and, in this point, the user can express his/her preferences about them, by using an ordinal scale composed of three options: *Non-preferred*, *No Opinion*, and *Preferred*.

For assessing the feasibility of COR-NSGA-II, multiple experiments were performed. The experiments were conducted using six different FMs, two types of reference points, five algorithms, and two scenarios to answer four research questions.

The first research question evaluates the efficacy of COR-NSGA-II regarding the reduction of the problem dimensionality towards the user preferences. For that purpose, we compared COR-NSGA-II to a random dimensionality objective algorithm and concluded that the proposed algorithm is, in fact, capable of guiding the search process to the objectives preferred by the users.

In the second research question, the results found by COR-NSGA-II were compared to those ones found by MOEAs and MaOEAs, specially algorithms based on reference-set, preferences and dimensionality reduction. By using R-HD, R-IGD, # of Solutions, # of Solution in the ROI, and Execution Time, we observed that COR-NSGA-II, in most instances and scenarios, obtained the best results or results statistically equivalent to the algorithms evaluated, even when the Compromised RP is considered. In this sense, we can conclude that COR-NSGA-II indeed generates a small set of good solutions taking less time to execute and, mainly, incorporating the user preferences.

The third research question addresses the quality of the solutions generated by COR-NSGA-II in the user's point of view. By using Nautilus, a group of users was invited to optimize a problem instance by using four search algorithms (NSGA-II, COR-NSGA-II, R-NSGA-II) and a manual approach and, at the end, to select a solution for each algorithm. The great majority of the users answered that it is easier to pick a solution up generated by COR-NSGA-II, and chose this algorithm as the best comparing with the other ones.

Finally, concerning the fourth research question, the same group of users evaluated the Nautilus tool. A questionnaire was applied seeking to collect qualitative feedback about the tool

and the results show that Nautilus is useful in the task of selecting a good solution, since it is easy to learn how to use it. The tool provides an interface where it is easy to navigate and locate relevant solutions. As the best features, the users stated that Pareto-front visualization and its interface are the best ones.

The main advantage of COR-NSGA-II is that it can generate a small set of solutions to be selected at the final, in less time when the number of objectives increases and, with this, it may reduce the intensive task of selecting a good solution. In this way, the user can focus their expertise on a small set of relevant solutions and not on the optimization in general. Also, even when all objectives to be optimized are conflicting and traditional dimensionality reduction approaches do not perform well, COR-NSGA-II can generate a reduction based on the user preferences to deal with this situation.

Hence, considering the evaluation done in this work and the answers found for the research questions, we can accept the main hypothesis of this work presented in the introduction. COR-NSGA-II is capable of generating a small set of solutions quickly and still keeping the solutions as good as those ones conventionally used in MOEAs and MaOEAs commonly used in the literature.

Finally, the main contributions of this work can be summarized as follows:

1. Generation of an algorithm that can minimize the execution time and the number of solutions for many-objective problems;
2. An approach for dimensionality reduction based on the user feedback provided during the search (or in-the-loop), with information about activities and elements that can be instantiated to derive new algorithms;
3. Development of COR-NSGA-II, a preference-based dimensionality reduction algorithm, that takes into consideration the confidence level to remove an objective from the search process;
4. A set of preference-based quality metrics used for evaluating preference-based dimensionality reduction algorithms;
5. Application of COR-NSGA-II in the task of selecting a solution for the variability testing in SPL, obtaining a reduced set of solutions with less execution time and computation effort;
6. Introduction of Nautilus, a tool used for solving many-objective problems with support to Pareto-front visualization, multiple optimization algorithms and addressed problems;
7. Implementation and availability of COR-NSGA-II and Nautilus as an open-source software;
8. Experimental evaluation and comparison of COR-NSGA-II with multi- and many-objective evolutionary algorithms of the literature.

## 7.1 LIMITATIONS AND FUTURE WORK

This section presents the limitations of the proposed approach. These limitations will be addressed in future work.

The first observed limitation is related to the values of confidence level shown in Subsection 4.2.1 used by COR-NSGA-II for making the decision about which objectives should be

removed in this next execution. The table currently supports three levels of preferences such as *Non-preferred*, *No Opinion*, and *Preferred*. However, in some context, addressed problems or types of users, these levels can not be enough for expressing the user preferences.

Still in this context, the minimum confidence level defined by the user can be a limitation for the approach. If it is used a lower value for this one, the user is required to express his/her preferences for each objective in the set of available objectives. On the contrary, if it is used a greater value, it limits the preferences for the users.

Besides, COR-NSGA-II does not consider the existence of a set of preferred objectives defined by the user composed of just redundant objectives. Such a mechanism to deal with this is also required.

The small set of generated solutions can be used for reducing the burden in the task of selecting a good solution for the problem. However, it was not performed an evaluation comparing the results found by COR-NSGA-II and other interactive approaches (without dimensionality reduction).

Besides, we did not find other approaches or algorithms that use dimensionality reduction based on the user preferences. So there is a limitation, because our approach was not compared with other ones in the same category.

As future work, we intend to address the following subjects.

1. Perform more empirical study to evaluate the scalability of COR-NSGA-II in other FMs used in the industry;
2. Evaluate COR-NSGA-II in an environment in which the number of reductions is limited;
3. Develop new dimensionality reduction algorithms based on the user preferences for removing the non-preferred ones;
4. Study new values for confidence level and increase the feedback options used by COR-NSGA-II;
5. Perform an empirical to evaluate if COR-NSGA-II reduces the burden of selecting a solution with an interactive optimization algorithm. To this end, a solution that uses an interactive algorithm and preference-based ones for the same problem should be proposed and implemented;
6. Evolve Nautilus by supporting techniques for improvement of Pareto-front visualization with colors and more charts;
7. Consider the application of the proposed algorithm in other problems of SBSE research field such as Software Refactoring, Software Requirements and so one;
8. Consider the use of problems with more objectives to be optimized, addressing problems with and without redundant objectives;
9. Perform an empirical study analyzing different scenarios and subsets of preferred objectives;
10. Develop a mechanism to avoid an excessive number of user interactions, by providing relevant solutions as a suggestion to be used by the users;
11. Improve the user simulator by incorporating machine learning mechanism aiming to represent real users and their preferences;

12. Perform a comparative study with several values of  $\delta$ , aiming to verify the effect of the dimensionality reduction when the ROI size increases or decreases;
13. Capture more user preferences, taking into account the explicit and non-explicit ones. The explicit preferences are those ones in which the user, in fact, provides them. In the opposite way, the non-explicit ones are preferences captured during the user interaction but that were not directly provided by the user. For instance, to capture and use the number of times a user visualized a solution even without providing an explicit feedback;
14. Adapt COR-NSGA-II for supporting preferences provided by multiple users.

## REFERENCES

- [1] R. Akbari, R. Hedayatzadeh, K. Ziarati, and B. Hassanizadeh. A multi-objective artificial bee colony algorithm. *Swarm and Evolutionary Computation*, 2:39–52, February 2012.
- [2] A. A. Araújo and M Paixão. Machine learning for user modeling in an interactive genetic algorithm for the next release problem. In *Proceedings of the 6th International Symposium on Search Based Software Engineering (SSBSE '14)*, pages 228–233, Fortaleza, Brazil, 2014. Springer.
- [3] P. Arcaini, A. Gargantini, and P. Vavassori. Generating tests for detecting faults in feature models. In *Proceedings of the 8th IEEE International Conference on Software Testing, Verification and Validation (ICST '15)*, pages 1–10, Graz, Austria, 2015. IEEE.
- [4] M. Asadi, S. Soltani, D. Gasevic, M. Hatala, and E. Bagheri. Toward automated feature model configuration with optimizing non-functional requirements. *Information and Software Technology*, 56(9):1144 – 1165, September 2014.
- [5] S. Bechikh, M. Kessentini, L. B. Said, and K. Ghédira. Preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. In *Advances in Computers*, volume 98, pages 141–207. Elsevier, 2015.
- [6] D. Benavides, S. Trujillo, and P. Trinidad. On the modularization of feature models. In *Proceedings of the 1st European Workshop on Model Transformation (CMT '06)*, page 134, Bilbao, Spain, 2005.
- [7] D. Beuche and M. Dalgarno. Software product line engineering with feature models. *Overload Journal*, 78:5–8, 2007.
- [8] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA - a platform and programming language independent interface for search algorithms. In *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO '03)*, pages 494–508, Faro, Portugal, 2003. Springer.
- [9] J. Branke, K. Deb, K. Miettinen, and R. Słowiński. *Multiobjective Optimization - Interactive and Evolutionary Approaches*, volume 5252. Springer-Verlag Berlin, Heidelberg, 2008.
- [10] E. K. Burke and G. Kendall. *Search methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer, 2005.
- [11] P. Caleb-Solly and J. Smith. Adaptive surface inspection via interactive evolution. *Image and Vision Computing*, 25(7):1058 – 1072, July 2007.
- [12] D. M. Cohen, S. R. Dalal, J. Parelius, and G. C. Patton. The combinatorial design approach to automatic test generation. *IEEE Software*, 13(5):83–88, September 1996.
- [13] M. B. Cohen, M. B. Dwyer, and J. Shi. Coverage and adequacy in software product line testing. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis (ISSTA' 06)*, pages 53–63, Portland, USA, 2006. ACM.

- [14] D. Crockford. Json's web site. <https://www.json.org>. Accessed in 20th August 2019.
- [15] K. Czarnecki, S. Helsen, and U. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software process: Improvement and practice*, 10(1):7–29, March 2005.
- [16] T. J. Dea. Improving the performance of many-objective software refactoring technique using dimensionality reduction. In *Proceedings of the 8th International Symposium on Search Based Software Engineering (SSBSE '16)*, pages 298–303, Raleigh, USA, 2016. Springer.
- [17] T. J. Dea. *Mining, Understanding and Integrating User Preferences in Software Refactoring Using Computational Search, Machine Learning, and Dimensionality Reduction*. PhD thesis, University of Michigan, EUA, 2017.
- [18] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.
- [19] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, Aug 2014.
- [20] 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, August 2002.
- [21] K. Deb and D. Saxena. Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pages 3352–3360, Vancouver, Canada, 2006.
- [22] K. Deb, J. Sundar, U. Bhaskara, and S. Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, 2(3):27–286, July 2006.
- [23] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, October 2011.
- [24] F. Ensan, E. Bagheri, and D. Gašević. Evolutionary search-based test generation for software product line feature models. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAiSE '13)*, pages 613–628, Valencia, Spain, 2012. Springer.
- [25] J. M. Ferreira, S. R. Vergilio, and M. A. Quináia. A mutation approach to feature testing of software product lines. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE'13)*, pages 232–237, Boston, USA, June 2013. Knowledge Systems Institute Graduate School.
- [26] J. M. Ferreira, S. R. Vergilio, and M. A. Quináia. Software product line testing based on feature model mutation. *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, 27(5):817–839, 2017.

- [27] T. N. Ferreira, A. A. Araújo, A. D. Basílio-Neto, and J. T. de Souza. Incorporating user preferences in ant colony optimization for the next release problem. *Applied Soft Computing*, 49:1283–1296, December 2016.
- [28] T. N. Ferreira, J. N. Kuk, A. Pozo, and S. R. Vergilio. Product selection based on upper confidence bound MOEA/D-DRA for testing software product lines. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '16)*, pages 4135–4142, Vancouver, Canada, July 2016. IEEE.
- [29] T. N. Ferreira, J. A. Prado Lima, A. Strickler, J. N. Kuk, S. R. Vergilio, and A. Pozo. Hyper-heuristic based product selection for software product line testing. *IEEE Computational Intelligence Magazine*, 12(2):34–45, May 2017.
- [30] T. N. Ferreira and S. R. Vergilio. Utilizando otimização por colônia de formigas na seleção de produtos para o teste de mutação do diagrama de características. In *Proceedings of the 6th Brazilian Workshop on Search-Based Software Engineering (WESB '15)*, volume 1, pages 61–70, Belo Horizonte, Brazil, 2015. In Portuguese.
- [31] T. N. Ferreira, S. R. Vergilio, and J. T. de Souza. Incorporating user preferences in search based software engineering: A systematic mapping study. *Information and Software Technology*, pages 55–69, October 2017.
- [32] H. L. Jakubovski Filho, T. N. Ferreira, and S. R. Vergilio. Multiple objective test set selection for software product line testing: Evaluating different preference-based algorithms. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering (SBES '18)*, pages 162–171, Sao Carlos, Brazil, 2018. ACM.
- [33] H. L. Jakubovski Filho, T. N. Ferreira, and S. R. Vergilio. Preference based multi-objective algorithms applied to the variability testing of software product lines. *Journal of Systems and Software*, 151:194–209, May 2018.
- [34] I. K. Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Lab., CA (US), 2002.
- [35] D. Hadka. MOEA Framework: A free and open source Java framework for multiobjective optimization. user manual. <http://www.moeaframework.org/>, 2016.
- [36] L. A. Hannah. Stochastic optimization. *International Encyclopedia of the Social & Behavioral Sciences*, 2:473–481, April 2015.
- [37] M. Harman. Software engineering: An ideal set of challenges for evolutionary computation. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '13)*, pages 1759–1760, Amsterdam, The Netherlands, 2013. ACM.
- [38] M. Harman and B. F. Jones. Search-based software engineering. *Information and Software Technology*, 43:833–839, December 2001.
- [39] M. Harman, S. A. Mansouri, and Y. Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys*, 45(1):11, November 2012.
- [40] C. Henard, M. Papadakis, and Y. Le Traon. Mutation-based generation of software product line test configurations. In *Proceedings of the 6th International Symposium on Search Based Software Engineering (SSBSE '14)*, pages 92–106, Fortaleza, Brazil, 2014. Springer.



- [41] C. Henard, M. Papadakis, G. Perrouin, J. Klein, P. Heymans, and Y. Le Traon. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. *IEEE Transactions on Software Engineering*, 40(7):650–670, May 2014.
- [42] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. Le Traon. Assessing software product line testing via model-based mutation: An application to similarity testing. In *Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW '13)*, pages 188–197, Luxembourg, 2013. IEEE.
- [43] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. Le Traon. Multi-objective test generation for software product lines. In *Proceedings of the 17th International Software Product Line Conference (SPLC '13)*, pages 62–71, Tokyo, Japan, 2013. ACM.
- [44] R. M. Hierons, M. Li, X. Liu, S. Segura, and W. Zheng. SIP: Optimal product selection from feature models using many-objective evolutionary optimization. *ACM Transactions on Software Engineering and Methodology*, 25(2):17:1–17:39, April 2016.
- [45] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
- [46] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima. Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190, July 2016.
- [47] H. Jain and K. Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, Aug 2014.
- [48] H. L. Jakubovski Filho, T. N. Ferreira, and S. R. Vergilio. Incorporating user preferences in a software product line testing hyper-heuristic approach. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '18)*, pages 1–8, Rio de Janeiro, Brazil, July 2018. IEEE.
- [49] K. C. Kang, J. Lee, and P. Donohoe. Feature-oriented project line engineering. *IEEE Software*, 19(4):58–65, July 2002.
- [50] R. Kuhn, R. Kacker, Y. Lei, and J. Hunter. Combinatorial software testing. *Computer*, 42(8):94–96, August 2009.
- [51] B. Li, J. Li, K. Tang, and X. Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys*, 48(1):13, September 2015.
- [52] K. Li, K. Deb, and X. Yao. R-metric: Evaluating the performance of preference-based evolutionary multiobjective optimization using reference points. *IEEE Transactions on Evolutionary Computation*, 22(6):821–835, September 2017.
- [53] R. E Lopez-Herrejon, F. Chicano, J. Ferrer, A. Egyed, and E. Alba. Multi-objective optimal test suite computation for software product line pairwise testing. In *Proceedings of the 2013 IEEE International Conference on Software Maintenance (ICSM '13)*, pages 404–407, Eindhoven, The Netherlands, 2013. IEEE.

- [54] R. A. Matnei Filho and S. R. Vergilio. A mutation and multi-objective test data generation approach for feature testing of software product lines. In *Proceedings of the 29th Brazilian Symposium on Software Engineering (SBES'15)*, pages 21–30, Belo Horizonte, Brazil, September 2015. IEEE Computer Society.
- [55] R. A. Matnei Filho and S. R. Vergilio. A multi-objective test data generation approach for mutation testing of feature models. *Journal of Software Engineering Research and Development*, 4(1):1–29, July 2016.
- [56] M. Mendonça, M. Branco, and D. Cowan. SPLOT: software product lines online tools. In *Proceedings of the 24th ACM Conference Companion on Object Oriented Programming Systems Languages and Applications (OOPSLA '09)*, pages 761–762, Orlando, USA, 2009.
- [57] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Springer Science & Business Media, 2012.
- [58] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, March 1956.
- [59] M. W. Mkaouer, M. Kessentini, S. Bechikh, K. Deb, and M. Ó Cinnéide. High dimensional search-based software engineering: Finding tradeoffs among 15 objectives for automating software refactoring using NSGA-III. In *Proceedings of the 16th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '14)*, pages 1263–1270, Vancouver, Canada, 2014. ACM.
- [60] M. Ohsaki, H. Takagi, and K. Ohya. An input method using discrete fitness values for interactive GA. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, 6(1):131–145, January 1998.
- [61] S. Oster, M. Zink, M. Lochau, and M. Grechanik. Pairwise feature-interaction testing for spls: Potentials and limitations. In *Proceedings of the 15th International Software Product Line Conference (SPLC '11)*, page 6, Munich, Germany, 2011. ACM.
- [62] J.A. Parejo, A.B. Sánchez, S. Segura, A. Ruiz-Cortés, R. Lopez-Herrejon, and A. Egyed. Multi-objective test case prioritization in highly configurable systems: A case study. *Journal of Systems and Software*, 122:287–310, September 2016.
- [63] J. A. Pereira, L. Maciel, T. F. Noronha, and E. Figueiredo. Heuristic and exact algorithms for product configuration in software product lines. *International Transactions in Operational Research*, 24(6):1285–1306, May 2017.
- [64] G. Perrouin, S. Sen, J. Klein, B. Baudry, and Y. Le Traon. Automated and scalable t-wise test case generation strategies for software product lines. In *Proceedings of the 3rd IEEE International Conference on Software Testing, Verification and Validation (ICST '10)*, pages 459–468, Paris, France, 2010. IEEE.
- [65] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE '08)*, volume 8, pages 68–77, University of Bari, Italy, 2008.

- [66] D. Reuling, J. Bürdek, S. Rotärmel, M. Lochau, and U. Kelter. Fault-based product-line testing: Effective sample generation based on feature-diagram mutation. In *Proceedings of the 19th International Software Product Line Conference (SPLC '15)*, pages 131–140, Nashville, Tennessee, 2015. ACM.
- [67] T. L. Saaty. How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, 48(1):9–26, September 1990.
- [68] N. Salkind. *Encyclopaedia of research design, vol. 1*. SAGE, 2010.
- [69] D. K. Saxena and K. Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO '07)*, pages 772–787, Matsushima, Japan, 2007. Springer.
- [70] A. S. Sayyad and H. Ammar. Pareto-optimal search-based software engineering (POSBSE): A literature survey. In *Proceedings of the 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE' 03)*, pages 21–27, San Francisco, USA, 2013. IEEE.
- [71] S. Segura, R. M. Hierons, D. Benavides, and A. Ruiz-Cortés. Automated test data generation on the analyses of feature models: A metamorphic testing approach. In *Proceedings of the 3rd International Conference on Software Testing, Verification and Validation (ICST '10)*, pages 35–44, Paris, France, 2010. IEEE.
- [72] M. Shackelford and D. W. Corne. *A technique for evaluation of interactive evolutionary systems*. Springer, 2004.
- [73] A. Sinha, D. K. Saxena, K. Deb, and A. Tiwari. Using objective reduction and interactive procedure to handle many-objective optimization problems. *Applied Soft Computing*, 13(1):415–427, January 2013.
- [74] A. Strickler, J. A Prado Lima, S. R. Vergilio, and A. Pozo. Deriving products for variability test of feature models with a hyper-heuristic approach. *Applied Soft Computing*, 49:1232–1242, December 2016.
- [75] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, September 2001.
- [76] P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura, and A. Jimenez. FaMa framework. In *Proceedings of the 12th International Software Product Line Conference (SPLC '08)*, pages 359–359, Limerick, Ireland, 2008. IEEE.
- [77] E. Uzuncaova, S. Khurshid, and D. Batory. Incremental test generation for software product lines. *IEEE Transactions on Software Engineering*, 36(3):309–322, April 2010.
- [78] F. J. Van der Linden, K. Schmid, and E. Rommes. *Software product lines in action: The best industrial practice in product line engineering*. Springer Science & Business Media, 2007.

- [79] P. A. Vikhar. Evolutionary algorithms: A critical review and its future prospects. In *Proceedings of the International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC '16)*, pages 261–265, Jalgaon, India, 2016. IEEE.
- [80] D. J. Walker, R. Everson, and J. E. Fieldsend. Visualizing mutually nondominating solution sets in many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(2):165–184, November 2012.
- [81] H. Wang and M. Kessentini. Improving web services design quality using dimensionality reduction techniques. In *Proceedings of the 15th International Conference on Service-Oriented Computing (ICSOC '17)*, pages 499–507, Malaga, Spain, 2017. Springer.
- [82] S. Wang, S. Ali, and A. Gotlieb. Minimizing test suites in software product lines using weight-based genetic algorithms. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '13)*, pages 1493–1500, Amsterdam, The Netherlands, 2013. ACM.
- [83] S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan, and M. Liaaen. Multi-objective test prioritization in software product line testing: An industrial case study. In *Proceedings of the 18th International Software Product Line Conference (SPLC '14)*, pages 32–41, Florence, Italy, 2014. ACM.
- [84] S. Weißleder, D. Sokenou, and B. Schlingloff. Reusing state machines for automatic test generation in product lines. In *Proceedings of the 1st Workshop on Model-based Testing in Practice (MoTiP '08)*, pages 19 – 28, Berlin, Germany, 2008.
- [85] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14)*, page 38, London, United Kingdom, 2014. Citeseer.
- [86] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [87] J. Zhang and L. Xing. A survey of multiobjective evolutionary algorithms. In *Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE' 17)*, volume 1, pages 93–100, Guangzhou, China, July 2017.
- [88] Y. Zhang, M. Harman, and S. A. Mansouri. The multi-objective next release problem. In *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '07)*, pages 1129–1137, London, United Kingdom, 2007. ACM.
- [89] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, March 2011.
- [90] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999.
- [91] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, May 2003.

## APPENDIX A – ALGORITHMS QUESTIONNAIRE

### A.1 PREFERRED OBJECTIVES

Number of Products	Alive Mutants	Uncovered Pairs	Similarity	Cost	Unselected Features	Unimportant Features
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### A.2 MANUALLY

**Question 1:** Solution:

---

**Question 2:** Motivation:

---

**Question 3:** Time:

---

**Question 4:** How difficult was it to select this solution?

very difficult	difficult	neutral	easy	very easy
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### A.3 NSGA-II

**Question 1:** Solution:

---

**Question 2:** Motivation:

---

**Question 3:** Time:

---

**Question 4:** How difficult was it to select this solution?

very difficult	difficult	neutral	easy	very easy
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## A.4 R-NSGA-II

**Question 1:** Solution:

---

**Question 2:** Motivation:

---

**Question 3:** Time:

---

**Question 4:** How difficult was it to select this solution?

very difficult	difficult	neutral	easy	very easy
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## A.5 COR-NSGA-II

**Question 1:** Solution:

---

**Question 2:** Motivation:

---

**Question 3:** Time:

---

**Question 4:** How difficult was it to select this solution?

very difficult	difficult	neutral	easy	very easy
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



## APPENDIX B – NAUTILUS QUESTIONNAIRE

### Warning <sup>a</sup>

- Please decide spontaneously. Don't think too long about your decision to make sure that you convey your original impression.
- Sometimes you may not be completely sure about your agreement with a particular question or you may find that the question does not apply completely to the tool. Nevertheless, please tick a square in every question.
- It is your personal opinion that counts. Please remember: there is no wrong or right answer!

<sup>a</sup><https://www.ueq-online.org>

**Question 1:** How much time did you spend to get familiar with the tool?

<b>&lt; 5 min</b>	<b>6 min - 10 min</b>	<b>11 min - 20 min</b>	<b>20 min - 30 min</b>	<b>&gt; 31 min</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 2:** How difficult was it to learning to operate the tool?

<b>very difficult</b>	<b>difficult</b>	<b>neutral</b>	<b>easy</b>	<b>very easy</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 3:** The tool has a user friendly interface

<b>strongly agree</b>	<b>agree</b>	<b>neutral</b>	<b>disagree</b>	<b>strongly disagree</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 4:** The tool is easy to navigate

<b>strongly agree</b>	<b>agree</b>	<b>neutral</b>	<b>disagree</b>	<b>strongly disagree</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



**Question 5:** Error messages are helpful

<b>strongly agree</b>	<b>agree</b>	<b>neutral</b>	<b>disagree</b>	<b>strongly disagree</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 6:** What is your opinion about organization of information on the screen?

<b>very clear</b>	<b>clear</b>	<b>neutral</b>	<b>confusing</b>	<b>very confusing</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 7:** How difficult was it to understand the task you were asked to do?

<b>very difficult</b>	<b>difficult</b>	<b>neutral</b>	<b>easy</b>	<b>very easy</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 8:** How difficult was it to locate and identify relevant solutions?

<b>very difficult</b>	<b>difficult</b>	<b>neutral</b>	<b>easy</b>	<b>very easy</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 9:** How difficult was it to use the visualization support for the Pareto-front?

<b>very difficult</b>	<b>difficult</b>	<b>neutral</b>	<b>easy</b>	<b>very easy</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 10:** How much time did you spend to explore the Pareto-front using the visualization support?

<b>&lt; 5 min</b>	<b>6 min - 10 min</b>	<b>11 min - 20 min</b>	<b>20 min - 30 min</b>	<b>&gt; 31 min</b>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 11:** What do you find best about the tool? Enumerate your preferences where 1 means the best feature and so on.

- Ease of use
- Interface
- Cloud computing-based
- Pareto-front visualization
- Algorithms provided
- Other. What? \_\_\_\_\_

**Question 12:** Further Comments.

*Suggestions to improve the tool? Were the questions easy to understand? Did you have enough time to finish the activity? What were the most difficult or easiest tasks in the activities? etc.*

---

---

---

---

---

---

---

---

---

---

---

## APPENDIX C – FEATURE MODELS

In the next, we present feature models of the SPLs used in our experiments.

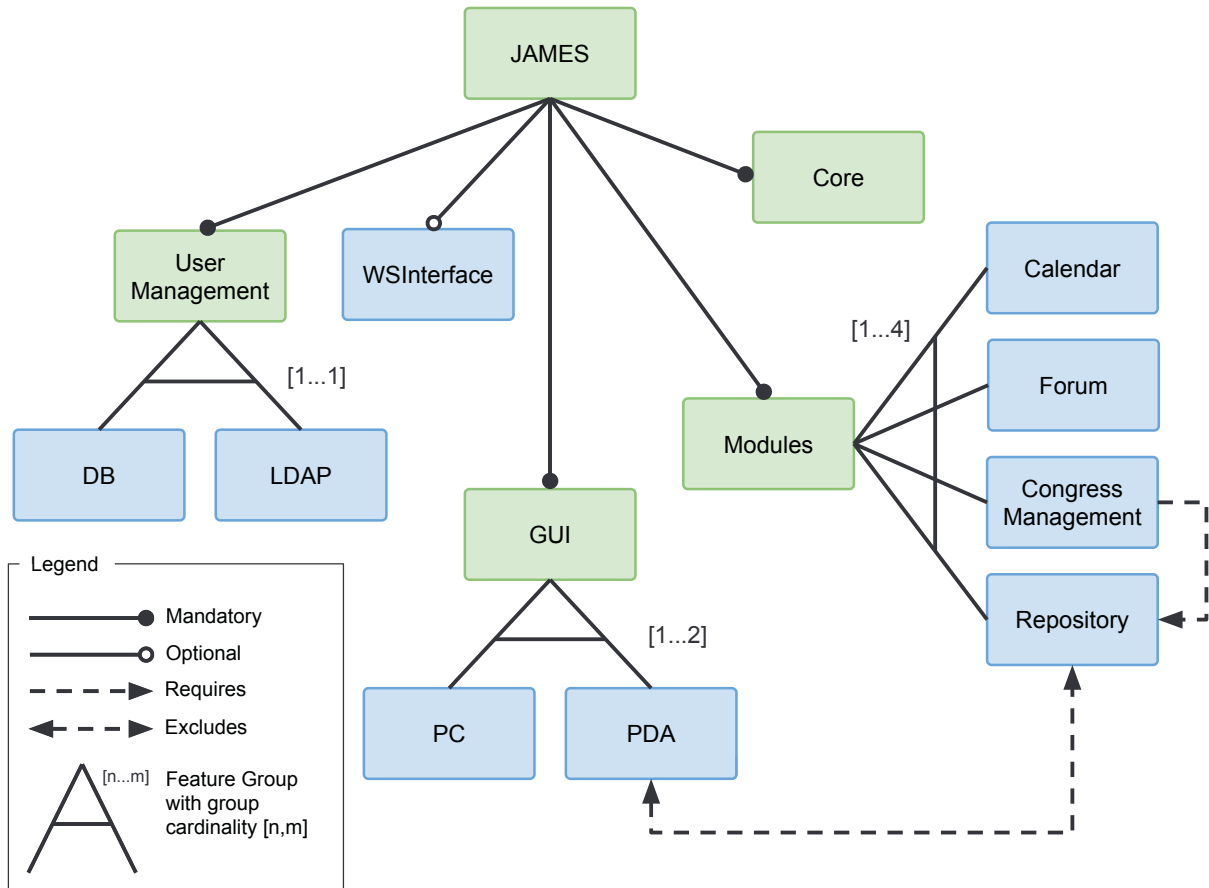


Figure C.1: Feature Model for James (Adapted from [6]).

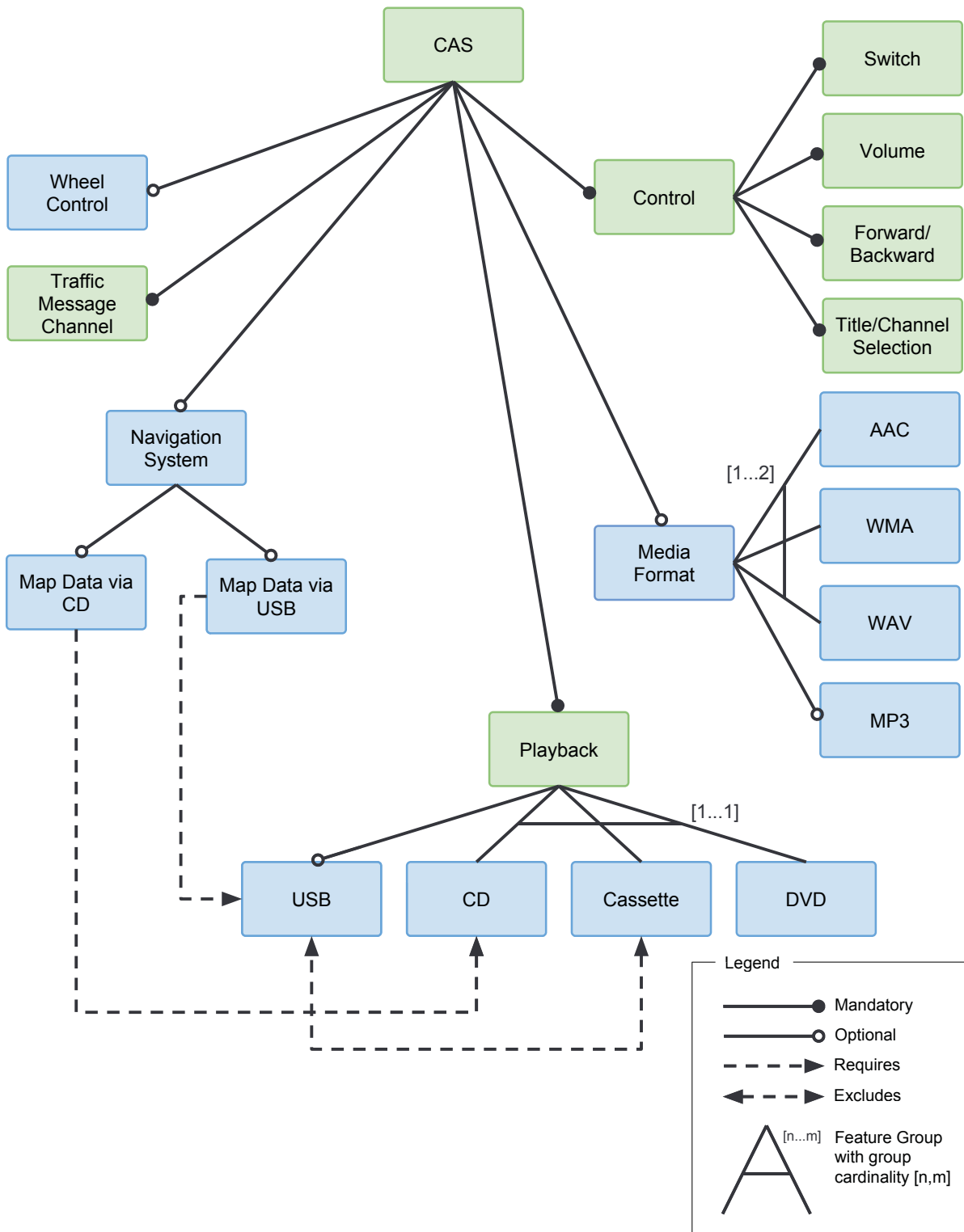


Figure C.2: Feature Model for CAS (Adapted from [84]).

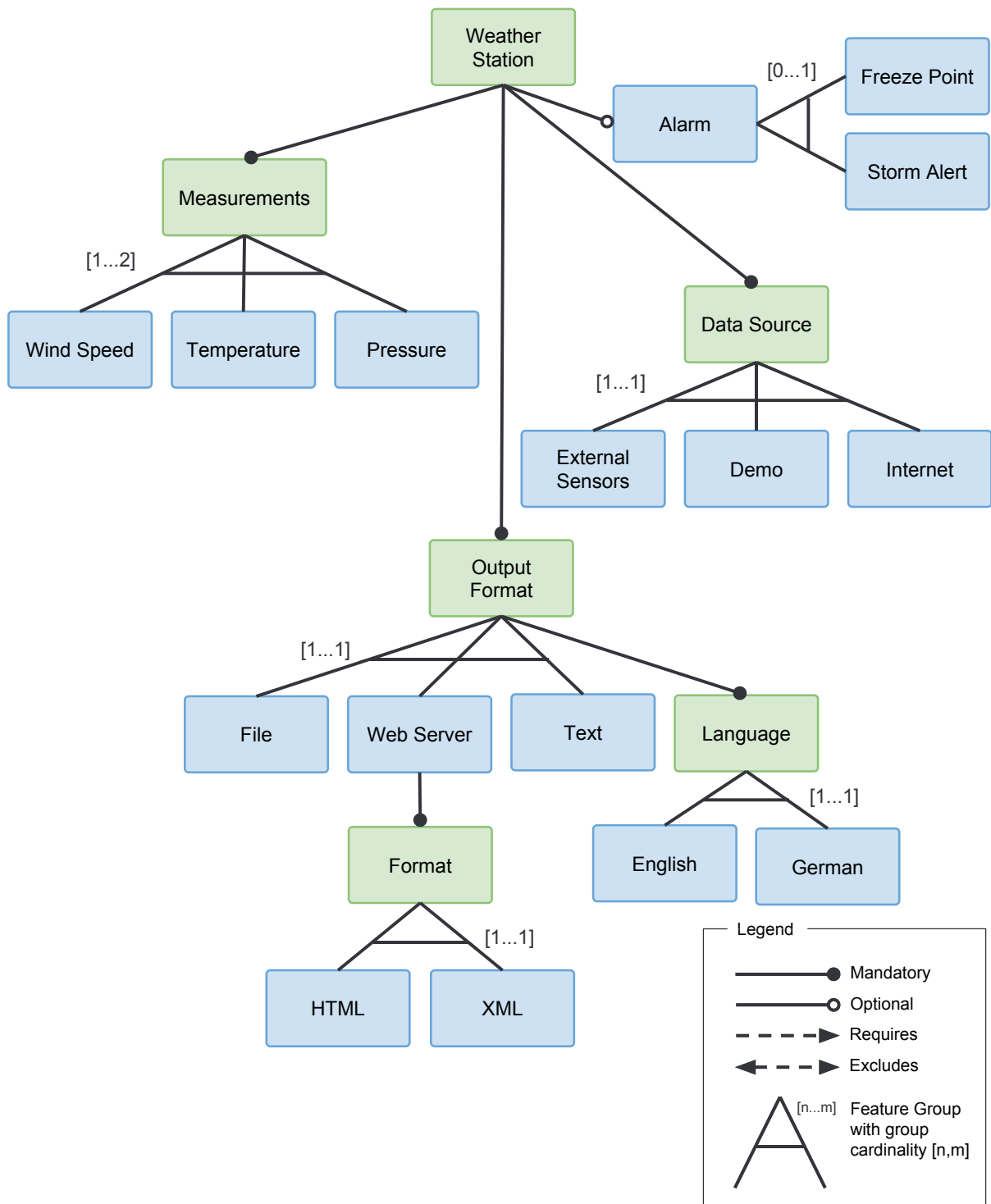


Figure C.3: Feature Model for WS (Adapted from [7]).

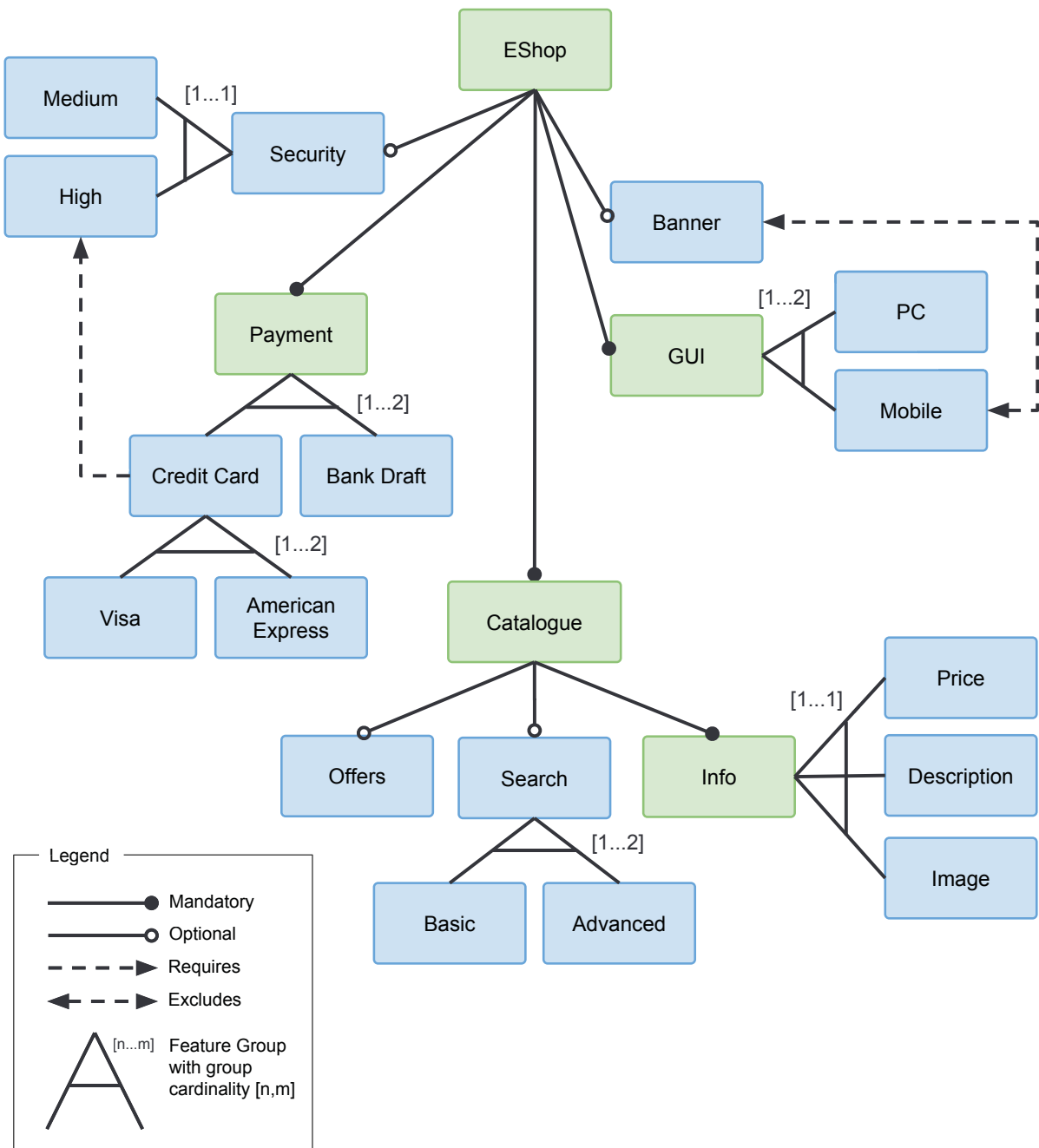


Figure C.4: Feature Model for E-Shop (Adapted from [71]).

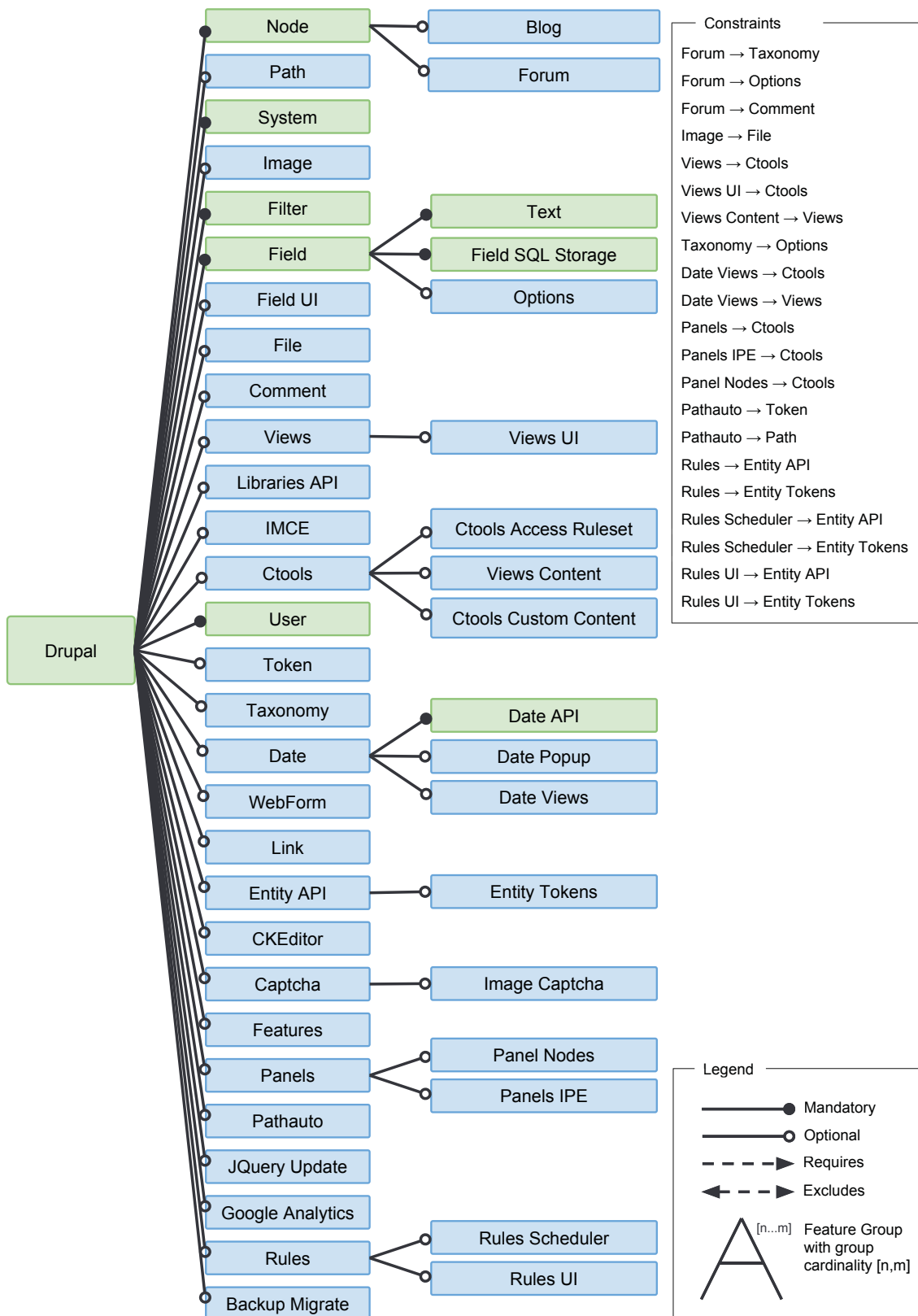


Figure C.5: Feature Model for Drupal (Adapted from [62])

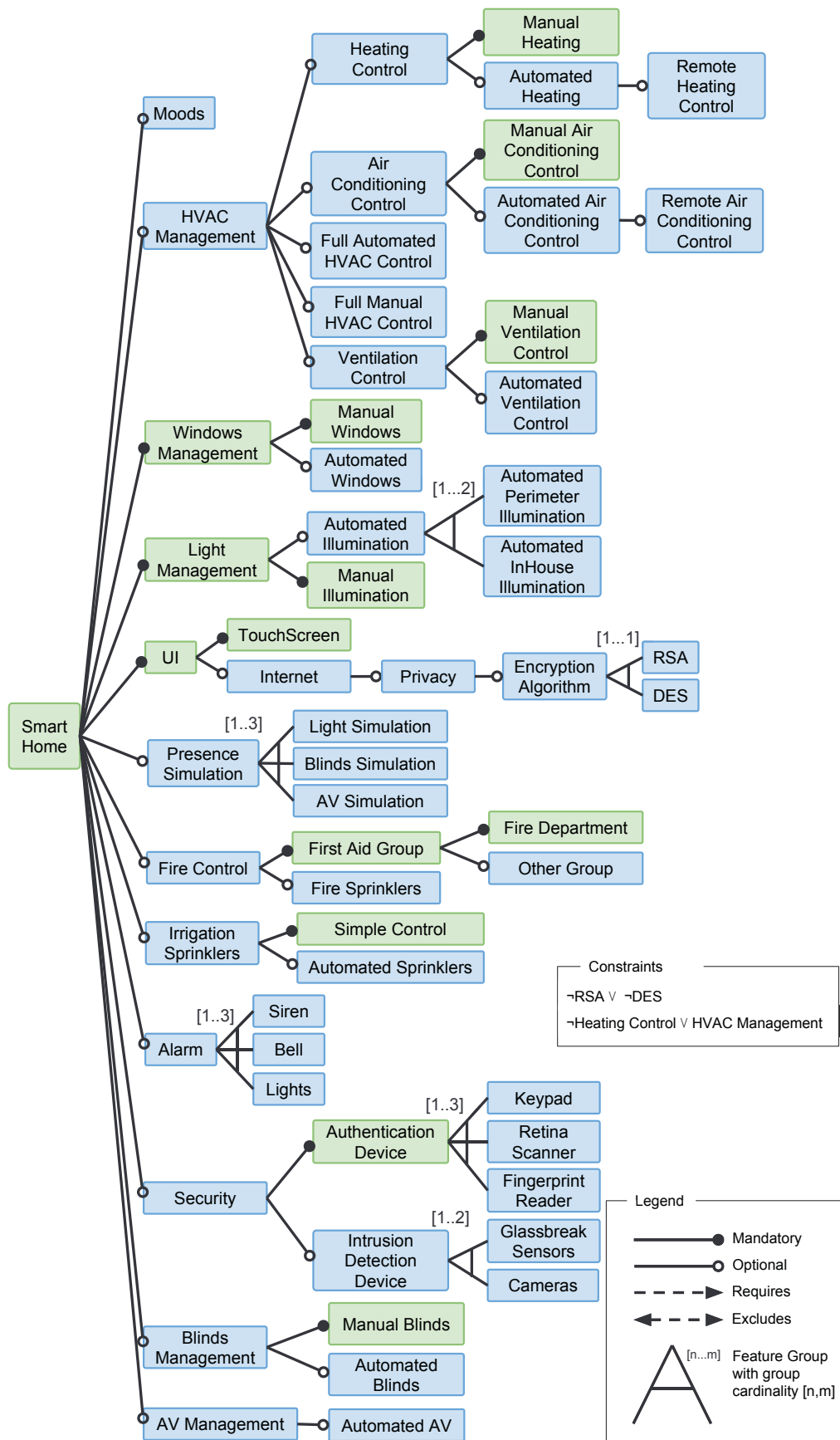


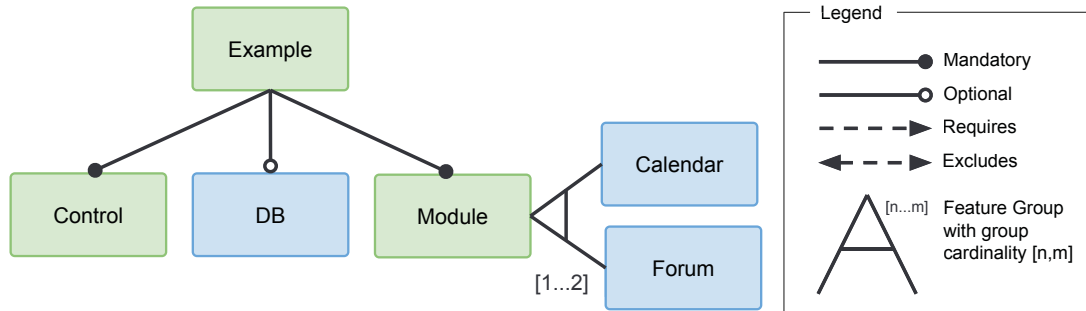
Figure C.6: Feature Model for Smarthome (Adapted from [43]).





## APPENDIX E – PRE-STUDY QUESTIONNAIRE

### E.1 FEATURE MODEL



### E.2 GENERAL INFORMATION

Feature	Cost	Import.
Example	3	0
Control	4	2
DB	2	3
Module	3	0
Calendar	2	4
Forum	1	3

**# of mutants:** 10

**# of pairs:** 10

**Mutants:** {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

**Alive Mutants:** {3, 4, 5, 8, 9, 10}

**Equivalent Mutants:** {1, 2, 6, 7}

**Pairs:** {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

### E.3 PRODUCTS

#	Features	Killed Mutants	Covered Pairs	Cost	Import.
1	Example, Control, Module, Calendar, Forum, DB	9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	15	12
2	Example, Control, Module, Calendar, Forum	3, 9	2, 3, 4, 8, 9, 10	13	9
3	Example, Control, Module, Calendar, DB	4, 8	1, 2, 3, 5, 6, 8	14	9
4	Example, Control, Module, Calendar	3, 4, 8	2, 3, 8	12	6
5	Example, Control, Module, Forum, DB	5, 8, 10	1, 2, 4, 5, 7, 9	13	8
6	Example, Control, Module, Forum	3, 5, 8	2, 4, 9	11	5
		<b>Sum</b>		<b>78</b>	<b>49</b>

**Possible Solutions:** {1, 2}, {3, 4, 5}, {1}, {2, 4, 5, 6}, etc.

## E.4 QUESTIONS

**Question 1: (2 Points)** How many optional features does the feature model have? Choose correct option.

- (a) 6
- (b) 1
- (c) 3
- (d) 2

**Question 2: (2 Points)** What does a mutant feature model mean? Choose correct option.

- (a) A modified version of the original diagram aiming to describe a fault
- (b) A diagram used to represent commonalities and variabilities, and also to derive products for testing
- (c) A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system
- (d) None of the above

**Question 3: (2 Points)** Please indicate how much you agree or disagree with the following sentence "*It is possible to generate multiple solutions from this feature model*"

strongly agree	agree	neutral	disagree	strongly disagree
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Question 4: (2 Points)** Provide a solution in which 100% of the **killed mutants** are covered. What is the cost value of this solution?

Product #1	Product #2	Product #3	Product #4	Product #5	Product #6
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Cost: \_\_\_\_\_

**Question 5: (2 Points)** Provide a solution in which 100% of the **features** are covered. What is the importance value of this solution?

Product #1	Product #2	Product #3	Product #4	Product #5	Product #6
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Importance: \_\_\_\_\_

## APPENDIX F – CONSENT TERM

You are being invited to participate in a research study. This one is being done by Thiago do Nascimento Ferreira from the Federal University of Parana and supervised by Prof. Silvia Regina Vergilio from the same university.

The purpose of this research is to evaluate Nautilus, a Java web platform tool, and the impact of the human preferences during the optimization process by reducing the dimensionality of the problem. If you agree to the term and participate in the study you will be asked to complete some online surveys/questionnaires.

The participants are assured that they will receive answers to any questions and clarify any questions regarding research-related subjects. The aforementioned researchers also undertake to provide up-to-date information obtained during the study.

The participants are also free to withdraw their consent at any time and to stop participating in the study, causing no burden or harm.

Responses will be completely anonymous and your name will not appear anywhere on the research' results. However, this consent form indicates that the results can be presented at scientific events or publications.

By checking “I agree” below you are indicating that you are at least 18 years old, have read and understood this consent form and agree to participate in this research study.

- I agree to the terms and conditions
- I disagree

## APPENDIX G – RQ2 DETAILED RESULTS

Table G.1: COR-NSGA-II Versus NSGA-II and NSGA-III in Scenario 2D.

	RPAlgorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time	
James	Restric.	nsga-ii	156E-3±5.4E-3	39.8E-3±410E-6	106E0±1.66E0	5.87E0±2.45E0	11.7E3±829E0
		nsga-iii	137E-3±15.2E-3	41.8E-3±1.84E-3	55.7E0±11.4E0	29.6E0±20.1E0	19.4E3±1.57E3
		cor-nsga-ii	<b>157E-3±310E-6</b>	<b>38.5E-3±40E-6</b>	<b>7.53E0±1.07E0</b>	<b>3.47E0±629E-3</b>	<b>3.55E3±670E0</b>
	Comp.	nsga-ii	<b>308E-3±9.01E-3</b>	<b>14.4E-3±670E-6</b>	106E0±1.66E0	12.9E0±2.69E0	11.7E3±829E0
		nsga-iii	226E-3±31.4E-3	18.2E-3±1.51E-3	55.7E0±11.4E0	30.4E0±18.8E0	19.4E3±1.57E3
		cor-nsga-ii	156E-3±2.91E-3	22.1E-3±70E-6	<b>7.53E0±1.07E0</b>	<b>1.17E0±379E-3</b>	<b>3.55E3±670E0</b>
CAS	Restric.	nsga-ii	70.4E-3±3.5E-3	55.5E-3±860E-6	109E0±1.69E0	90.8E0±12.2E0	116E3±8.54E3
		nsga-iii	86.1E-3±6E-3	51.9E-3±1.18E-3	72E0±4.03E0	72E0±4.03E0	99.6E3±6.99E3
		cor-nsga-ii	<b>185E-3±490E-6</b>	<b>39.1E-3±50E-6</b>	<b>20.4E0±5.47E0</b>	<b>13.6E0±4.19E0</b>	<b>9.77E3±2.44E3</b>
	Comp.	nsga-ii	149E-3±2.69E-3	10.3E-3±20E-6	109E0±1.69E0	103E0±8.76E0	116E3±8.54E3
		nsga-iii	<b>181E-3±5.72E-3</b>	<b>9.1E-3±240E-6</b>	72E0±4.03E0	72E0±4.03E0	99.6E3±6.99E3
		cor-nsga-ii	151E-3±1.38E-3	10E-3±40E-6	<b>20.4E0±5.47E0</b>	<b>14.7E0±4.1E0</b>	<b>9.77E3±2.44E3</b>
WS	Restric.	nsga-ii	64.9E-3±3.7E-3	65E-3±1.11E-3	110E0±1.25E0	97E0±10.3E0	133E3±7.82E3
		nsga-iii	102E-3±4.89E-3	55.6E-3±930E-6	74E0±3.8E0	74E0±3.8E0	96.7E3±5.9E3
		cor-nsga-ii	<b>189E-3±700E-6</b>	<b>44.5E-3±80E-6</b>	<b>20.5E0±3.74E0</b>	<b>12.5E0±2.34E0</b>	<b>12.4E3±501E0</b>
	Comp.	nsga-ii	148E-3±2.17E-3	46.1E-3±610E-6	110E0±1.25E0	106E0±8.97E0	133E3±7.82E3
		nsga-iii	142E-3±1.36E-3	46.5E-3±330E-6	74E0±3.8E0	74E0±3.8E0	96.7E3±5.9E3
		cor-nsga-ii	<b>154E-3±600E-6</b>	<b>45.1E-3±120E-6</b>	<b>20.5E0±3.74E0</b>	<b>12.5E0±2.34E0</b>	<b>12.4E3±501E0</b>
E-Shop	Restric.	nsga-ii	63E-3±2.64E-3	92.3E-3±1.08E-3	112E0±0E0	111E0±2.05E0	399E3±23.8E3
		nsga-iii	80E-3±3.52E-3	84.8E-3±1.21E-3	71.1E0±4.63E0	71.1E0±4.63E0	319E3±16.5E3
		cor-nsga-ii	<b>180E-3±1.09E-3</b>	<b>62.8E-3±150E-6</b>	<b>5.93E0±4.76E0</b>	<b>4.37E0±2.67E0</b>	<b>43.4E3±1.82E3</b>
	Comp.	nsga-ii	<b>145E-3±1.72E-3</b>	14.5E-3±40E-6	112E0±0E0	112E0±1.83E0	399E3±23.8E3
		nsga-iii	141E-3±4.6E-3	<b>14.4E-3±210E-6</b>	71.1E0±4.63E0	71.1E0±4.63E0	319E3±16.5E3
		cor-nsga-ii	137E-3±840E-6	14.5E-3±60E-6	<b>5.93E0±4.76E0</b>	<b>4.37E0±2.67E0</b>	<b>43.4E3±1.82E3</b>
Drupal	Restric.	nsga-ii	49.1E-3±500E-6	80.6E-3±210E-6	112E0±0E0	112E0±0E0	21E6±1.11E6
		nsga-iii	54.6E-3±850E-6	77.9E-3±360E-6	72E0±4.98E0	72E0±4.98E0	19.1E6±1.41E6
		cor-nsga-ii	<b>83.7E-3±410E-6</b>	<b>68E-3±110E-6</b>	<b>1.1E0±305E-3</b>	<b>1.1E0±305E-3</b>	<b>7.94E6±296E3</b>
	Comp.	nsga-ii	<b>139E-3±350E-6</b>	<b>12.3E-3±0E0</b>	112E0±0E0	112E0±0E0	21E6±1.11E6
		nsga-iii	138E-3±260E-6	<b>12.3E-3±0E0</b>	72E0±4.98E0	72E0±4.98E0	19.1E6±1.41E6
		cor-nsga-ii	137E-3±10E-6	<b>12.3E-3±0E0</b>	<b>1.1E0±305E-3</b>	<b>1.1E0±305E-3</b>	<b>7.94E6±296E3</b>
Smarthome	Restric.	nsga-ii	49E-3±620E-6	61.9E-3±200E-6	112E0±0E0	112E0±0E0	30.5E6±1.65E6
		nsga-iii	54.7E-3±1.05E-3	59.8E-3±340E-6	73.5E0±5.61E0	73.5E0±5.61E0	27.5E6±1.03E6
		cor-nsga-ii	<b>83.6E-3±420E-6</b>	<b>52.3E-3±90E-6</b>	<b>1.2E0±407E-3</b>	<b>1.17E0±379E-3</b>	<b>10.9E6±559E3</b>
	Comp.	nsga-ii	<b>139E-3±470E-6</b>	<b>9.69E-3±0E0</b>	112E0±0E0	112E0±0E0	30.5E6±1.65E6
		nsga-iii	138E-3±230E-6	<b>9.69E-3±0E0</b>	73.5E0±5.61E0	73.5E0±5.61E0	27.5E6±1.03E6
		cor-nsga-ii	137E-3±10E-6	<b>9.69E-3±0E0</b>	<b>1.2E0±407E-3</b>	<b>1.17E0±379E-3</b>	<b>10.9E6±559E3</b>

Table G.2: COR-NSGA-II Versus NSGA-II and NSGA-III in Scenario 3D.

	RPAlgorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time	
James	Restrict.	nsga-ii	94.6E-3±3.75E-3	62.3E-3±810E-6	106E0±1.77E0	3.63E0±718E-3	11.8E3±879E0
		nsga-iii	69.9E-3±39.4E-3	71.8E-3±12.6E-3	55.3E0±10.2E0	20.3E0±16.2E0	19.2E3±1.66E3
		cor-nsga-ii	<b>120E-3±1.09E-3</b>	<b>57.7E-3±280E-6</b>	67.2E0±2E0	11.1E0±1.66E0	<b>4.45E3±511E0</b>
	Comp.	nsga-ii	205E-3±13.7E-3	<b>39.3E-3±1.76E-3</b>	106E0±1.77E0	<b>4.77E0±1.57E0</b>	11.8E3±879E0
		nsga-iii	149E-3±61.4E-3	49.1E-3±10.9E-3	55.3E0±10.2E0	20.8E0±15.8E0	19.2E3±1.66E3
		cor-nsga-ii	<b>207E-3±4.85E-3</b>	39.4E-3±690E-6	67.2E0±2E0	13.2E0±1.88E0	<b>4.45E3±511E0</b>
CAS	Restrict.	nsga-ii	14.2E-3±330E-6	91.8E-3±580E-6	110E0±1.87E0	96.8E0±15.4E0	116E3±5.41E3
		nsga-iii	18.5E-3±740E-6	87.5E-3±790E-6	<b>70.9E0±4.14E0</b>	70.9E0±4.14E0	100E3±5.15E3
		cor-nsga-ii	<b>126E-3±2.22E-3</b>	<b>53E-3±400E-6</b>	96.4E0±2.98E0	<b>12.4E0±1.45E0</b>	<b>9.39E3±2.16E3</b>
	Comp.	nsga-ii	43.9E-3±950E-6	57.1E-3±610E-6	110E0±1.87E0	96.8E0±15.4E0	116E3±5.41E3
		nsga-iii	54.5E-3±1.82E-3	53.8E-3±720E-6	<b>70.9E0±4.14E0</b>	70.9E0±4.14E0	100E3±5.15E3
		cor-nsga-ii	<b>182E-3±1.98E-3</b>	<b>33.8E-3±240E-6</b>	96.4E0±2.98E0	<b>12.5E0±1.66E0</b>	<b>9.39E3±2.16E3</b>
WS	Restrict.	nsga-ii	14.1E-3±270E-6	81.2E-3±450E-6	110E0±1.53E0	106E0±6.6E0	135E3±7.4E3
		nsga-iii	13.1E-3±180E-6	81.8E-3±230E-6	<b>73.5E0±3.96E0</b>	73.5E0±3.96E0	96.1E3±6.14E3
		cor-nsga-ii	<b>123E-3±990E-6</b>	<b>47.1E-3±200E-6</b>	100E0±2.84E0	<b>15.6E0±1.61E0</b>	<b>12.5E3±327E0</b>
	Comp.	nsga-ii	43.6E-3±780E-6	58.5E-3±560E-6	110E0±1.53E0	106E0±6.6E0	135E3±7.4E3
		nsga-iii	40.8E-3±500E-6	59.3E-3±290E-6	<b>73.5E0±3.96E0</b>	73.5E0±3.96E0	96.1E3±6.14E3
		cor-nsga-ii	<b>182E-3±1.99E-3</b>	<b>34.2E-3±180E-6</b>	100E0±2.84E0	<b>17.5E0±1.81E0</b>	<b>12.5E3±327E0</b>
E-Shop	Restrict.	nsga-ii	13.6E-3±270E-6	79.4E-3±440E-6	112E0±379E-3	112E0±802E-3	399E3±28.2E3
		nsga-iii	12.8E-3±110E-6	79.8E-3±130E-6	71E0±6.64E0	71E0±6.64E0	323E3±12.5E3
		cor-nsga-ii	<b>23.4E-3±4.46E-3</b>	<b>71.6E-3±2.75E-3</b>	<b>29.6E0±11.6E0</b>	<b>21.8E0±6.42E0</b>	<b>44.9E3±2.18E3</b>
	Comp.	nsga-ii	42.3E-3±820E-6	60.6E-3±590E-6	112E0±379E-3	112E0±802E-3	399E3±28.2E3
		nsga-iii	40.1E-3±300E-6	61.1E-3±170E-6	71E0±6.64E0	71E0±6.64E0	323E3±12.5E3
		cor-nsga-ii	<b>66.5E-3±10.8E-3</b>	<b>53E-3±2.78E-3</b>	<b>29.6E0±11.6E0</b>	<b>21.8E0±6.42E0</b>	<b>44.9E3±2.18E3</b>
Drupal	Restrict.	nsga-ii	<b>12.7E-3±80E-6</b>	<b>114E-3±80E-6</b>	112E0±0E0	112E0±0E0	21.1E6±1.64E6
		nsga-iii	12.4E-3±30E-6	114E-3±20E-6	72.7E0±3.74E0	72.7E0±3.74E0	19.1E6±1.43E6
		cor-nsga-ii	12.3E-3±0E0	114E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>8.24E6±319E3</b>
	Comp.	nsga-ii	<b>39.9E-3±230E-6</b>	<b>22.5E-3±20E-6</b>	112E0±0E0	112E0±0E0	21.1E6±1.64E6
		nsga-iii	39.2E-3±100E-6	22.5E-3±10E-6	72.7E0±3.74E0	72.7E0±3.74E0	19.1E6±1.43E6
		cor-nsga-ii	38.9E-3±0E0	22.5E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>8.24E6±319E3</b>
Smarthome	Restrict.	nsga-ii	<b>12.7E-3±90E-6</b>	<b>135E-3±110E-6</b>	112E0±0E0	112E0±0E0	30.4E6±1.16E6
		nsga-iii	12.4E-3±30E-6	135E-3±30E-6	73.8E0±5.96E0	73.8E0±5.96E0	27.3E6±1.16E6
		cor-nsga-ii	12.3E-3±0E0	135E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>11.5E6±911E3</b>
	Comp.	nsga-ii	<b>39.9E-3±270E-6</b>	<b>17.3E-3±20E-6</b>	112E0±0E0	112E0±0E0	30.4E6±1.16E6
		nsga-iii	39.2E-3±80E-6	17.3E-3±0E0	73.8E0±5.96E0	73.8E0±5.96E0	27.3E6±1.16E6
		cor-nsga-ii	38.9E-3±0E0	17.3E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>11.5E6±911E3</b>

Table G.3: COR-NSGA-II Versus R-NSGA-II in Scenario 2D.

	RPAlgorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time	
James	Restric.	r-nsga-ii	96.8E-3±9.27E-3	46.5E-3±1.66E-3	110E0±2.46E0	110E0±2.46E0	16.7E3±903E0
		cor-nsga-ii	<b>157E-3±310E-6</b>	<b>38.5E-3±40E-6</b>	<b>7.53E0±1.07E0</b>	<b>3.47E0±629E-3</b>	<b>3.55E3±670E0</b>
	Comp.	r-nsga-ii	<b>196E-3±18E-3</b>	<b>19.3E-3±900E-6</b>	111E0±2.01E0	111E0±2.15E0	17.5E3±1.5E3
		cor-nsga-ii	156E-3±2.91E-3	22.1E-3±70E-6	<b>7.53E0±1.07E0</b>	<b>1.17E0±379E-3</b>	<b>3.55E3±670E0</b>
CAS	Restric.	r-nsga-ii	70.7E-3±4.13E-3	55.4E-3±980E-6	109E0±1.4E0	88.8E0±13.7E0	121E3±8.07E3
		cor-nsga-ii	<b>185E-3±490E-6</b>	<b>39.1E-3±50E-6</b>	<b>20.4E0±5.47E0</b>	<b>13.6E0±4.19E0</b>	<b>9.77E3±2.44E3</b>
	Comp.	r-nsga-ii	148E-3±2.82E-3	10.4E-3±20E-6	109E0±1.42E0	100E0±12.4E0	122E3±6.56E3
		cor-nsga-ii	<b>151E-3±1.38E-3</b>	<b>10E-3±40E-6</b>	<b>20.4E0±5.47E0</b>	<b>14.7E0±4.1E0</b>	<b>9.77E3±2.44E3</b>
WS	Restric.	r-nsga-ii	65.9E-3±3.39E-3	64.7E-3±990E-6	110E0±1.17E0	96.1E0±11.4E0	138E3±5.91E3
		cor-nsga-ii	<b>189E-3±700E-6</b>	<b>44.5E-3±80E-6</b>	<b>20.5E0±3.74E0</b>	<b>12.5E0±2.34E0</b>	<b>12.4E3±501E0</b>
	Comp.	r-nsga-ii	148E-3±2.33E-3	46.1E-3±600E-6	110E0±1.42E0	108E0±5.74E0	139E3±6.01E3
		cor-nsga-ii	<b>154E-3±600E-6</b>	<b>45.1E-3±120E-6</b>	<b>20.5E0±3.74E0</b>	<b>12.5E0±2.34E0</b>	<b>12.4E3±501E0</b>
E-Shop	Restric.	r-nsga-ii	63.3E-3±2.07E-3	92.2E-3±800E-6	112E0±254E-3	110E0±3.07E0	409E3±18.9E3
		cor-nsga-ii	<b>180E-3±1.09E-3</b>	<b>62.8E-3±150E-6</b>	<b>5.93E0±4.76E0</b>	<b>4.37E0±2.67E0</b>	<b>43.4E3±1.82E3</b>
	Comp.	r-nsga-ii	<b>146E-3±1.83E-3</b>	<b>14.5E-3±40E-6</b>	112E0±183E-3	112E0±254E-3	408E3±15E3
		cor-nsga-ii	137E-3±840E-6	14.5E-3±60E-6	<b>5.93E0±4.76E0</b>	<b>4.37E0±2.67E0</b>	<b>43.4E3±1.82E3</b>
Drupal	Restric.	r-nsga-ii	49.2E-3±600E-6	80.6E-3±250E-6	112E0±0E0	112E0±0E0	20.9E6±1.33E6
		cor-nsga-ii	<b>83.7E-3±410E-6</b>	<b>68E-3±110E-6</b>	<b>1.1E0±305E-3</b>	<b>1.1E0±305E-3</b>	<b>7.94E6±296E3</b>
	Comp.	r-nsga-ii	<b>139E-3±480E-6</b>	<b>12.3E-3±0E0</b>	112E0±0E0	112E0±0E0	20.7E6±1.7E6
		cor-nsga-ii	137E-3±10E-6	<b>12.3E-3±0E0</b>	<b>1.1E0±305E-3</b>	<b>1.1E0±305E-3</b>	<b>7.94E6±296E3</b>
Smarthome	Restric.	r-nsga-ii	49E-3±650E-6	61.9E-3±200E-6	112E0±0E0	112E0±0E0	30.4E6±1.5E6
		cor-nsga-ii	<b>83.6E-3±420E-6</b>	<b>52.3E-3±90E-6</b>	<b>1.2E0±407E-3</b>	<b>1.17E0±379E-3</b>	<b>10.9E6±559E3</b>
	Comp.	r-nsga-ii	<b>139E-3±340E-6</b>	<b>9.69E-3±0E0</b>	112E0±0E0	112E0±0E0	30.4E6±1.34E6
		cor-nsga-ii	137E-3±10E-6	<b>9.69E-3±0E0</b>	<b>1.2E0±407E-3</b>	<b>1.17E0±379E-3</b>	<b>10.9E6±559E3</b>

Table G.4: COR-NSGA-II Versus R-NSGA-II in Scenario 3D

	RPAlgorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time	
James	Restric.	r-nsga-ii	46.4E-3±21.8E-3	78.3E-3±8.06E-3	110E0±8.78E0	88.4E0±33.5E0	15.6E3±1.27E3
		cor-nsga-ii	<b>120E-3±1.09E-3</b>	<b>57.7E-3±280E-6</b>	<b>67.2E0±2E0</b>	<b>11.1E0±1.66E0</b>	<b>4.45E3±511E0</b>
	Comp.	r-nsga-ii	65.3E-3±21.1E-3	65.6E-3±5.37E-3	112E0±740E-3	111E0±1.93E0	16.7E3±1.59E3
		cor-nsga-ii	<b>207E-3±4.85E-3</b>	<b>39.4E-3±690E-6</b>	<b>67.2E0±2E0</b>	<b>13.2E0±1.88E0</b>	<b>4.45E3±511E0</b>
CAS	Restric.	r-nsga-ii	14.1E-3±410E-6	92E-3±730E-6	109E0±1.27E0	95E0±14.9E0	120E3±5.38E3
		cor-nsga-ii	<b>126E-3±2.22E-3</b>	<b>53E-3±400E-6</b>	<b>96.4E0±2.98E0</b>	<b>12.4E0±1.45E0</b>	<b>9.39E3±2.16E3</b>
	Comp.	r-nsga-ii	43.5E-3±980E-6	57.3E-3±670E-6	109E0±1.67E0	98E0±14.6E0	121E3±6.95E3
		cor-nsga-ii	<b>182E-3±1.98E-3</b>	<b>33.8E-3±240E-6</b>	<b>96.4E0±2.98E0</b>	<b>12.5E0±1.66E0</b>	<b>9.39E3±2.16E3</b>
WS	Restric.	r-nsga-ii	14.1E-3±330E-6	81.1E-3±570E-6	110E0±999E-3	106E0±6.27E0	139E3±7.23E3
		cor-nsga-ii	<b>123E-3±990E-6</b>	<b>47.1E-3±200E-6</b>	<b>100E0±2.84E0</b>	<b>15.6E0±1.61E0</b>	<b>12.5E3±327E0</b>
	Comp.	r-nsga-ii	43.5E-3±1.05E-3	58.6E-3±710E-6	110E0±1.1E0	107E0±6.75E0	138E3±7.05E3
		cor-nsga-ii	<b>182E-3±1.99E-3</b>	<b>34.2E-3±180E-6</b>	<b>100E0±2.84E0</b>	<b>17.5E0±1.81E0</b>	<b>12.5E3±327E0</b>
E-Shop	Restric.	r-nsga-ii	13.7E-3±290E-6	79.4E-3±450E-6	112E0±183E-3	111E0±2.8E0	403E3±20E3
		cor-nsga-ii	<b>23.4E-3±4.46E-3</b>	<b>71.6E-3±2.75E-3</b>	<b>29.6E0±11.6E0</b>	<b>21.8E0±6.42E0</b>	<b>44.9E3±2.18E3</b>
	Comp.	r-nsga-ii	42.3E-3±820E-6	60.6E-3±610E-6	112E0±379E-3	112E0±379E-3	404E3±19.6E3
		cor-nsga-ii	<b>66.5E-3±10.8E-3</b>	<b>53E-3±2.78E-3</b>	<b>29.6E0±11.6E0</b>	<b>21.8E0±6.42E0</b>	<b>44.9E3±2.18E3</b>
Drupal	Restric.	r-nsga-ii	<b>12.7E-3±90E-6</b>	<b>114E-3±90E-6</b>	112E0±0E0	112E0±0E0	21E6±1.19E6
		cor-nsga-ii	12.3E-3±0E0	114E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>8.24E6±319E3</b>
	Comp.	r-nsga-ii	<b>39.8E-3±250E-6</b>	<b>22.5E-3±20E-6</b>	112E0±0E0	112E0±0E0	20.9E6±1.23E6
		cor-nsga-ii	38.9E-3±0E0	22.5E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>8.24E6±319E3</b>
Smarthome	Restric.	r-nsga-ii	<b>12.7E-3±70E-6</b>	<b>135E-3±100E-6</b>	112E0±0E0	112E0±0E0	30.4E6±1.33E6
		cor-nsga-ii	12.3E-3±0E0	135E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>11.5E6±911E3</b>
	Comp.	r-nsga-ii	<b>40E-3±290E-6</b>	<b>17.3E-3±10E-6</b>	112E0±0E0	112E0±0E0	30.4E6±1.29E6
		cor-nsga-ii	38.9E-3±0E0	17.3E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>11.5E6±911E3</b>



Table G.5: COR-NSGA-II Versus PCA-NSGA-II in Scenario 2D.

	RPAlgorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time	
James	Restric.	pca-nsga-ii	13.3E-3±20.1E-3	81.3E-3±9.31E-3	7.8E0±25.7E0	2.87E0±8.53E0	22.9E3±685E0
		cor-nsga-ii	<b>157E-3±310E-6</b>	<b>38.5E-3±40E-6</b>	<b>7.53E0±1.07E0</b>	3.47E0±629E-3	<b>3.55E3±670E0</b>
	Comp.	pca-nsga-ii	42.9E-3±37.7E-3	41.3E-3±5.87E-3	7.8E0±25.7E0	3.87E0±10.9E0	22.9E3±685E0
		cor-nsga-ii	<b>156E-3±2.91E-3</b>	<b>22.1E-3±70E-6</b>	<b>7.53E0±1.07E0</b>	<b>1.17E0±379E-3</b>	<b>3.55E3±670E0</b>
CAS	Restric.	pca-nsga-ii	56.7E-3±1.95E-3	58.9E-3±510E-6	112E0±434E-3	63.8E0±6.48E0	163E3±12.2E3
		cor-nsga-ii	<b>185E-3±490E-6</b>	<b>39.1E-3±50E-6</b>	<b>20.4E0±5.47E0</b>	<b>13.6E0±4.19E0</b>	<b>9.77E3±2.44E3</b>
	Comp.	pca-nsga-ii	145E-3±1.29E-3	10.3E-3±10E-6	112E0±434E-3	63.8E0±6.48E0	163E3±12.2E3
		cor-nsga-ii	<b>151E-3±1.38E-3</b>	<b>10E-3±40E-6</b>	<b>20.4E0±5.47E0</b>	<b>14.7E0±4.1E0</b>	<b>9.77E3±2.44E3</b>
WS	Restric.	pca-nsga-ii	56.2E-3±1.96E-3	67.5E-3±590E-6	112E0±0E0	69E0±5.75E0	191E3±23.5E3
		cor-nsga-ii	<b>189E-3±700E-6</b>	<b>44.5E-3±80E-6</b>	<b>20.5E0±3.74E0</b>	<b>12.5E0±2.34E0</b>	<b>12.4E3±501E0</b>
	Comp.	pca-nsga-ii	144E-3±1.12E-3	46.4E-3±350E-6	112E0±0E0	69E0±5.77E0	191E3±23.5E3
		cor-nsga-ii	<b>154E-3±600E-6</b>	<b>45.1E-3±120E-6</b>	<b>20.5E0±3.74E0</b>	<b>12.5E0±2.34E0</b>	<b>12.4E3±501E0</b>
E-Shop	Restric.	pca-nsga-ii	51.9E-3±840E-6	96.8E-3±370E-6	112E0±0E0	53.7E0±5.11E0	533E3±90E3
		cor-nsga-ii	<b>180E-3±1.09E-3</b>	<b>62.8E-3±150E-6</b>	<b>5.93E0±4.76E0</b>	<b>4.37E0±2.67E0</b>	<b>43.4E3±1.82E3</b>
	Comp.	pca-nsga-ii	<b>142E-3±670E-6</b>	<b>14.5E-3±20E-6</b>	112E0±0E0	53.7E0±5.11E0	533E3±90E3
		cor-nsga-ii	137E-3±840E-6	14.5E-3±60E-6	<b>5.93E0±4.76E0</b>	<b>4.37E0±2.67E0</b>	<b>43.4E3±1.82E3</b>
Drupal	Restric.	pca-nsga-ii	47.1E-3±300E-6	81.4E-3±130E-6	112E0±0E0	69.7E0±5.21E0	27.1E6±5.16E6
		cor-nsga-ii	<b>83.7E-3±410E-6</b>	<b>68E-3±110E-6</b>	<b>1.1E0±305E-3</b>	<b>1.1E0±305E-3</b>	<b>7.94E6±296E3</b>
	Comp.	pca-nsga-ii	<b>138E-3±190E-6</b>	<b>12.3E-3±0E0</b>	112E0±0E0	69.7E0±5.21E0	27.1E6±5.16E6
		cor-nsga-ii	137E-3±10E-6	<b>12.3E-3±0E0</b>	<b>1.1E0±305E-3</b>	<b>1.1E0±305E-3</b>	<b>7.94E6±296E3</b>
Smarthome	Restric.	pca-nsga-ii	47E-3±300E-6	62.5E-3±100E-6	112E0±0E0	79.2E0±4.98E0	41.8E6±6.81E6
		cor-nsga-ii	<b>83.6E-3±420E-6</b>	<b>52.3E-3±90E-6</b>	<b>1.2E0±407E-3</b>	<b>1.17E0±379E-3</b>	<b>10.9E6±559E3</b>
	Comp.	pca-nsga-ii	<b>138E-3±270E-6</b>	<b>9.69E-3±0E0</b>	112E0±0E0	79.2E0±4.98E0	41.8E6±6.81E6
		cor-nsga-ii	137E-3±10E-6	<b>9.69E-3±0E0</b>	<b>1.2E0±407E-3</b>	<b>1.17E0±379E-3</b>	<b>10.9E6±559E3</b>

Table G.6: COR-NSGA-II Versus PCA-NSGA-II in Scenario 3D.

	RPAlgorithm	R-HV	R-IGD	# of Solutions	# of Solutions in the ROI	Execution Time	
James	Restric.	pca-nsga-ii	13.1E-3±2.61E-3	98.5E-3±2.09E-3	<b>11.3E0±31.5E0</b>	<b>6.1E0±15.9E0</b>	22.2E3±1.99E3
		cor-nsga-ii	<b>120E-3±1.09E-3</b>	<b>57.7E-3±280E-6</b>	67.2E0±2E0	11.1E0±1.66E0	<b>4.45E3±511E0</b>
	Comp.	pca-nsga-ii	40.8E-3±6.51E-3	74.9E-3±2.3E-3	<b>11.3E0±31.5E0</b>	<b>6.1E0±15.9E0</b>	22.2E3±1.99E3
		cor-nsga-ii	<b>207E-3±4.85E-3</b>	39.4E-3±690E-6	67.2E0±2E0	13.2E0±1.88E0	<b>4.45E3±511E0</b>
CAS	Restric.	pca-nsga-ii	13.5E-3±220E-6	92.4E-3±390E-6	112E0±346E-3	65.5E0±4.78E0	165E3±12.3E3
		cor-nsga-ii	<b>126E-3±2.22E-3</b>	<b>53E-3±400E-6</b>	<b>96.4E0±2.98E0</b>	<b>12.4E0±1.45E0</b>	<b>9.39E3±2.16E3</b>
	Comp.	pca-nsga-ii	42E-3±640E-6	57.7E-3±410E-6	112E0±346E-3	65.5E0±4.78E0	165E3±12.3E3
		cor-nsga-ii	<b>182E-3±1.98E-3</b>	<b>33.8E-3±240E-6</b>	<b>96.4E0±2.98E0</b>	<b>12.5E0±1.66E0</b>	<b>9.39E3±2.16E3</b>
WS	Restric.	pca-nsga-ii	13.5E-3±180E-6	81.5E-3±260E-6	112E0±0E0	68.5E0±5.02E0	189E3±21.5E3
		cor-nsga-ii	<b>123E-3±990E-6</b>	<b>47.1E-3±200E-6</b>	<b>100E0±2.84E0</b>	<b>15.6E0±1.61E0</b>	<b>12.5E3±327E0</b>
	Comp.	pca-nsga-ii	41.9E-3±510E-6	59E-3±320E-6	112E0±0E0	68.5E0±5.02E0	189E3±21.5E3
		cor-nsga-ii	<b>182E-3±1.99E-3</b>	<b>34.2E-3±180E-6</b>	<b>100E0±2.84E0</b>	<b>17.5E0±1.81E0</b>	<b>12.5E3±327E0</b>
E-Shop	Restric.	pca-nsga-ii	13E-3±130E-6	79.7E-3±180E-6	112E0±0E0	56E0±5.95E0	540E3±92.4E3
		cor-nsga-ii	<b>23.4E-3±4.46E-3</b>	<b>71.6E-3±2.75E-3</b>	<b>29.6E0±11.6E0</b>	<b>21.8E0±6.42E0</b>	<b>44.9E3±2.18E3</b>
	Comp.	pca-nsga-ii	40.8E-3±370E-6	61E-3±240E-6	112E0±0E0	56E0±5.95E0	540E3±92.4E3
		cor-nsga-ii	<b>66.5E-3±10.8E-3</b>	<b>53E-3±2.78E-3</b>	<b>29.6E0±11.6E0</b>	<b>21.8E0±6.42E0</b>	<b>44.9E3±2.18E3</b>
Drupal	Restric.	pca-nsga-ii	<b>12.5E-3±40E-6</b>	<b>114E-3±40E-6</b>	112E0±0E0	70.3E0±6.07E0	28.4E6±4.99E6
		cor-nsga-ii	12.3E-3±0E0	114E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>8.24E6±319E3</b>
	Comp.	pca-nsga-ii	<b>39.5E-3±120E-6</b>	<b>22.5E-3±10E-6</b>	112E0±0E0	70.3E0±6.07E0	28.4E6±4.99E6
		cor-nsga-ii	38.9E-3±0E0	22.5E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>8.24E6±319E3</b>
Smarthome	Restric.	pca-nsga-ii	<b>12.6E-3±50E-6</b>	<b>135E-3±50E-6</b>	112E0±0E0	79.3E0±5.11E0	41.5E6±6.22E6
		cor-nsga-ii	12.3E-3±0E0	135E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>11.5E6±911E3</b>
	Comp.	pca-nsga-ii	<b>39.5E-3±130E-6</b>	<b>17.3E-3±10E-6</b>	112E0±0E0	79.3E0±5.11E0	41.5E6±6.22E6
		cor-nsga-ii	38.9E-3±0E0	17.3E-3±0E0	<b>1E0±0E0</b>	<b>1E0±0E0</b>	<b>11.5E6±911E3</b>

## APPENDIX H – RQ3 DETAILED RESULTS

Table H.1: Preferred and final subset of objectives for each participant.

<b>Participant</b>	<b>Preferred Objectives</b>	<b>Final Subset of Objectives</b>
#1	Number of Products, Cost	Number of Products, Similarity, Cost
#2	Number of Products, Cost	Number of Products, Cost
#3	Number of Products, Similarity, Cost	Number of Products, Similarity, Cost
#4	Cost, Unimportant Features	Cost, Unimportant Features
#5	Number of Products, Similarity, Cost, Unselected Features	Number of Products, Similarity, Cost, Unselected Features
#6	Number of Products, Similarity, Cost, Unselected Features	Number of Products, Cost, Unselected Features
#7	Number of Products, Similarity, Cost, Unimportant Features	Number of Products, Cost, Unimportant Features
#8	Alive Mutants, Similarity, Cost, Unselected Features	Alive Mutants, Uncovered Pairs, Similarity, Cost, Unselected Features
#9	Number of Products, Alive Mutants, Cost, Unimportant Features	Number of Products, Alive Mutants, Similarity, Cost, Unimportant Features
#10	Number of Products, Alive Mutants, Similarity, Cost	Number of Products, Alive Mutants, Uncovered Pairs, Similarity, Cost, Unimportant Features
#11	Number of Products, Alive Mutants, Uncovered Pairs, Unimportant Features	Number of Products, Alive Mutants, Uncovered Pairs, Similarity, Unimportant Features
#12	Number of Products, Alive Mutants, Uncovered Pairs, Cost	Number of Products, Alive Mutants, Uncovered Pairs, Cost