

2014

Modeling And Applying Biomimetic Metaheuristics To Product Life Cycle Engineering

Patrick TchaptiÃ© Wanko
North Carolina Agricultural and Technical State University

Follow this and additional works at: <https://digital.library.ncat.edu/dissertations>

Recommended Citation

Wanko, Patrick TchaptiÃ©, "Modeling And Applying Biomimetic Metaheuristics To Product Life Cycle Engineering" (2014). *Dissertations*. 71.
<https://digital.library.ncat.edu/dissertations/71>

This Dissertation is brought to you for free and open access by the Electronic Theses and Dissertations at Aggie Digital Collections and Scholarship. It has been accepted for inclusion in Dissertations by an authorized administrator of Aggie Digital Collections and Scholarship. For more information, please contact iyanna@ncat.edu.

Modeling and Applying Biomimetic Metaheuristics to Product Life Cycle Engineering

Patrick Tchaptié Wanko

North Carolina A&T State University

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department: Industrial and Systems Engineering

Major: Industrial and Systems Engineering

Major Professor: Dr. Paul M. Stanfield

Greensboro, North Carolina

2014

The Graduate School
North Carolina Agricultural and Technical State University
This is to certify that the Doctoral Dissertation of

Patrick Tchapdié Wanko

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2014

Approved by:

Dr. Paul M. Stanfield
Major Professor

Dr. Lauren B. Davis
Committee Member

Dr. Abdollah Homaifar
Committee Member

Dr. Xiuli Qu
Committee Member

Dr. Tonya Smith-Jackson
Department Chair

Dr. Sanjiv Sarin
Dean, The Graduate School

© Copyright by
Patrick Tchapié Wanko
2014

Biographical Sketch

Patrick Tchaptié Wanko was born on April 5, 1981, in Garoua, Cameroon, to Joseph Wanko and Jeanne Tchanchou. He received a Bachelor of Science degree in Computer Engineering from the United States Air Force Academy in 2007 and returned to Cameroon where he served as a second lieutenant in the Cameroonian Air Force. In January 2008, he returned to the United States to further his education at North Carolina Agricultural and Technical State University. In December 2009, he received a Master of Sciences degree in Electrical Engineering, option Computer Engineering. He decided to shift his concentration to Industrial and Systems Engineering for a doctoral degree. He currently is a doctorate candidate in the Department of Industrial and Systems Engineering of North Carolina Agricultural and Technical State University, where he is specializing on the application of biomimetics to product design within the operational research field.

Dedication

I dedicate this dissertation to my parents. Without their understanding, support, and most of all love, the completion of this work would not have been possible. I also dedicate this dissertation to my beautiful and loving wife; Carole Lise Waguem Kouam. She has stood by me and has given me her unconditional support and devoted motivation from day one of my doctoral journey. Without her support and understanding, I would have not been able to complete my work in due time.

Acknowledgements

Special thanks are given to my main advisor: Dr. Paul Stanfield, and to the rest of my committee members; Dr. Lauren Davis, Dr. Abdollah Homaifar, and Dr. Xiuli Qu, for their contributions to this dissertation and to my overall learning experiences at North Carolina Agricultural and Technical State University.

This research was based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this work are those of the author and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

List of Figures	xii
List of Tables	xvi
List of Abbreviations	xviii
Abstract	2
CHAPTER 1 Introduction.....	3
1.1 Product Life cycle Engineering.....	4
1.2 Durable Product Evolution.....	5
1.3 Motivation of Research.....	6
1.4 Objectives of Research	8
1.5 Research Contribution	9
1.6 Dissertation Overview	10
1.7 Summary	11
CHAPTER 2 Overview of Product Design and Biomimetics	12
2.1 Product Design.....	12
2.1.1 Integrated product and process development.....	18
2.1.2 Deductive product development approaches.	20
2.1.2.1 Design for operational feasibility.....	21
2.1.2.2 User-behavior based.....	23
2.1.2.3 Bio-inspired product design methodology.....	24
2.2 Biomimetics	24

2.2.1 Biomimetics a computing tool.	24
2.2.1.1 Ant colony optimization.....	25
2.2.1.2 Particle swarm optimization.....	26
2.2.1.3 Genetic algorithms.	27
2.2.1.4 Schooling genetic algorithms.....	28
2.2.2 Biomimetic as a conceptual framework.	28
2.2.2.1 Bio-inspired design.	28
2.2.2.2 Life cycle assessment.	29
2.3 Summary	30
CHAPTER 3 Product Design.....	32
3.1 Current Product Design Limits	33
3.2 Life Cycle Parameters.....	36
3.2.1 Design parameters.....	41
3.2.1.1 Material.	41
3.2.1.2 Functions.....	41
3.2.1.3 Sensor.....	41
3.2.2 Operational parameters.	42
3.2.2.1 Type of use.....	42
3.2.2.2 Frequency of use.	42
3.2.2.3 Type of maintenance.....	43
3.2.3 Environmental parameters.	43

3.2.3.1 Physical environment	43
3.2.3.2 Alternative products	44
3.2.3.3 Culture.....	44
3.3 Sustainable Performance.....	44
3.4 Generalized Life cycle Product Design	47
3.5 Summary	52
CHAPTER 4 Schooling Genetic Algorithms.....	53
4.1 Introduction.....	53
4.2 Parallel Genetic Algorithms.....	55
4.3 SGA Overview.....	57
4.3.1 Fish school.	57
4.3.2 Terminology and taxonomies.....	58
4.4 SGA Procedure	60
4.5 SGA Modeling.....	61
4.5.1 School merging and splitting.	61
4.5.1.1 Computational aspect of GEMAC.	63
4.5.1.2 Clustering in action with GEMAC.....	64
4.5.2 Behavior setup.	67
4.5.3 Predator avoidance.....	68
4.5.4 Food foraging.....	70
4.5.5 School maintenance.	72

4.5.6 SGA Life Cycle.....	73
4.6 Summary	74
Chapter 5 Applying Schooling Genetic Algorithms to Generalized Life cycle Product Design	75
5.1 Introduction.....	75
5.2 Problem Definition.....	75
5.3 Applying SGA to GLPD.....	77
5.4 Experimental Design.....	79
5.4.1 Environment driving design for performance.	81
5.4.2 Environment driving both design and operations for performance.....	81
5.4.3 Environment and design both driving operations for performance.....	81
5.4.4 Environment and the design both driving performance.	82
5.4.5 Environment, design and operations driving performance.	84
5.5 Results and Interpretation	85
5.5.1 Environment driving design for performance.	86
5.5.2 Environment driving both design and operations for performance.....	90
5.5.3 Environment and design both driving operations for performance.....	93
5.5.4 Environment and the design both driving performance.	96
5.5.4.1 Using Griewank to characterize LCE's relationship.....	96
5.5.4.2 Using Schwefel to characterize LCE's relationship.....	100
5.5.5 Environment, design and operations all driving performance.	102
5.5.5.1 Using Ackley to characterize LCE's relationship.	103

5.5.5.2 Using Schwefel to characterize LCE's relationship.....	108
5.6 Summary	114
CHAPTER 6 Genetic Social Networks	117
6.1 Introduction.....	117
6.2 GSN Overview.....	118
6.2.1 Gravitational pull.	122
6.2.2 Proportionate breeding.....	122
6.3 GSN Procedure	123
6.4 GSN Modeling.....	125
6.4.1 Joining and leaving groups.	125
6.4.2 Fuzzy membership.	126
6.5 Summary	128
Chapter 7 Applying Genetic Social Networks to Generalized life cycle product design	129
7.1 Introduction.....	129
7.2 Applying GSN to GLPD.....	129
7.3 GSN Approach to the Problem	130
7.4 Experimental Design.....	131
7.4.1 Environment driving design for performance.	133
7.4.2 Environment driving both design and operations for performance.....	133
7.4.3 Environment and design both driving operations for performance.....	133
7.4.4 Environment and design both driving performance.....	133

7.4.5 Environment, design and operations all driving performance.	135
7.5 Results and Interpretation	136
7.5.1 Environment driving design for performance.	137
7.5.2 Environment driving both design and operations for performance.....	140
7.5.3 Environment and design both driving operations for performance.....	141
7.5.4 Environment and the design both driving performance.	143
7.5.4.1 Using Griewank to characterize LCE's relationship.....	143
7.5.4.2 Using Schwefel to characterize LCE's relationship.....	145
7.5.5 Environment, design and operations all driving performance.	146
7.5.5.1 Using Ackley to characterize LCE's relationship.	146
7.5.5.2 Using Schwefel to characterize LCE's relationship.....	151
7.6 Summary	155
Chapter 8 Conclusion.....	158
8.1 Contributions.....	159
8.2 Future Directions	159
References.....	161
Appendix A.....	178
Appendix B	193
Appendix C.....	205
Appendix D.....	210
Appendix E	215

List of Figures

Figure 1.1. AIT technologies in modern product parts	5
Figure 1.2. Product life cycle information and material flow (Hong-Bae Jun, Dimitris Kiritsis, & Xirouchakis, 2007).....	8
Figure 2.1. Model of reasoning by designers. (Roozenburg & Eekels, 1995).....	13
Figure 2.2. LCE methodologies: (a) Waterfall model (Horner, 1993), and (b) Spiral model (Boehm, 1986)	15
Figure 2.3. LCE methodologies: (a) IPPD model (DEFENSE, 1996), and (b) Dual Vee Model (Kevin Forsberg & Mooz, 1997)	15
Figure 2.4. Integrate product design process (Hock, 1997).....	19
Figure 2.5. Common tasks in a manufacturing firm and relevant biological analogies(Mill & Sherlock, 2000).....	25
Figure 3.1. Product design factors	32
Figure 3.2. Systemic view of a product	37
Figure 3.3. Impact of parameters on product performance.....	38
Figure 3.4. Collaborative life cycle.....	40
Figure 3.5. System Operational Effectiveness (SOE) (Verma & Gallois, 2001)	47
Figure 3.6. Generalized life cycle product design (GLPD) approach.....	49
Figure 3.7. Grouping vs. performance	51
Figure 4.1. Food vs. Predator.....	58
Figure 4.2. Taxonomy of search techniques	59
Figure 4.3. Taxonomy of parameter setting in evolutionary algorithms (Michalewicz & Fogel, 2004).....	59

Figure 4.4. Genetic Algorithms vs. Schooling Genetic Algorithms (* often omitted).....	61
Figure 4.5. Domains with predetermined cluster centers	64
Figure 4.6. GEMAC output for Test 1	65
Figure 4.7. GEMAC output for Test 2.....	65
Figure 4.8. GEMAC output for Test 3.....	66
Figure 4.9. Fish school behavior assignment.....	67
Figure 4.10. Predator avoidance maneuver.....	68
Figure 5.1. Transformation of a designed product into a GA entity.....	76
Figure 5.2. Griewank function in [-100, 100] plotted using Matlab.....	83
Figure 5.3. Schwefel function in [-100, 100] plotted using Matlab.....	84
Figure 5.4. Ackley's function in [-25, 25] plotted using Matlab.....	85
Figure 5.5. Tabu list contents plot for experiment set 1	87
Figure 5.6. SGA results per generation for experiment 1	89
Figure 5.7. SGA results per generation for experiment set 2.....	91
Figure 5.8. Tabu list contents plot for experiment set 2	93
Figure 5.9. SGA results per generation for experiment set 3.....	94
Figure 5.10. Tabu list contents plot for experiment set 3	96
Figure 5.11. SGA results per generation for experiment set 4 – Griewank.....	97
Figure 5.12. Tabu list contents plot for experiment set 4 – Griewank.....	99
Figure 5.13. SGA results per generation for experiment set 4 – Schwefel.....	100
Figure 5.14. Tabu list contents plot for experiment set 4 – Schwefel	102
Figure 5.15. SGA results per generation for experiment set 5 – Ackley	103
Figure 5.16. Average population fitness over time with PGA for experiment set 5 – Ackley...	105

Figure 5.17. Average population fitness over time with IGA for experiment set 5 – Ackley	107
Figure 5.18. SGA results per generation for experiment set 5 – Schwefel.....	109
Figure 5.19. Average population fitness over time with PGA for experiment set 5 – Schwefel	111
Figure 5.20. Average population fitness over time with IGA for experiment set 5 – Schwefel.	113
Figure 6.1. Genetic social networking	121
Figure 6.2. GSN high level diagram	124
Figure 7.1. Transformation of a designed product into a GA entity.....	130
Figure 7.2. Griewank function in [-100, 100] plotted using Matlab.....	135
Figure 7.3. Schwefel function in [-100, 100] plotted using Matlab.....	136
Figure 7.4. Ackley's function in [-25, 25] plotted using Matlab.....	136
Figure 7.5. GSN results per generation for experiment set 1.....	137
Figure 7.6. Final population plot for experiment set 1	140
Figure 7.7. GSN results per generation for experiment set 2.....	140
Figure 7.8. Final population plot for experiment set 2	141
Figure 7.9. GSN results per generation for experiment set 3.....	142
Figure 7.10. Final population plot for experiment set 3	143
Figure 7.11. GSN results per generation for experiment set 4 – Griewank.....	144
Figure 7.12. Final population plot for experiment set 4 – Griewank.....	145
Figure 7.13. SGA results per generation for experiment set 4 – Schwefel.....	145
Figure 7.14. Final population plot for experiment set 4 – Schwefel.....	146
Figure 7.15. GSN results per generation for experiment set 5 – Ackley	147
Figure 7.16. Average population fitness over time with PGA for experiment set 5 – Ackley ...	149
Figure 7.17. Average population fitness over time with IGA for experiment set 5 – Ackley	151

Figure 7.18. GSN results per generation for experiment set 5 – Schwefel.....	151
Figure 7.19. Average population fitness over time with PGA for experiment set 5 – Schwefel	153
Figure 7.20. Average population fitness over time with IGA for experiment set 5 – Schwefel.	155

List of Tables

Table 1.1 Sikorsky 70 models in the U.S. military	6
Table 1.2 Research contribution	10
Table 2.1 System engineering and life cycle engineering resources	14
Table 3.1 Samples for life cycle parameters	39
Table 3.2 Classifying LCE data	40
Table 3.3 GLPD process elements.....	48
Table 4.1 GEMAC clustering output summary	66
Table 5.1 Factors, assessment criteria, and methods for SGA.....	80
Table 5.2 Sample of initial population for experiment 1	86
Table 5.3 Tabu list contents for experiment 1	88
Table 5.4 Sample of final population for experiment 1	89
Table 5.5 Sample of initial population for experiment 2	90
Table 5.6 Sample of final population.....	91
Table 5.7 Tabu list contents for experiment 2	92
Table 5.8 Tabu list contents for experiment set 3.....	95
Table 5.9 Tabu list contents for experiment set 4 – Griewank	98
Table 5.10 Tabu list contents for experiment set 4 – Schwefel.....	101
Table 5.11 Tabu list contents for experiment set 5 – Ackley	104
Table 5.12 Sample of final population for PGA experiment set 5 – Ackley	106
Table 5.13 Sample of final population for IGA experiment set 5 – Ackley	108
Table 5.14 Tabu list contents for experiment set 5 – Schwefel	110
Table 5.15 Sample of final population for SGA in experiment set 5 – Schwefel.....	111

Table 5.16 Sample of final population for PGA experiment set 5 – Schwefel.....	112
Table 5.17 Sample of final population for IGA experiment set 5 – Schwefel.....	114
Table 5.18 Summary of experiments on SGA.....	116
Table 7.1 Factors, assessment criteria, and methods	132
Table 7.2 Sample of final population of GSN for experiment set 1	138
Table 7.3 Starting and ending themes in experiment set 1	139
Table 7.4 Sample of final population for GSN in experiment set 2	141
Table 7.5 Sample of final population for SGN in experiment set 3	142
Table 7.6 Sample of final population for SGN in experiment set 4– Griewank.....	144
Table 7.7 Sample of final population for GSN experiment set 5 – Ackley	147
Table 7.8 Sample of final population for PGA experiment set 5 – Ackley	148
Table 7.9 Sample of final population for IGA experiment set 5 – Ackley	150
Table 7.10 Sample of final population for GSN experiment set 5 – Schwefel.....	152
Table 7.11 Sample of final population for PGA experiment set 5 – Schwefel.....	153
Table 7.12 Sample of final population for IGA experiment set 5 – Schwefel.....	154
Table 7.13 Summary of experiments on GSN.....	156
Table 7.14 Comparative results of GSN, SGA, IGA, and PGA	157

List of Abbreviations

ACO	Ant Colony Optimization
AIT	Automated Identification Technology
BOL	Beginning Of Life
EOL	End Of Life
ERP	Enterprise Resource Planning
GA	Genetic Algorithms
GLPD	Generalized Life cycle Product Design
GSN	Genetic Social Network
IGA	Island Genetic Algorithm
INCOSE	International Council of Systems Engineering
IPPD	Integrated Product and Process Design
LCE	Life Cycle Engineering
MOL	Middle Of Life
PLE	Product Life cycle Engineering
PSO	Particle Swarm Optimization
SE	System Engineering
SGA	Schooling Genetic Algorithm
SIAIT	Sensor-Integrated Automatic Identification Technology

Abstract

Due to its potential for significant impact, interest continues to grow in the assessment of products from a life cycle perspective. As the nature of products shifts from mechanized and Newtonian to more adaptive and complex, the behavior of products more closely resembles biological organisms in community. The change in product nature is increasingly mirrored at the component level. The work presented in this dissertation is twofold. First, the research proposes a general, systematic and holistic classification of life cycle data to transform the design problem into an optimization problem. Second, the research proposes two new metaheuristics (bio-inspired and socio-inspired) to solve optimization problems to produce grouped solutions that are efficient, evolvable and sustainable. The bio-inspired approach is schooling genetic algorithms (SGA), while the socio-inspired approach is referred to as genetic social networks (GSN). SGA is an approach that combines fish schooling concepts with genetic algorithms (GAs) to enable a dynamic search process. The application of GA operators is subject to the perception of the immediate local environment by clusters of candidate solutions behaving as schools of fish. GSN is an approach that adds social network concepts to GAs, implementing single and dyadic social interactions of social groups (clusters of similar candidate solutions) with GA operators. SGA and GSN both use phenotypic representations of a hypothetical product or system as input. The representations are derived from the proposed life cycle engineering (LCE) data classification. The outputs of either method are the representations that are more than likely to perform better, longer, and more autonomously within their environment during their life cycle. Both methods can also be used as a decision making tool. Both approaches were tested on product design problems with differing parametric relations, underlying solution space, and problem size.

CHAPTER 1

Introduction

Systems engineering is defined as an “interdisciplinary approach and means to enable the realization of large and complex systems that meet a defined set of organizational and technical requirements” (INCOSE, 2006). Systems engineering (SE) as a scientific approach has been around since the 1940s and has evolved significantly from its prior engineering approaches. SE development post WW II was driven by U.S aerospace and defense industries, which formulated SE theory and best practices. Today, many techniques developed by those pioneering industries (e.g. parts traceability, materials and process control, improved product accountability) are being applied in other industries. In many ways, this field is mature. However, with the incorporation of information technology (IT) in ordinary products to create smart systems, the methods and tools that have made traditional SE successful are in need of improvement.

Traditionally, SE has emphasized: (1) design optimization into a fixed configuration, (2) system decomposition in order to facilitate system analysis, and (3) the guiding role of systems engineers to design and maintain systems. Such an emphasis does not account for products and/or product parts that are getting smarter, and tend to make SE heavily rely on the design engineer’s knowledge and expertise. These limitations, and the increasingly shortened life cycle of products (Griffin, 1997b) make it difficult for engineers to innovate and to sustain their design. With products becoming more complex and resembling biological entities (sense, process and act depending on environment), tools and approaches are necessary that will allow engineers in general, and design engineers in particular, to achieve system efficiencies. The work presented here is an attempt at crafting such an approach and associated tools. The research

provides a holistic approach, and relies both on the data gathered during a product's life cycle, and on the evolution of durable products.

1.1 Product Life cycle Engineering

SE is interdisciplinary and proceeds from concept to production and to operation by considering both the business and the technical needs with the goal of providing a quality product that meets the user needs at a low cost. SE integrates life cycle data, and has the same objectives as product life cycle engineering (PLE). PLE is a holistic business concept that was developed in the late 1980's to manage a product throughout its life cycle. PLE is the activity of managing, in the most effective way, a company's products across their life cycles from product concept to retirement and disposal (Stark, 2011). PLE allows any organization to oversee the whole lifespan of a product and the information connected with it (Sääksvuori & Immonen, 2008). To achieve its goal, PLE has become a central approach for the integrated management of product related data, engineering processes, and applications along the different phases of the product life cycle. PLE enables an organization to learn from its customers, analyze challenges and constraints, forecast changes in the development of a product or process, and make decisions based on the changes. PLE evolves with the product, its associated processes and its targeted market.

The remainder of the chapter is organized as follow: Section 1.2 addresses durable product evolution followed by the motivations of the research work in Section 1.3. The objectives of the research and the research contribution follow in Section 1.4 and 1.5 respectively. Finally, an overview of the remainder of the dissertation is given in Section 1.6, followed by a summary of the chapter.

1.2 Durable Product Evolution

Increasingly, the design of durable products, such as automobiles and aircraft, has expanded from traditional mechanical design to include more biologically inspired capabilities - learn, morph, communicate, and sustain. The trend of using analogies to biological systems to develop solutions for engineering problems, also called biologically inspired design, is somewhat new and keeps gaining importance as a wide-spread movement in design (Anastas & Warner, 2000; Benyus, 1997). Biologically inspired design often results in innovation (Collins & Brebbia, 2004; Forbes, 2005; Vogel, 2000). The timeline of the growth of biologically inspired design patents is described by Bosner (Bosner, 2006; Bosner & Vincent, 2006). The transition to biologically inspired design is making its way to high value assemblies and parts on such products. These changes have resulted in terms such as “evolving parts/products families” (ElMaraghy, 2007; Wiendahl et al., 2007) to address and describe the changes occurring to those product families as mutations, with product features losses and gains through generations, and the appearance of new families of products. The transition is enabled by Sensor-Integrated Automatic Identification Technology (SIAIT), which can provide data collection, storage, processing, and communication capabilities with minimal power requirements as depicted in Figure 1.1.

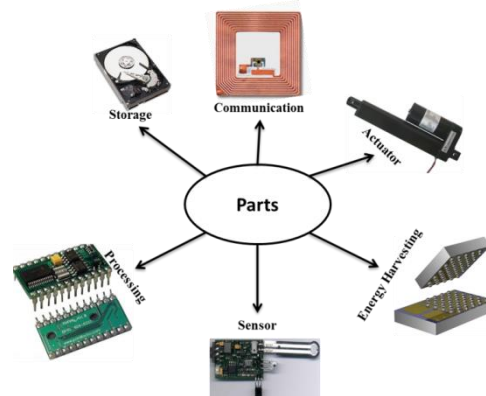


Figure 1.1. AIT technologies in modern product parts




The intelligent use of these enhanced capabilities depends primarily on the development of integrated processes. Processes that are needed to use the collected data to improve part/product design and operating parameters in order to minimize total cost of ownership, extend product life cycles, and enhance sustainability. As an example, the DoD alone spends US\$10s billions each year on these issues (AT&L, 2012). Due to the biological nature of the parts, bio/eco systems are expected to be the primary sources of process innovation.

1.3 Motivation of Research

Consider the Sikorsky 70 helicopter (Table 1.1). The U.S Army, the U.S Coast Guard, and the U.S Navy all use different variations of the same helicopter model. The variation the U.S Army uses is known as Black Hawk and operates in a typically arid environment; whereas the variation the U.S Coast Guard uses, the Jayhawk, operates in a damp environment and was designed to better accommodate its type of missions.

Table 1.1

Sikorsky 70 models in the U.S. military

	Army	USCG	Navy
	UH-60 Black Hawk	HH-60 Jayhawk	SH-60 Seahawk
Model			
Missions	Combat Search and Rescue, Special Forces operation...	Security and interdiction, offshore rescue...	Search and rescue, vertical replenishment, logistics support...
Environment	Land (Desert, Sahel)	Water	Sea, Land

Such a variation in the environment makes the annual acquisition and maintenance of the Sikorsky 70 inventory costly and difficult. In addition, the complexity makes it hard to know where to standardize designs and operational processes for the operational efficiency of the helicopter. The complexity also makes it difficult to detect where to customize a specific helicopter model to meet its mission objectives.

Today, organizations use a segmented LCE approach to remain competitive and innovative while managing the life cycle of their diverse portfolio. Firms would customize their tools so they can better integrate them with their different processes and at the same time, streamline the flow of material and information. However, the customization, segmentation of LCE phases, and integration of tools do not only come at a high cost (Jardim-Goncalves, Grilo, & Steiger-Garcao, 2006; Lin, Harding, & Shahbaz, 2004), but also fail (in its current state) to address one of the main issues the systems engineers still face. The main problem encountered is that the decisions taken during the beginning of life (BOL), which comprises conception, design and production are fixed and infrequently change; yet they are known to have a huge impact on middle of life (MOL) and end of life (EOL) decisions. The MOL stage of a product includes product's sale, operation, support and sustainment; whereas the EOL stage includes product's retirement for disposal or recycling. The data that is used in a segmented fashion could provide the good results for its segment, but not necessarily for the life cycle system. Information and material flow in a typical product life cycle implementation is represented in Figure 1.2. Figure 1.2 does not account for pieces of information such as consumers/users gained experience through recurrent product usage, or of the possible interactions among a life cycle BOL, MOL, and EOL.

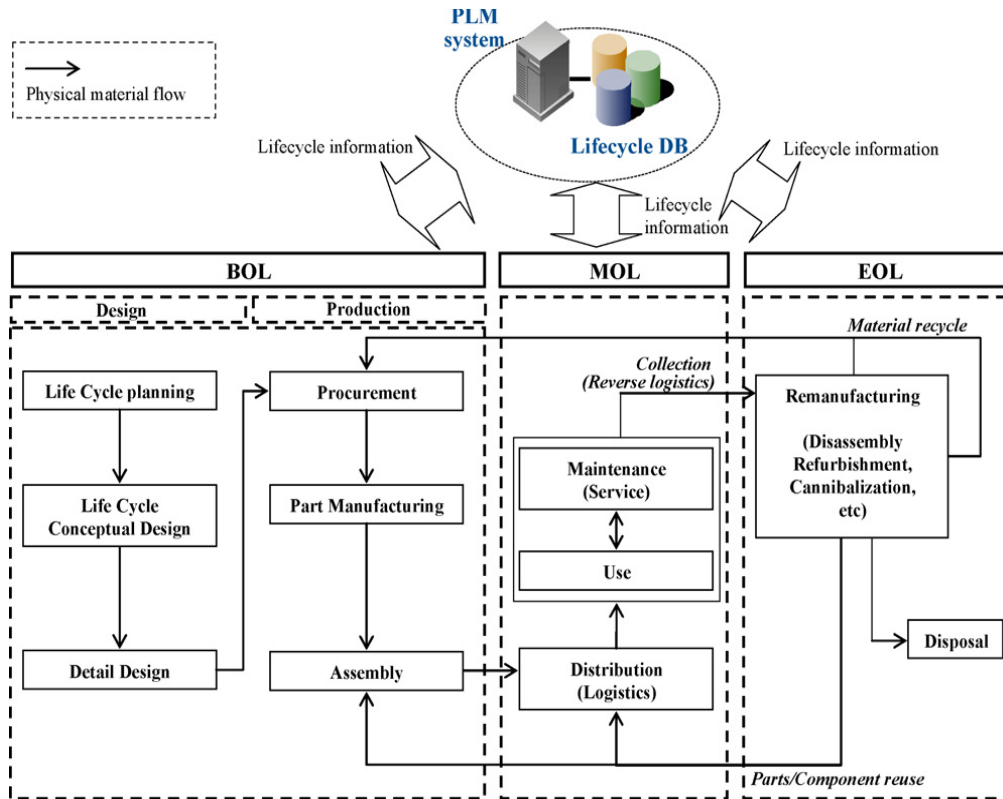


Figure 1.2. Product life cycle information and material flow (Hong-Bae Jun, Dimitris Kiritsis, & Xirouchakis, 2007)

1.4 Objectives of Research

Working within the context of product life cycle management, the objective of this research was to develop a framework that allows capturing the complex changes occurring in products and their attributes during their life cycle. This representation is an important first step towards their integration and the effective management of life cycle product evolution.

Considering the nature of the problems described within the previous sections, and the fact that there is no existing exact method to approach them, metaheuristics are suggested as a basis for the research. The research problem addressed the following problem. *Based on the shift in product nature, how does one characterize and extend PLE, using biological and sociological inspiration, to incorporate evolvability (evolution in design and operational parameters of*

products and product's parts), grouping (based on environmental parameters related to a product), and sustainability (ability of the system to maintain and improve itself)? In other words, how does one develop metaheuristic search algorithms, with the emphasis here on evolutionary approaches, using biological/sociological inspiration and grouping to design processes that can use a product's collected life cycle data to maintain and improve it totally (whole product) or partially (parts of the product) in a way that minimizes costs, and human/expert intervention?

Regarding the other part of the research, which is the development of a PLE-data based methodology for continuous sustainable product design, there is another research question. The research question is to find out whether and how metaheuristic search algorithms can be used to iterate through product life cycle data and find meaningful patterns to help engineers within an organization design better products for their users. The goal is to make available to the systems/design engineers the knowledge captured from the products' interactions with both the users, and the environment.

1.5 Research Contribution

The intellectual contribution of the research presented here falls in two categories associated with modeling product life cycle. First, a general characterization of life cycle data was made. The characterization was then used to develop a new, generic, and iterative approach for life cycle based product development. The new approach is called generalized life cycle product design (GLPD). Next, two new metaheuristic tools were developed, implemented and tested as metaheuristic tools applicable to both product design, via GLPD, and general stochastic optimization. The developed tools are a bio-inspired approach known as schooling genetic algorithms (SGA), and a socio-inspired approach known as genetic social network (GSN). Both

tools solve problems by not only looking for solutions that can perform better, but also by looking for solutions that have groupings, evolvability, and sustainability characteristics. The intellectual contribution is summarized in Table 1.2

Table 1.2

Research contribution

	Modeling	Product Life cycle
		Characterization of LCE data Generalized Life cycle Product Development (GLPD) Chapter 3
Biology	Schooling Genetic Algorithms (SGA) Chapter 4	SGA in GLPD Chapter 5
Sociology	Genetic Social Network (GSN) Chapter 6	GSN in GLPD Chapter 7

1.6 Dissertation Overview

The dissertation is comprised of eight chapters. Chapter 2 contains the literature review of biomimetics and life cycle engineering. Chapter 3 describes a new suggested PLE-data based representation for continuous sustainable product design that is consistent with life cycle principles. Chapter 4 introduces SGA in terms of concepts, parameters and implementation. Chapter 5 is about using PLE-data to apply SGA to product design. Chapter 6 introduces GSN. Chapter 7 is about the application of GSN to product design using PLE-data. Finally, Chapter 8 concludes the dissertation and discusses possible future work.

1.7 Summary

Within this chapter, the dissertation topic was introduced and PLE defined. The progression of durable product was explained and the motivations of the research work were given. The objectives of the research work were then explained, followed by the intellectual contribution. Finally, a complete overview of the dissertation, chapter by chapter was given.

CHAPTER 2

Overview of Product Design and Biomimetics

A literature review of product design and biomimetics is performed in this chapter. LCE methodologies for product design are reviewed as well. Some of the gaps, inherent to traditional SE, are identified. Uses of biomimetics in industry are also reviewed from a conceptual and computational perspective that fits within the traditional view of SE.

From an engineering standpoint, the design for durable goods consists of finding and defining the geometry and materials so the required prescribed physical behavior of that system is realized. Product design is the efficient and effective generation and development of ideas through a process that leads to new products (Morris, 2009).

Biomimetics on the other hand, also known as biomimicry is the examination of nature, its models, systems, processes, and elements to emulate or take inspiration in order to solve human problems ("The University of Reading: What is Biomimetics?," Retrieved June 5, 2012). Biomimetics is the abstraction of good design from nature (Low, 2009). This chapter covers both concepts.

2.1 Product Design

Usually embedded in a larger process called “product development” or “new business development”, the design of a product requires engineers reasoning from function to form and use. Figure 2.1 shows the model of reasoning by designers. This model of reasoning is based on induction (bottom-up reasoning) and is also known as synthesis. Despite the fact that companies are aware (Roozenburg & Eekels, 1995) of the necessity to learn to innovate effectively, and if possibly to overhaul their new product processes to incorporate ideas for successful new

products, Griffin (1997a) reported that almost 40% of firms surveyed still use no formalized product development process.

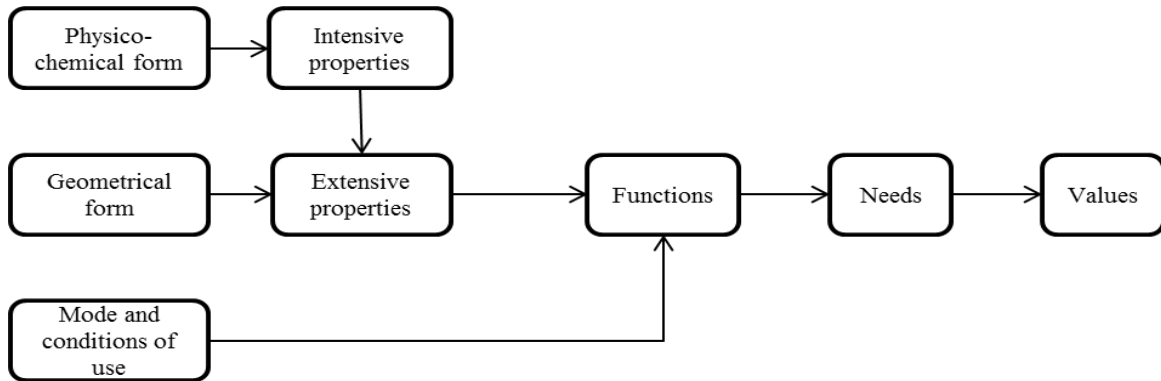


Figure 2.1. Model of reasoning by designers. (Roozenburg & Eekels, 1995)

The function and sustained performance of a product does not only depend on its properties (geometrical and physico-chemical form), but also on its environment, mode and conditions of use. However, one can reasonably say that product design stage decisions play the most important role in a product's performance during its entire life cycle.

Traditional SE is a mature field. Table 2.1 shows a brief summary of available resources on the topic of SE from commonly used academic resources. "Systems Engineering and Analysis" by Blanchard and Fabrycky, "Product Lifecycle Management" by Saaksvuori and Immonen, and "Product Lifecycle Management: 21st Century Paradigm for Product Realisation" by Stark, are well-known and often cited books in SE. The International Society for the Systems Sciences (ISSS), and the International Council Of Systems Engineering (INCOSE) are two professional organizations established in 1956 and chartered in 1991 respectively. Both organizations have been establishing guidelines, and are references in the field of systems engineering.

Table 2.1

System engineering and life cycle engineering resources

Literature on SE and LCE	Source
3,500,000+ Articles and (e)books	Google Scholar
2,400,000+ Articles and (e)books	Engineering Village
1,500,000+ Articles	Science Direct
7,000,000+ Articles and (e)books	Bluford library
1,500,000+ Articles	IEEE Xplore
6,000,000+ Articles and (e)books	ProQuest

Figure 2.2 and Figure 2.3 represent a sample of the more well-known LCE product design methodologies. The waterfall model, often used in software development processes, was first formally described by (Royce, 1970) as a sequential design process in which progress is seen as flowing steadily downwards through the phases of requirements specification, design, coding, integration, testing and debugging, installation, and maintenance. The waterfall model is the classic software and durable good life cycle model. The model represents the life cycle using processes and products, with each process transforming a product to produce a new product as output. The new product becomes the input of the next process, marking the completion and perfection of the preceding phase, and the progression of a product development processes. Because it requires the completion of a phase of a product's life cycle perfectly before moving to the next phases and learning from them, the waterfall model is viewed as a rigid approach to

product development as a project constantly changes due to requirement modifications and new realizations about the project itself.

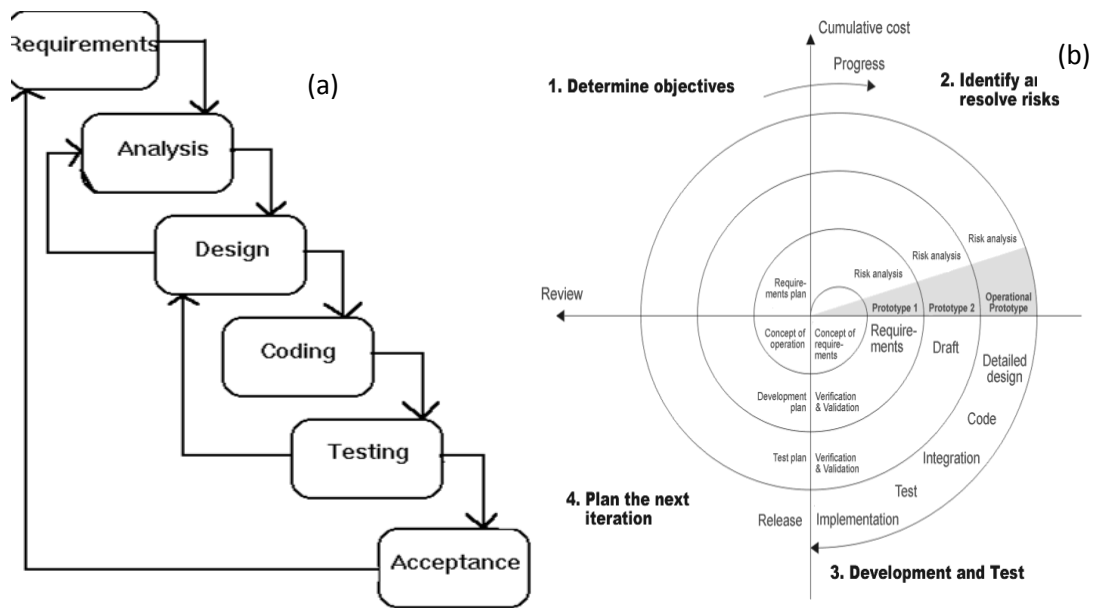


Figure 2.2. LCE methodologies: (a) Waterfall model (Horner, 1993), and (b) Spiral model (Boehm, 1986)

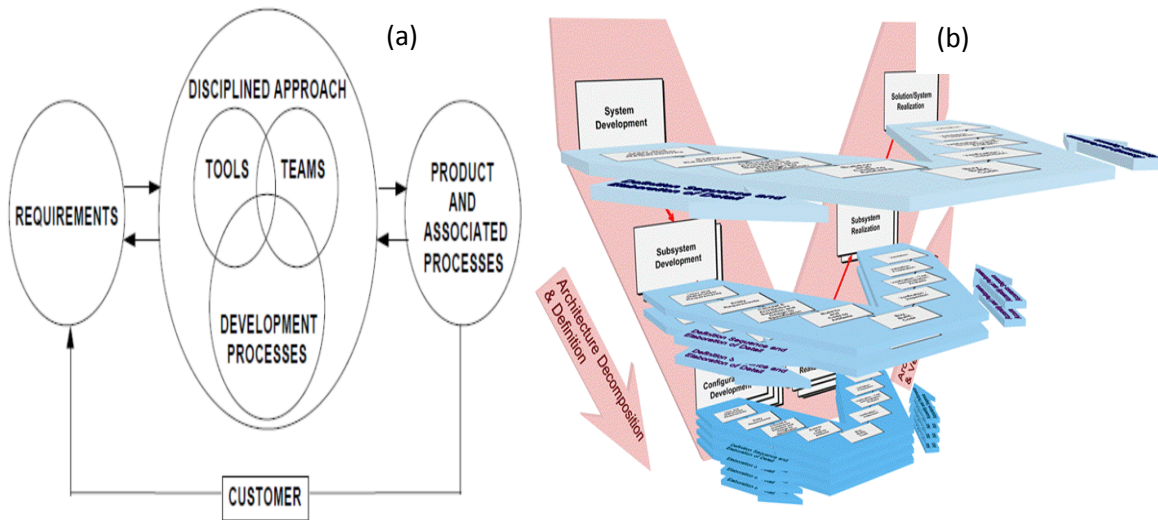


Figure 2.3. LCE methodologies: (a) IPPD model (DEFENSE, 1996), and (b) Dual Vee Model (Kevin Forsberg & Mooz, 1997)

Different approaches have been used to overcome the limitations of the waterfall model. Such methods include having an experienced developer spending time early to consolidate the design or using modularity with interfaces to adjust to the forward momentum the model creates in order to increase the flexibility of the product with respect to the design. Based on the review of the waterfall model, the model would not be adequate as a continuous design approach for sustainable product development.

The spiral model (Figure 2.2.b), also often used in software and durable goods' development process, was originally described by (Boehm, 1986) as a "process model generator" that guides a team of developers working on a design project, to adopt elements of one or more process models depending on the risks associated with the project. Also known as the spiral life cycle model, the spiral model combines elements of one or more process models in an effort to combine advantages of top-down and bottom-up concepts. In (Boehm, 2000), Boehm lists six characteristics or invariants common to all authentic applications of the spiral model. The focus on the system and its life cycle is the last (sixth) invariant of the model, and it highlights the importance of the overall system and the long-term concerns spanning its entire life cycle. As the spiral model continues towards the final phase, the customer's expertise on the new system grows, enabling smooth development of the product meeting client's needs. However, the model needs extensive skill in evaluating uncertainties or risks associated with the project and its abatement. Depending on how intensive the risk evaluation process is, it might translate to extra cost for building the system. The model also requires strict adherence to the project's protocol for its smooth operation, potentially building some rigidity within the overall development process.

The dual Vee model (Figure 2.3.b), often used in systems engineering for the design and development of complex systems, is a top-down model built on the Vee Model to manage a system of systems. The model uses two Vees: (1) an architecture Vee that manages the system, and (2) an entity Vee that branches off the architecture Vee to manage sub-systems. The architecture Vee produces the *what*, *why*, and *who* (which entity level) that are responsible for a system's architecture. The entity Vee illustrates the entity development and realization process, which describes how each entity, will be obtained (development, purchase, reuse, etc.). Within each Vee, the model organizes development phases into levels of complexity with the most complex item on top and least complex item on bottom (Kevin; Forsberg & Mooz, October 1991). The left side of the Vee is about a project definition; the bottom is about the project implementation whereas the right side deals with the project's test and integration. Proceeding this way, the Vee model connects the requirements to the operation, while connecting verification to design. Each Vee within the dual Vee model is flexible as it can either be expanded to meet system requirements or evolve its architecture baseline from initial requirements to a delivered system. A major advantage of the dual Vee model over the waterfall model is the lack of prohibition against exploratory design and analysis at any point in the project cycle to investigate or prove performance or feasibility. A major advantage of the dual Vee model over the spiral model is the opportunity and risk investigations that may be performed either serially or in parallel in the dual Vee model rather than being conducted sequentially and prior to the design development process, as it is the case with the spiral model. Working on a system of systems, the dual Vee model would provide excellent horizontal scaling. However, the model appears not to be inclusive of the life cycle of the system it designs, and not to be accounting for possible similarities between components across subsystems. The dual Vee model

guaranteed performance of a system is limited to the as-integrated and as-verified performance. The dual Vee model also appears to be an expert-based system design approach that does not account for the use of SIAIT in design improvements.

The last of the sampled methodologies is the Integrated Product and Process Design (IPPD). The IPPD is explored within the next section. The IPPD is a model that, in a rather clear fashion, encapsulates some of the emphasis of traditional SE that was mentioned earlier.

2.1.1 Integrated product and process development. Developed in the early 1980s by the U.S. industry as a way to improve global competitiveness, the integrated product and process design (IPPD) concept has its roots in integrated design and production practices, concurrent engineering, and total quality management (DEFENSE, 1996). The U.S Department of Defense (DoD) defines IPPD as, “a management process that integrates all activities from product concept through production/field support, using a multifunctional team, to simultaneously optimize the product and its manufacturing and sustainment processes to meet cost and performance objectives.” IPPD is a generic iterative process with no single solution or implementation strategy. This means that IPPD’s implementations are product and process specific.

In the ideal IPPD scenario, the user knows and communicates his/her needs. The experts, within the design process, listen to the users. An integrated product team (IPT) of people, using their technical expertise, set the requirements, design and manufacture the product. The team works by using multidisciplinary tools with axiomatic design methodology for durable product development (Goel & Singh, 1998). An axiomatic design methodology is a systems design methodology that uses matrix methods to systematically analyze the transformation of customer needs into functional requirements, design parameters, and process variables. Figure 2.4 has a

more detailed overview of the process. With a strict IPPD approach, creativity and innovation are not always part of the solution. Also, the socio-cultural aspect of innovation, the life cycle of the product being designed, the smart capabilities of today's product, and the possible interactions between the parameters affecting the performance of a product, among other factors, were not considered.

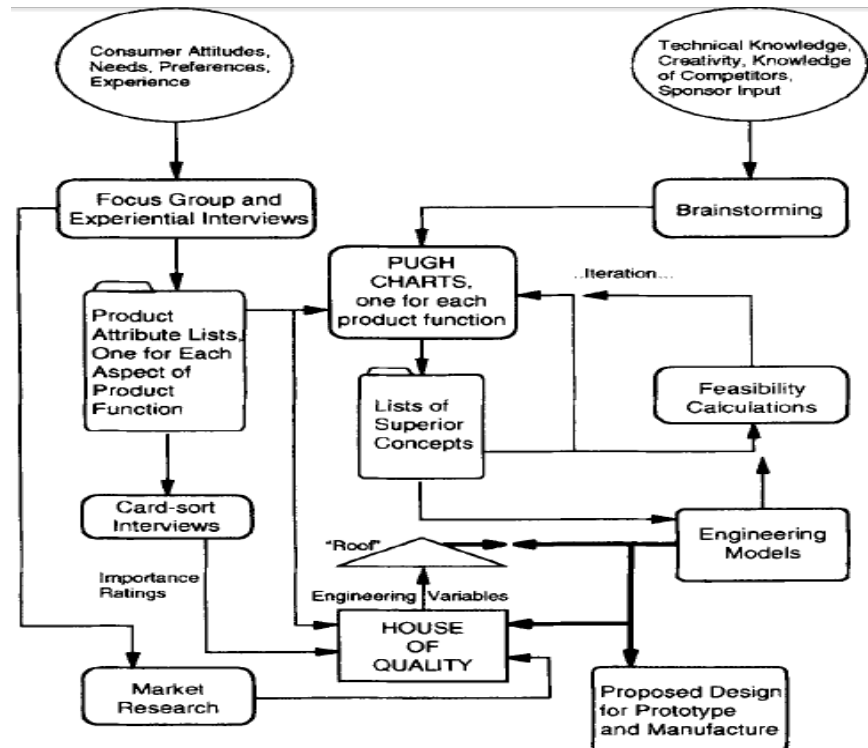


Figure 2.4. Integrate product design process (Hock, 1997)

As a bottom-up approach, the IPPD methodology puts the system engineers as the experts and the enablers of the system. The user of the system is part of the IPPD design process as an input provider. The engineer creates a solution to a problem, serves as the expert; and the consumers and users communicate their concerns. However, the users of a product can contribute more to help the designers generate innovative, functional and more intuitive

products, the users can be turned into co-designers who can add valuable information to the process.

Within the IPPD process, a multidisciplinary team of engineers works to design the best product that satisfies a given set of requirements. The team achieves that by (1) decomposing the system to be built to facilitate its analysis, and then (2) building the system into a configuration that would allow the system to perform well under some given criteria. With the more frequent integration of AIT in products today, what this entails is that the IPPD methodology helps building smart products with rich sensorial and actuator capacities. Those capacities are able to collect data during the life cycle of a given product. IPPD is unable to utilize effectively that data to keep improving the quality of that product.

So far, some of the well-known LCE design methodologies were reviewed and some of their strengths and limits were assessed. The next sections give us some elements of answer to the questions raised within the previous sections.

2.1.2 Deductive product development approaches. Deductive product development approaches (DPDA) are top-down reasoning approaches to product design. DPDA is product design in reverse. Using the data gathered during the life cycle of a product, hidden patterns are mined that can better inform product designers, or IPTs. Such methods are geared towards products wide acceptance via mass customization and/or rigorous testing and validation. Two methods for product design are reviewed: (1) a design for operational feasibility approach; and (2) a user-behavior based approach. Either method naturally contributes to product design with a creativity and innovation touch, key elements to survival and profitability in a rapidly evolving, complex and competitive global business environment.

2.1.2.1 Design for operational feasibility. Generally known as *Design for X*, the design for operational feasibility is used by organizations to guarantee that some essential and desired operational parameters are built into a product/system being realized. The life cycle factors, once selected, are imparted during the design and development of the considered product. A non-exhaustive list of such parameters includes reliability, maintainability, usability, affordability, producibility, supportability, sustainability, recyclability, and disposability. The first four parameters are further explored.

Reliability is defined as the probability of a product to accomplish its designated goal or mission for a given period and when used under specified operating conditions. Reliability is a critical life cycle factor that must be properly defined during the conceptual design phase of a product in meaningful quantitative terms (Henley & Kumamoto, 1985). Designing for reliability allows an organization to have its product evaluated using precisely defined reliability concepts and measures. Three accepted ways or methods of reliability measure are the *mean time between failure* (MTBF), the *mean time to failure* (MTTF), and the *failure rate* (λ). Qualitative and quantitative reliability requirements for a product are developed through feasibility analysis, operational requirements and the maintenance concept identification (Blanchard & Fabrycky, 2011).

Maintainability is defined as the ease, accuracy, safety, and economy in the performance of the maintenance function (Bloom, 2005; Dhillon, 2006). Two accepted metrics for maintainability are the *mean time to repair* (MTTR), and the *mean down time* (MDT). Like reliability, maintainability is design-dependent. Two approaches of dealing with maintainability are through the use of *corrective maintenance* to restore a system or product to a specified level

of performance, and *preventive/predictive maintenance* to retain a system at a specified level of performance (Blanchard & Fabrycky, 2011).

Designing for usability means designing with consideration for the user/operator of the product. Also known as ergonomics or human factors, usability acknowledges the fact that product hardware and software alone will not guarantee good system operability (Lehto, Landry, & Buck, 2007). Designing for usability, the design team would normally consider factors such as: anthropometric (by considering the dimensions of the human body), sensory (by being cognizant of certain human sensory capabilities), physiological (by recognizing the effects of environmental stresses on the human body while performing system tasks), and psychological (by acknowledging the human mind and the aggregate of emotions, traits, and behavior patterns as they relate to job performance) (Blanchard & Fabrycky, 2011). Similarly, a designing team would choose the most adequate approach for measuring the impact of human factors on a product. Two such approaches could be the quantity of personnel required to operate a system or the number of human errors committed per period of time.

Designing for affordability, an organization would design with life-cycle cost in mind. Life cycle cost (LCC) refers to all costs associated with a system: such costs include enterprise costs, users' costs, and societal costs. LCC represents the estimated total incremental cost of developing, producing, using, supporting and retiring a system (Asiedu & Gu, 1998). Initially applied by the US Department of Defense (DoD), the importance of the LCC concept in defense was stimulated by findings that operation and support costs for typical weapon systems accounted for as much as 75% of the total cost (Gupta, 1983). There are many existing tools and approaches to perform LCC analysis. Two such approaches are the LCC by money flow modeling and the LCC by economic optimization. The former approach relies on economic

equivalence expressed as the present/annual/future equivalent, the internal rate of return and the payback period. The latter approach is based on the models of economic evaluation, design optimization, and finite population queuing (Blanchard & Fabrycky, 2011). LCC is the most important of all life cycle factors that organizations designing for X would consider as it is inclusive of the costing of all the activities related to the life cycle of a product.

Organizations use different approaches to achieve their goals when designing for operational feasibility. Approaches used rely on surveys, simulations, stress testing, failure testing, validation testing, experimental design, statistical analyses, and use case scenarios. Design for X reinforces the design of systems to best configuration, and organizational design activities are considered completed right as the product enters its production phase. Within the context of durable product evolution, customers now have a wide range of life cycle decisions they can take that will impact a product life and performance. Such life cycle decisions include but are not limited to change(s) in an organization's policies, the frequency and type of maintenance uses, a decision to scale up an existing system, or a decision to extend the life of a system beyond the manufacturer's recommendations.

2.1.2.2 User-behavior based. Design based on user behavior can be a difficult goal to attain, as that would require a design team to account for the occasional or opportunistic user of the system. Designing with the user-behavior can be achieved for some systems. Computer-based products and services having some sort of user interface, as well as some ergonomically designed goods such as car seats or desks have been designed for a while now with the user-behavior and attitude in mind (Kühme, 1993; Oyewole, Haight, & Freivalds, 2010). Working on the benefits and costs of adaptive user interfaces, (Lavie & Meyer, 2010) reached the conclusion that the preferred type of system depends on a number of factors, such as the frequency at which

the tasks are performed, the user's age, the difficulty level of the task and the level of user involvement in the task. In other terms, a robust system is not enough; the system must be considerate of the user. At the end of a couple of case studies, (Z.-j. Wu, Li, Chen, & Cai, 2010) concluded that designers can acquire interactive relationships between user and product by behavioral process analysis, and design creativity can be realized by creating any new variables of scenarios, actions, or schemes of product part.

2.1.2.3 Bio-inspired product design methodology. There is no known framework that approaches product design from a holistic and complex adaptive system view. Although bio-inspired has been around for some time, it has been used as a way of directly capturing and abstracting the metaphors of nature into product design. Bio-inspired design is used to design products in the traditional sense: leveraging the knowledge of multi-disciplinary teams to design innovative and durable products.

2.2 Biomimetics

The term biomimetics was coined by Otto Schmitt in the 1950s for the transfer of ideas and analogues from biology to technology (J. F. V. Vincent, Bogatyreva, R., Adrian, & Pahl, 2006). Biomimetics operate under the premise that nature works for maximum achievement at minimum effort. In engineering, the reason of mimicking life is to make engineering products adaptable, self-functioning, energy-efficient and reliable (J. Vincent, Bogatyreva, & Bogatyrev, 2007). Biomimetics are used both as computing tools and as a conceptual framework when it comes to engineering design. A review of biomimetics as a tool is given first, followed by its use as a framework.

2.2.1 Biomimetics a computing tool. A subfield of optimization, known as metaheuristics, provides a general algorithmic framework consisting of problem-independent

general heuristic approaches, which can be applied to many optimization problems. Many of the metaheuristic approaches are computational biomimetics. These approaches mimic biological and other natural processes. Genetic algorithms (GAs) are a notable example as they mimic the natural evolutionary process, survival of the fittest, and the natural selection process. Many types of metaheuristic approaches exist including simulated annealing (Cerný, 1985; Kirkpatrick, Jr., & Vecchi, 1983), Tabu search (Glover, 1989, 1990; Glover & Laguna, 1997), iterated local search (Lourenço, Martin, & Stützle, 2002), evolutionary computation (Fogel, Owens, & Walsh, 1966; Holland, 1975; Rechenberg, 1973; Schwefel, 1981), and ant colony optimization (Dorigo & Caro, 1999; Dorigo, Caro, & Gambardella, 1999; Dorigo, Maniezzo, & Coloni, 1996; Dorigo & Stützle, 2004). This section focuses on four metaheuristic types that are bio-inspired. Figure 2.5 shows how some biology metaphors are used in manufacturing.

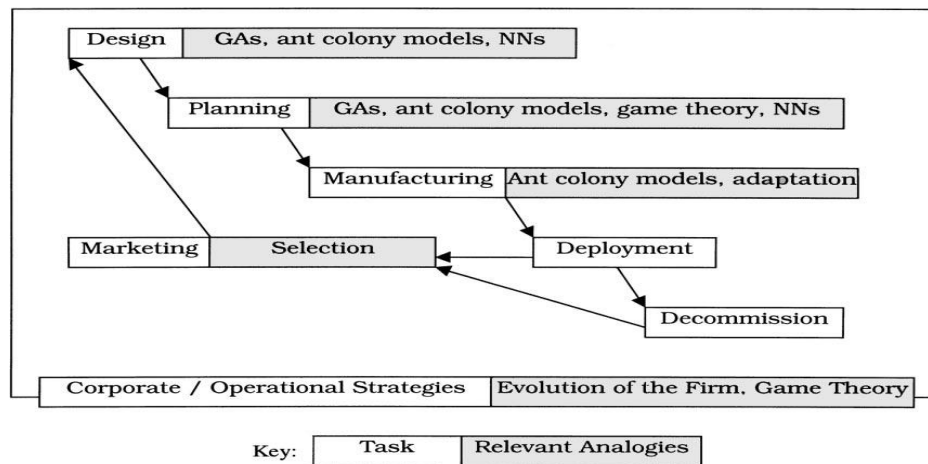


Figure 2.5. Common tasks in a manufacturing firm and relevant biological analogies (Mill & Sherlock, 2000)

2.2.1.1 Ant colony optimization. Ant colony optimization (ACO) metaheuristic mimics the behavior of ants depositing and following pheromone (Dorigo, Birattari, & Stützle, 2006; Dorigo & Stützle, 2003; Dorigo & Stützle, 2004). Ants leave and return to their nest discharging

pheromone on their path. Other ants follow the pheromone before it dissipates, and eventually mark a path that leads to a food source. ACO's premise is that as the amount of pheromone discharged on the path to the food increases, the path to the food will become more "obvious" to the ants. This trait emerges because at the colony-level, the behavior of ants is based on autocatalysis, the exploitation of positive feedback that the ants use to find the shortest path. Developed by Goss et al. (Goss, Aron, Deneubourg, & Pasteels, 1989), a model is built of ants observed behavior in a double bridge experiment in which one bridge is significantly longer. Assuming that at a given moment in time m_1 ants have used the first bridge and m_2 the second one, the probability p_1 for an ant to choose the first bridge is given as:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}$$

where, parameters k and h are to be fitted to the experimental data, and $p_2 = 1 - p_1$ is the probability for ants to choose the second bridge.

The computational model of this behavior has many applications. ACO has been successfully used on different types of problems to include routing, assignment, scheduling, and subset (Dorigo et al., 2006).

2.2.1.2 Particle swarm optimization. Particle swarm optimization (PSO) (Kennedy & Eberhart, 1995; Ozcan & Mohan, 1999) combines social psychology principles and evolutionary computation to mimic social behavior (Kennedy, 1997) as a stylized representation of the movement of organisms in a bird flock or fish school. The movements of the particles are guided by their best-known position in the search space as well as the swarm's best-known position. PSO's premise is that as each particle improves and updates its position relatively to all other particles, all particles will eventually converge to a satisfactorily solution. It is postulated that

some same rules available in PSO underlie animal social behavior, including herds, schools, and flocks, and even that of humans. As particles move within a domain, they modify their velocities based on previous best and global (or neighborhood) best.

$$v_{id} = w_i \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

Where d is the dimension of the domain, c_1 and c_2 are positive constants, $rand()$ and $Rand()$ are random functions, w is the inertia weight of the particle, p_{id} is the particle's best known position, x_{id} and v_{id} are the current position and velocity of particle i , and p_{gd} is the swarm best known position. The adjustment toward p_{id} and p_{gd} by the particle swarm optimizer is conceptually similar to the crossover operation utilized by genetic algorithms.

PSO has various applications and does not need the previous knowledge of the problem space. Applications include scheduling, sequencing, forecasting, traffic management and data mining (Sedighizadeh & Masehian, 2009).

2.2.1.3 Genetic algorithms. The genetic algorithm (GA) metaheuristic (Davis, 1991; Goldberg, 1994; Holland, 1975) mimics evolution and the survival of the fittest. A population of individuals (solution candidates to a problem) interacting evolves over time (generations). The interactions are through mating of “randomly” selected sets of individuals, or mutation of single individuals. GAs’ premise is that as time progresses, the population will naturally improve by preserving its more fit children (survival of the fittest) while discarding its unfit members. Like PSO, GAs belong to the ontogeny category of natural computing paradigms in the sense that it requires adaptation of special organisms to their environment.

GAs have been used for timetabling, scheduling, design, network, rule discovery, and a wide range of engineering problems (Ross & Corne, 1994). Besides their strengths, GAs have

some shortcomings such as its built-in inductive evolution, naturally occurring genetic drift that sometimes causes suboptimal solutions to be created, the highly individualized nature of its operations (crossover, mutation and selection), or their operations and processes that tend to be static rather than adaptive. Those shortcomings cause, to some extent, GAs to underperform for problems where grouping and evolvability are prevalent.

2.2.1.4 Schooling genetic algorithms. Introduced by Wanko and Stanfield in 2011 (Wanko & Stanfield, 2012), schooling genetic algorithms (SGA) are a new GA-based model that enable process and operator adaptability by mimicking fish schooling. Within SGA, operators behave differently depending on the perceived immediate environment and of school dynamics. SGA was designed to address some of the listed shortcomings of GAs, to make GAs suitable for problems where grouping and evolvability are prevalent such as product design for different geographic, social, or economical users' categories.

2.2.2 Biomimetic as a conceptual framework. As a conceptual framework, biomimetic is used both as a way of innovative ideation and as an assessment tool. The next two subsections detail those two uses, their strengths and their limits.

2.2.2.1 Bio-inspired design. From the perspective of design, a number of characteristics make biologically inspired design an especially interesting and attractive problem to study. Biologically inspired design is inherently interdisciplinary (engineering and biology). Both biologists and engineers typically use different terminology, creating communication challenges. Because biologists seek to understand designs occurring in nature while design engineers generally seek to generate designs for new problems, biological designs characteristically result in more multi-functional and interdependent designs than engineering designs. Therefore, the resources, such as materials and processes, available in nature to realize an abstract design

concept typically are very different from the resources available in the engineering domain (Helms, Vattam, & Goel, 2009). Investigating the use of biologically-inspired design as a context from which to teach innovative design, Nelson, Wilson, and Yen worked with mechanical engineering students on design projects (Nelson, Wilson, & Yen, 2009). They found that ideation behavior among mechanical engineering that had a semester-long course specifically focused on biologically inspired design had an average novelty score 80% higher than those from a control group of students that did not take such a class. The results of the findings were statistically significant. Such a study was one of the first to put in evidence the link between bio-inspired design and innovation. Using biological concepts to design can yield to designs that are innovative since it forces the engineer to think like a biologist. However, it still is the responsibility of the designer to find and to harness the analogies.

2.2.2.2 Life cycle assessment. Life Cycle Assessment (LCA) is another framework commonly used by organizations wanting to measure the total environment effect of their product from “cradle to grave.” LCA is a tool used to evaluate the potential environmental impact of a product, process or activity throughout its entire life cycle by quantifying the use of resources (“inputs” such as energy, raw materials, water) and environmental emissions (“outputs” to air, water and soil) associated with the system that is being evaluated (EPA, 17 October 2010). LCA is based around three principles (Duda & Shaw, 1997).

The first principle, known as inventory analysis, entails the identification and quantification of material and energy inputs and outputs for each stage of the product life cycle. The second principle, called impact assessment, helps characterizing the various impacts identified during inventory analysis. And the third principle, called improvement assessment, involves identifying options for reducing environmental burden in product systems and

developing strategies for environmental improvements in the product life cycle. LCA places the onus of the design on the engineer who must carefully inventory the inputs and outputs of his/her product. It emphasizes designing sustainable products, but it does not account for the end user's interests, preferences, and concerns.

2.3 Summary

Within this chapter, a literature review of both product design and of biomimetics was performed. The product design process was viewed both from a top-down approach, and from a bottom-up approach. The structured approach of the latter was elaborated and contrasted with the rather newer and reverse course of the former, which is based on latent knowledge. The design for operational feasibility, also known as design for X was reviewed to show the impact of SE factors on product design. Four life cycle factors namely reliability, maintainability, usability and affordability were further defined and explained. Some gaps were identified within the current approaches to product design to include (1) the non-inclusion of life cycle data from smart product/systems back into the design process for traditional product design approaches, and (2) the reliance of product design processes on expert knowledge.

Biomimetics was defined and reviewed. Application of biomimetics to stochastic optimization processes (select metaheuristics) was reviewed. A metaheuristic was defined as a higher-level search method that uses incomplete or imperfect information to provide a sufficiently good solution to an optimization problem. Some heuristic approaches were defined and explained including ant colony optimization, particle swarm optimization, genetic algorithms, and schooling genetic algorithms. A case was made for the lack of adequate stochastic models dealing with problems where grouping and evolvability are prevalent. These types of problems are very crucial in life cycle engineering and design where the environment,

the culture, legislative and competitive pressure among others require firms to think differently to stay competitive while being innovative and sustainable. Some gaps were identified within the current applications of biomimetics to product design to include (1) the lack of in-depth research and appropriate methods that look at design as an optimization problem, and (2) the lack of known framework that characterizes product design enabling evolvability, grouping, and sustainability.

The gaps identified within the review reinforce and make more specific the intellectual contribution of the current dissertation work. The contribution includes (1) the elaboration of a biologically-inspired framework for product design that uses PLE data, and (2) the conception of a biologically-inspired analytical tool that could at the very least, be used as a complement tool of the framework.

CHAPTER 3

Product Design

In the previous chapter, a literature review of product design and biomimetics was provided. The strengths and weaknesses of some of the existing tools and frameworks were identified, and the gaps addressed by this dissertation were detailed. Chapter 3 details a generalized methodology for product design. The methodology discussed here is about characterizing PLE data in a general way that facilitates the search of metaheuristic solutions, and assists the system/design engineers in making better sense of the factors affecting the product design space as shown in Figure 3.1.

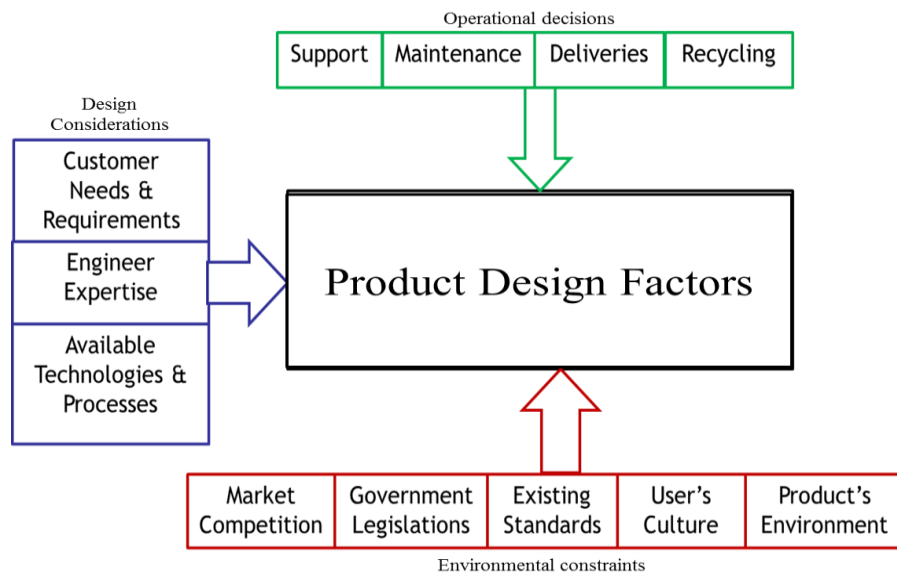


Figure 3.1. Product design factors

Chapter 3 is organized as follows. First, the existing gaps on product design are reviewed. Next, the parameters driving the performance of a product are discussed in more detail, and a non-exhaustive list of some attributes is constructed and explained. After that, a suitable

sustainable performance measurement for our approach is defined. Finally, a new PLE-data representation of continuous sustainable product design is given and discussed.

3.1 Current Product Design Limits

The field of engineering design can be divided into three branches: the traditional school (still dominant), the algorithmic school, and the axiomatic school (Suh, 1999). The traditional school believes that design is a creative process, which cannot be completely performed by deductive reasoning, and requires experience. The algorithmic school relies on optimization tools such as Genetic Algorithms, Neural Networks, or Fuzzy Logic to achieve the best possible design based on some design goals. The axiomatic school is based on the premise that there are generalizable principles that form the basis for distinguishing between good and bad designs. According to (Suh, 1999), a good design needs to use all three methodologies when going through all the required design activities. Different approaches exist that use or combine together any of the three approaches.

(Nelson et al., 2009) research with engineering students working on their design projects found in a statistically relevant experiment that ideation behavior, and therefore the creative process, can be infused through the use of biologically inspired design. On a study focusing on the collective beliefs of managers in competing firms and how they interpret and respond to successful technological innovation, Jenkins identified some of the potential interplay between design innovation and design imitation by organizations in order to sustain an incremental innovation (Jenkins, 2013). The study suggested a more nuanced way of considering incremental innovation by extending the potential opportunities for creating competitive advantage through innovative imitation and also imitative innovation. Therefore, there are many ways a company

can use and sustain the traditional design views of design. Both the use of biological models and the use of models from competitive marketed products appear to be viable sources of inspiration.

The algorithmic approach to design relies on mathematical processes to solve problems related to design. The algorithmic approach automates some aspects of the design process, enabling design engineers to compute optimal parameters and dimensions of the design that would maximize or minimize some design objective (Kumar, 2005). The considered aspects of the design process would include, among others, the enhancement of customer satisfaction (Chen & Chuang, 2008), the streamlining of the supply chain (Akanle & Zhang, 2008; Elimam & Dodin, 2013; Ghasimi, Ramli, & Saibani; Kabak & Ülengin, 2011), the product specification process (Wallace, Jakiela, & Flowers, 1996), the improvement of the production system (Jeong, 2000; Ohno, 2011; Stanfield, King, & Joines, 1996), or the minimization of the product overall life cycle cost (Janz, Sihm, & Warnecke, 2005; Massarutto, Carli, & Graffi, 2011). However, whether considering SE with life cycle based factors or the engineering activities of an enterprise, existing algorithmic approaches do not consider the ambient intelligence concept. Ambient intelligence is the convergence of ubiquitous computing (useful, pleasant and unobtrusive presence of computing devices everywhere), ubiquitous communication (access to network and computing facilities everywhere), and intelligent user adaptive interfaces (perception of the system as intelligent by people who naturally interact with the system that automatically adapts to their preferences) (Kopácsi, Kovács, Anufriev, & Michelini, 2007). Ambient intelligence is a natural result of the evolution of durable product, which is enabled by smart capabilities that can provide data collection, storage, processing, and communication capabilities with minimal power requirements such as previously depicted in Figure 1.1. Within an ambient intelligence area, the algorithmic approach of system design ought to include the best

way to integrate smart capabilities into a design and to be part of the product life cycle from conception to retirement/recycling instead of sales. Using the algorithmic approach to blend the life cycle results with the reuse of the enterprise expertise acquired while working on previous products can help an organization make its design sustainable. However, such a new way of using the algorithmic method to design requires a different approach that will make the design process life cycle-based and continuous (design never ends).

The axiomatic design approach has contributed to the advancement of design practice, and design evaluation criteria based on design axioms. Such design evaluation criteria include (but are not limited to) level of innovativeness, quality of design, intuitiveness of design, functionality, choice of material, safety, the positive influence on the environment, and some life cycle factors such as ergonomics, reliability or affordability. Most of the metrics set in place look at design as a static activity. Once prototyped, tested and validated, a product is set to have met the design requirements, and as the design phase of the product is considered completed, the product enters its production phase. Extrapolating from the axiomatic design perspective, a good design is a differentiation factor of a product from the competition. A good design is one of the key factors a consumer would consider when deciding whether to acquire a product. However, the factors affecting the way a user looks and assesses a product change over time. As listed in Figure 3.1, some of those factors can be decided by the user such as the frequency and the type of maintenance to perform. Some other factors, such as the functions to be built into a product, can be decided only by the manufacturing firm. There are other remaining factors, such as the operational environment, that are not set at the discretion of the user or the designers. Therefore, knowing the relationship between the performance of a product and the life cycle factors that impact it can only facilitate the work of the designer by putting in place a sustainable product

design process. Yet, the axiomatic approach typically does not consider design as a holistic life cycle sustainable process.

Finally, consider the objective of a systems engineer to find the best solution that optimizes the operation of a product within a given configuration, context, and environment. From the SE traditional perspective, sustainability means ecological balance to avoid depletion of natural resources. In other words, being sustainable equates to being environment friendly. Sustainability is accomplished by having a small footprint on the environment by using less material, shipping with smaller or recycled packaging, being free of many toxic substances and being as energy efficient and recyclable as possible. Several methods, both qualitative and quantitative, have been proposed to solve the design problem from the sustainability point of view. Some of the methods include qualitative matrices (Allenby, 1992), abridged LCA (Graedel, Allenby, & Combrie, 1995), checklists (Clark & Charter, 1999; Fiksel, 1996), LCA streamlining (Mueller & Besant, 1999), eco-design (Braungart, McDonough, & Bollinger, 2007; Knight & Jenkins, 2009), and Whole Systems Design (Blizzard & Klotz, 2012). Nevertheless, all the methods listed recommend good and sustainable design based on both the knowledge of the engineer and the projected impact of a product on its environment. The approaches appear not to build on the smart capabilities built into products. The design approach presented here defines sustainability as the ability of a product to be designed, operated and supported with the least possible intervention of the systems engineers. The approach is life cycle centered and relies on life cycle parameters.

3.2 Life Cycle Parameters

Product life cycle parameters are factors that impact the life cycle performance of a product. The objective is to find the different factors, from conception to retirement or recycling,

that are key determinants of the product's performance. The concern this section addresses is a way of measuring, tracing and tracking the inputs that affects performance in an objective, and comparable manner. The case was made in last section on the necessity for using PLE-data to make designed products perform better and be more innovative. The case was made about factors that affect the performance of a product or system. Some factors such as the type of maintenance performed on a system, the type of functions available on a system, and the environment where the product is used are functions of the user, the designer, and the environment respectively. In order to proceed further, a categorization of the parameters that can impact the performance of a product is made. Life cycle parameters are divided into three categories, namely design, operational, and environmental. Figure 3.2 shows the relationships between the parameters and the performance within the systemic view of a product.

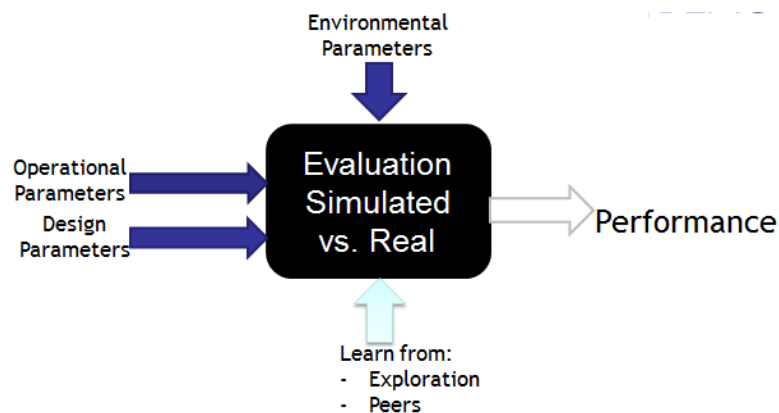


Figure 3.2. Systemic view of a product

Figure 3.2 illustrates that a product's performance is a function (whether simulated by a model or observed in the real system) of the interactions between that product design, operational, and environmental parameters. Figure 3.2 also illustrates individual learning by a product through user's interactions, and group learning by the product through interactions with other products. Using the same parameters to group products according to their similarities can inform the systems engineer or designer of the relevance of one parameter or type of parameter.

A similarity based grouping can help the systems engineer make some informed decisions to either decrease production cost while maintaining product performance, or to improve performance by changing an existing system configuration. Using a similarity based grouping, a design parameter could be turned into an operational parameter, and vice versa, provided that the existing product makes the shift feasible. Figure 3.3 shows the performance plot of the instances of a hypothetical product grouped according to its design parameters (vertical axis), operational parameters (horizontal), and its operating environment (shape: green diamond vs. red square).

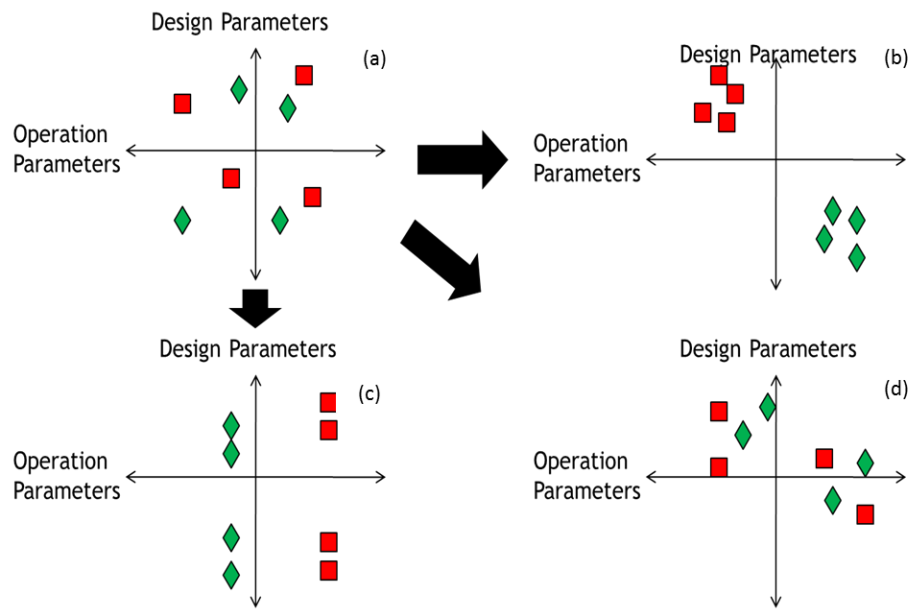


Figure 3.3. Impact of parameters on product performance

Suppose Figure 3.3(a) is the starting point for a metaheuristic search. It represents the product designed with diversity (different design and operational value combinations for each environment) built-in. The other three graphs (following directional arrows) represent three possible outcomes if each product is able to improve itself during its life cycle. Figure 3.3(b) shows two distinct clusters, one from each environment. Such a grouping tells the designer that the combination of design and operational parameters should be customized to the environment.

Figure 3.3(d) shows no distinct grouping and with reduced range along the design parameter axis. Finally, Figure 3.3(c) represents a case the operational parameter should be customized based on environment. The design parameter might be common (and appears to be flexible) for both environments. Table 3.1 shows a sample of parameters. It is important to note that engineers set the design parameters of any product before the production of that product begins. They are infrequently changed. Operational parameters are those that can be changed by the user to meet a specific use or need. The environmental parameters are tied to the environment and are out of the designers of users' control. The attributes listed within the table come from the literature review, and they will be described later in this chapter.

Table 3.1

Samples for life cycle parameters

Design	Operational	Environment
Layout	Type of use	Temperature
Material	Frequency of use	Humidity
Functions	Type of Maintenance	Culture
Sensors	Frequency of Maintenance	Infrastructure
Size	Custom settings	Regulations
Shape	Number of resets	Similar Products
Dimension		Climate
Manufacturing Process		Location

Setting the operational parameters is the responsibility of the user or maintainer. Figure 3.4 shows a collaborative life cycle. A collaborative life cycle is formed by the different user-to-product (operational parameters), and product-to-product interactions that occur during a

product's life. A collaborative life cycle can better inform the designers of a product when working on the next generations of the product. A collaborative life cycle enables group learning through products' interactions. Considering that the success of product updates strongly depends on consumer heterogeneity, on the rate of content consumption, and on the social interaction of consumers (Albuquerque & Nevskaya, 2012; Keeney & Lilien, 1987), using a product smart capabilities to capture the changes in operational parameters during product's interactions can give designers valuable insights on the next generation of a product.

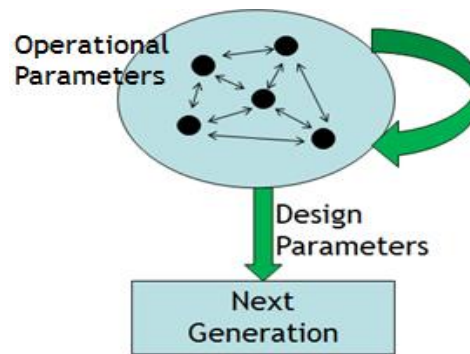


Figure 3.4. Collaborative life cycle

Therefore, by using PLE in product design, the PLE-data the engineers need to work on can be modeled as an aggregation of design, operational, and environmental parameters. Table 3.2 shows a brief summary of the attributes of each category of parameter.

Table 3.2

Classifying LCE data

Parameters	Controllable	Evolution cycle	Associated LCE stage
Design	Yes (Engineer)	Slow	BOL
Operational	Yes (User)	Fast	MOL+EOL
Environmental	No (Environment)	N/A	N/A

Each of those categories of parameters will now be defined, and the meaning of the use of some attributes given.

3.2.1 Design parameters. Design parameters are the qualitative and quantitative aspects of physical and functional characteristics of a component, device, product, or system that are input to the product design process. The design parameters determine the cost, design, and risk tradeoffs in an item's production. The design parameters are set by the engineers who use matrix methods to systematically analyze the transformation of customer needs into functional requirements, design parameters, and process variables. Design parameters are set by the engineers during the design process, and are not changed easily. Examples of commonly used design parameters are now described.

3.2.1.1 Material. A material is the matter from which a product is or can be made. The choice of materials to be used for the design of a product is of strategic importance. These days, the criteria choice of material used in components not only have to meet some given functional and performance requirements, but must also account for environmental considerations to minimize the environmental impact associated with the product's entire life-cycle (Giudice, La Rosa, & Risitano, 2005; Mayyas, Qattawi, Mayyas, & Omar, 2012).

3.2.1.2 Functions. Functions are defined as the "product's answer to the set of user tasks"; unlike features that are the "user tools" inherent in the product used to perform the functions (Technologies, 2012). It is common to see products with the same functions, but with different features. Product types have different sets of functions, and each model within a type of product accomplishes its functions through potentially different features (Technologies, 2012).

3.2.1.3 Sensor. By definition, a sensor is a device that can detect or measure a physical property and record, indicate, or otherwise responds to it. Sensors are used to allow devices

becoming aware of something via the senses. Sensors are the counterparts of actuators. As mentioned earlier, the evolution of products from a mechanized to a more biological entity has been enabled thanks to the use of AIT components, some of them being sensors. Today, a cell phone for instance is a very complex device capable of sensing temperature, altitude, location, and motion at a minimum. The decision to include a specific type of sensor within a product may be more of a strategic decision (Seltzer, 2012) than an engineering decision, but the choice of the make and the integration of a given type of sensors are the prerogative of the design engineers.

3.2.2 Operational parameters. Operational parameters are defined as a product functional attributes that can change with values set by the user or maintainer of that product. Operational parameters are product features that allow flexibility. The operational parameters are set by the user to get a service. Operational parameters are under the control of the user and may be changed frequently. The following are examples of operational parameters that can impact the performance of a product.

3.2.2.1 Type of use. The type of use is defined within the context of a product with many different features. Within that context, the type of use is a measure that tracks the type of feature the user of the product makes use of. The type of use reveals among others the preferential use of the product by the main user. The type of use can prompt engineers to make a given feature of a product better, or to attach a given service or related product to the initial product.

3.2.2.2 Frequency of use. The frequency of use keeps track of the frequency with which any feature or component of a product is used at any given time by its user or other components. From an engineering reliability standpoint, some components are more likely to fail based on their usage frequency. Metal fatigue, in material sciences, is a typical example. Therefore, the

frequency of use can inform engineers about the part of the design that can focus on to make their product more reliable or more appealing.

3.2.2.3 Type of maintenance. Maintenance is defined as the process of keeping something in good condition. Complex products such as an airplane will require different types of maintenance during its life cycle. The maintenance can be preventive (performed specifically to prevent faults from occurring), predictive (performed to determine the condition of a system in order to predict when maintenance should be performed) or corrective (performed to identify, isolate, and rectify a fault so that a failed system can be restored to an operational condition within the prescribed tolerances or limits). The type and/or frequency of maintenance tend to be set according to an organization's policy.

3.2.3 Environmental parameters. We define environmental parameters as the attributes of the context or environment a product is being operated or used. It is not part of the product per se, but part of its surroundings. The user cannot change environmental parameters, but they do affect the way the product performs. The following are examples of environmental parameters that can impact the performance of a product.

3.2.3.1 Physical environment. Temperature, humidity, and vibration all fit within this category. Temperature for instance can directly affect product performance (Fakhim, Behnia, Armfield, & Srinarayana, 2011; Larrosa-Guerrero et al., 2010; PILCHER, NADLER, & BUSCH, 2002). The effects of temperature, whether low or high can either enhance or impair the performance of a product. Keeping track of the performance of a device along with the operating temperature over time can help an engineer not only better understand the relation between performance and temperature for given environments, but also better decide on the make of a specific product component.

3.2.3.2 Alternative products. Those are also known as competition. Similar products altogether with the product of interest define the ecosystem. Watching the competition's market serves a twofold interest. On one hand, it allows a company to stay knowledgeable and carefully follow the design trend of the market, and on the other, it allows a company to engage in a proactive learning. According to (Bapuji & Beamish, 2008) any company, in order to avoid hazardous design flaws, should at a minimum study competitors' recalls, overall recall trends, issues leading to recalls, or regulators' comments. There is a lot that can be learnt, either directly or indirectly, from the competition.

3.2.3.3 Culture. The definition of culture used here is the distinct ways that people living in different parts of the world or of a country classified and represented their experiences, and acted creatively. Although (Im, Hong, & Kang, 2011) showed that the UTAUT model is affected by culture, the model strengthen the evidences according to which the acceptance of a product depends not only on the way it was designed by the engineers, but also on the way it is being perceived by the users within a both personal and socio-cultural context. The knowledge of such a context calls for the need of a design tool that can inform the engineers about the users at large and their environment in order to design great products.

3.3 Sustainable Performance

The mapping of life cycle parameters to performance, treated by this section, is not a trivial issue. Performance measurements should carefully be defined whenever designing a system. A performance measure is an indicator of progress toward achieving a goal. The design of performance measurement systems appropriate for modern organizations is a topic of increasing concern for both academics and practitioners (Neely, 1998). According to (MWG, 2010), performance measurement in SE aims at helping managers controlling the SE processes to

improve the quality, timeliness, efficiency, and effectiveness of the products and supporting processes.

The ability to measure any significant activity associated with a product during its life cycle is critical. Such measurements give the engineers the aptitude to improve the design of the product and of all the associated processes, to allocate or reallocate resources to that product, or to compare that product against the competition. A system with sustainable performance accounts for the evolution of the product and processes tied to it.

As an example, the performance of a procuring function in a firm could be gauged by the costs and availability of raw materials. The availability of a production system within an organization could be gauged by its reliability (probability of a system to perform its designated mission for a given period when used under specified conditions), its maintainability (the ease, accuracy, safety and economy in the performance of maintenance function), its supportability (ability to install, configure, monitor, identify issues, and restore systems), and its producibility (the capacity of making goods and services). The performance of a manufacturing function could be gauged by capacity utilization, defects, output, and down time; whereas the performance of sales within an organization can be gauged by the (preferably high) amount of sales and of the (preferably low) number of returns. However, devising a model that spans across the entirety of the life cycle spectrum requires an adequate performance measure that will go with it.

Performance or measures can be based on *a priori* set multi-criteria, or on aggregated targeted objectives. The MACBETH (Measuring Attractiveness by a Categorical Based Evaluation Technique) or the Analytic Hierarchy Process (AHP) approaches are examples of multi-criteria decision analysis approaches that are used in different industries and fields as an efficient technique for rank ordering alternatives (R. W. Saaty, 1987; T. L. Saaty, 1982, 1990,

2005; Zahedi, 1986). However, when defined as an aggregated value, performance can be formalized using the following mapping (Berrah, Mauris, & Vernadat, 2004):

$$A_g : E_1 \times E_2 \dots E_i \times \dots E_n \rightarrow E$$

$$(p_1, p_2, \dots, p_i, \dots, p_n) \rightarrow p_{A_g} = A_g(p_1, p_2, \dots, p_i, \dots, p_n)$$

Where the E_i 's are the universes of discourse of the elementary performance expression

$(p_1, p_2, \dots, p_i, \dots, p_n)$ and E is the universe of discourse of the global performance expression p_{A_g} .

And since that the universes E_i 's and E can be different, the determination of the aggregation mapping A_g , which is generally not straightforward, would require some heuristics to deal with the heterogeneity of the life cycle data. Going further, since the objective here is to assess a life cycle-based performance measurement from an SE point of view, a more appropriate example of an aggregated performance measurement known as the System Operational Effectiveness (SOE) is described.

The SOE is a concept that was defined to reflect the holistic objective of SE and integration efforts in achieving a balance between system performance, availability, process efficiency (operational, maintenance, and support processes), and total system ownership costs. Figure 3.5 shows the SOE model. The SOE requires proper attention and balance among all the factors included in the SOE model in order to maximize operational effectiveness, and to prevent risks and challenges associated with end-of-life obsolescence (Verma, Farr, & Johannesen, 2003). An example of unbalanced approach could consist of a disproportionate allocation of resources and attention to one area (e.g. performance) at the expenses of others (e.g. availability

or efficiency) and could lead to an excessive ownership costs. The SOE model is an aggregated performance measure.

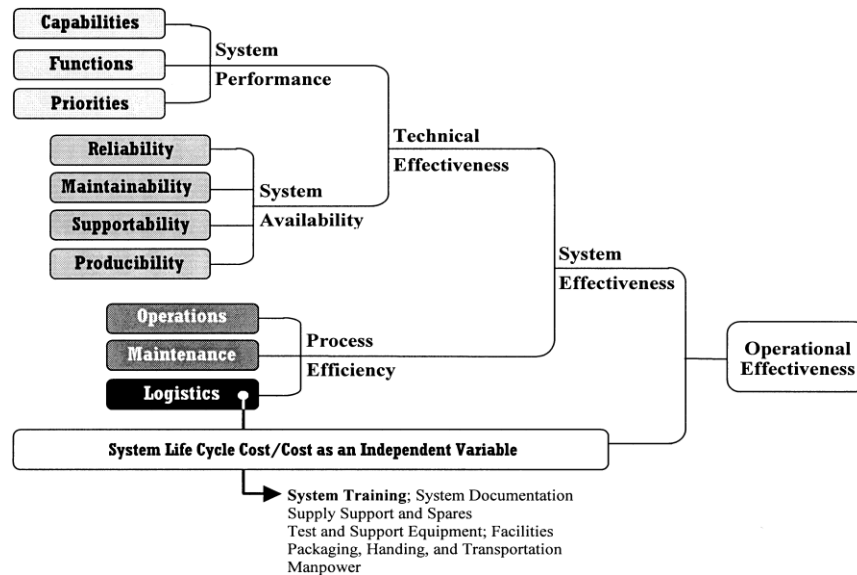


Figure 3.5. System Operational Effectiveness (SOE) (Verma & Gallois, 2001)

The next section synthesizes all the components of the LCE data characterization, and explains how the components are integrated. The section also makes the case for a generalized approach to product design that could be applied to any design methodology that is inclusive of the life cycle data.

3.4 Generalized Life cycle Product Design

The generalized life cycle product design (GLPD) is a generalized PLE-data based product design approach resulting from the proposed LCE data classification. As its name implies, GLPD is a product design approach that relies on the data collected during the life cycle of a product or system to better design products or systems. Table 3.3 shows some GLPD process elements. The processes are categorized both by the life cycle phase they belong to, and

by the desired characteristic of the product they target: evolvability (E), sustainability (S), and grouping (G).

Table 3.3

GLPD process elements

Life Cycle Phase	Processes	E/G/S
Research / Development	- Determine key life cycle parameters	
	- Create initial parameter categorization	
Design / Simulation	- Create performance evaluator/simulator	
	- Search to find	
	- High performance configurations	S
	- Grouping opportunities	G
	- Consider parameter shift	
	- Design to operational	E
Operation / Sustainment	- Operational to design	G
	- Use actual system performance	S
	- Enable evolution through	E
	- Change in objective	E/S
	- Change in parameter representation	E/S
	- Enable group-based efficiency by	G
	- Group configuration evolution	E/G
	- Group merges and division	E/G
	- Enable sustainability through	S
	- Metaheuristic-driven group and individual change	E/G/S

The GLPD approach is a top-down-up approach. A top-down-up approach is one that is both top-down and bottom-up. A top-down-up approach is defined here as an approach that decomposes an existing system into smaller components with the goal of providing a holistic perception and an improved reconstruction of the system with a subset of the same or similar

components. GLPD aims at assisting engineers with the design of innovative and creative products that are user-centered, evolvable, and sustainable. GLPD is a life cycle data sourced approach that integrates with current practices and views a product design process as continuous. GLPD recognizes the ongoing evolution in durable product, as well as the increasing use of smart components within systems to offer a continuous design representation. Rather than solely focusing on the BOL part of the life cycle during the design process, GLPD recommends using the data gathered during the lifetime of similar products or previous versions of a product when working on a prototype for that product next generation. Figure 3.6 represents the GLPD approach for continuous design, along with all its components.

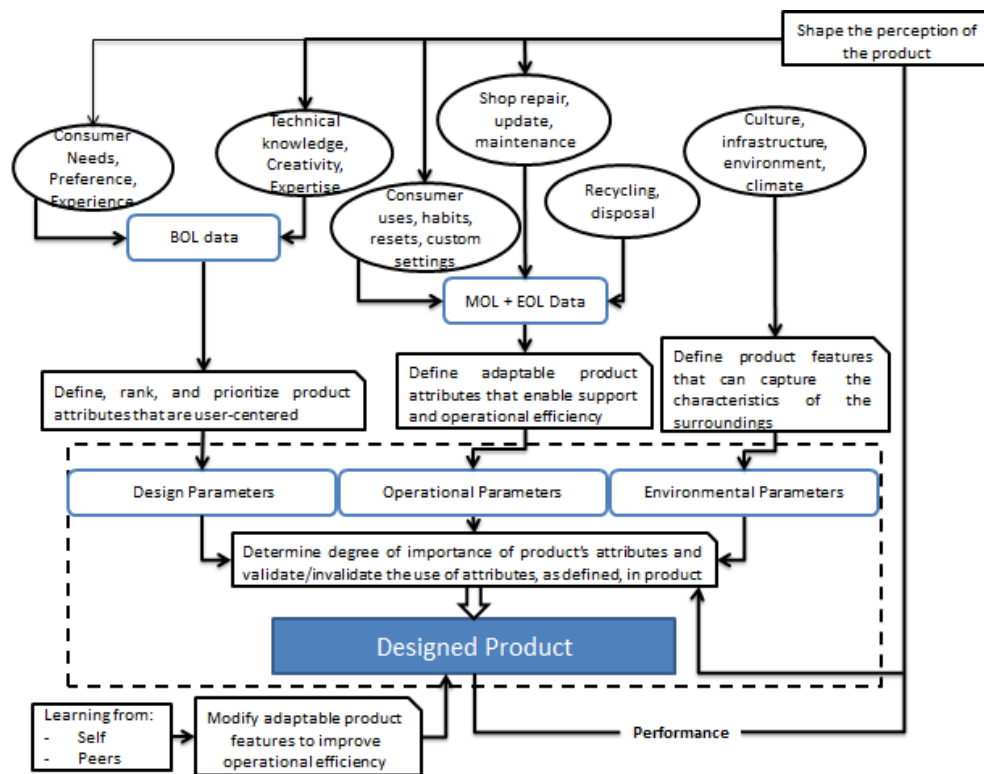


Figure 3.6. Generalized life cycle product design (GLPD) approach

The GLPD approach is iterative, continuous, and spans through the life cycle of a product or family of products. The approach forms a closed loop process that can sustain the design process by constantly processing inputs (life cycle data) to turn those inputs into “better”

characteristics for a product. The iterative nature of the GLPD approach enables a rapid transition from simulated to realistic design, supporting this way any design methodology that promotes or uses short life cycle. Being continuous and iterative, GLPD is also able to respond rapidly to changing customer needs occurring during the life cycle usage of a given product. Therefore, GLPD can be viewed as a decision making tool as it allows organizations to respond to local changes by using information specific to a given locale: the life cycle data. The classification of life cycle data enables GLPD to provide a life cycle holistic view of the design process by integrating the various life cycle parameters into the design process. Such a holistic view of the life cycle, when used with an aggregated performance measurement, provides the design process with a performance modeling tool. By representing a product as an assembly of three defined life cycle classes, GLPD creates an implicit mapping between an aggregated life cycle-based SE performance indicator and the evolving characteristics of a product. That mapping or relationship between product's performance and the life cycle data classes enables the designer to decide on whether to turn a design parameter into an operating parameter (or conversely), without impacting the performance of the product.

Using the GLPD suggested representation of a product (dashed box in Figure 3.6), a metaheuristic search tool can be used that cluster products along the defined life cycle parameters, looking for possible relationships between the life cycle classes and performance, and determining the degree of importance of lifecycle attributes to the performance of a product.

Grouping, as enabled by the GLPD representation provides the designer with design alternatives for a given range of desired performance values. Grouping would make the solution of a GLPD-based approach capable of scaling and of being flexible from a configuration point of view as shown in Figure 3.7.

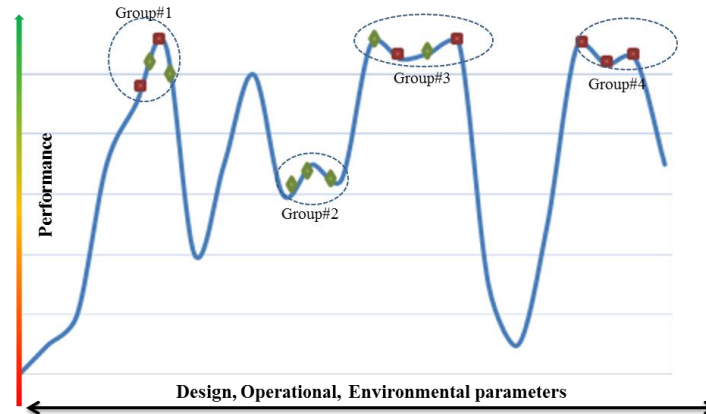


Figure 3.7. Grouping vs. performance

Using the same shape nomenclature for the product as in Figure 3.3, Figure 3.7 shows a possible outcome of grouping which results in four groups with two of them presenting interesting properties for a system designer. Solutions from group#3, like those from group#1 and group#4 are performing well. However, because group#3 and group#4 are more spread out than group#1, they offer a better starting point for risk analysis, and further performance improvement for a system. Also, when comparing group#3 against group#4, group#3 offers a diversity that can be used for instance: (1) to build efficient systems that the change of environment will not significantly affect, (2) to scale a system without adverse effects on the overall performance of the system, or (3) to build efficient systems capable of operating in heteroclitic environments. By being cognizant of the relationships in case they exist, between LCE parameters, a systems engineer could decide to change the class of a product performance parameter. Such a change could include a shift of a lesser-used performance parameter from operational to design without hurting a system performance.

Finally the GLPD approach, when applied to a bigger and complex system such as a vehicle or an aircraft inventory with numerous interchangeable parts can help a systems engineer, using the appropriate grouping tools, to simplify the design of that system by enabling

him/her to either discover interoperable parts that impact that system's performance, or parts of the design that, because of their nature, could be pushed from design to operational without compromising that system long term performance.

3.5 Summary

Within this chapter, three views of engineering design, known as the traditional, the algorithmic, and the axiomatic views were explored, and their limitations identified. The limitations were primarily identified as caused by the ongoing trend of using smart components in complex systems. The use of smart components is changing the way systems engineers look at life cycle data. A case was made for a continuous design process that can be self-sustaining. A generalized, iterative, and highly responsive design process called generalized life cycle product design (GLPD) was given. Performance measurement, key attribute of the proposed GLPD methodology was discussed. After addressing some commonly used performance measuring systems, a case was made for one approach that uses aggregated data, with heuristics or real experience as a mapping function between the proposed life cycle parameters set and the performance universe that any organization can set to assess and improve its life cycle. The components of the proposed methodology namely the life cycle parameters (LCP) were explained and some components were defined, with their importance and their meanings explained from a design point of view. An example of interactions between design and operational parameters was discussed for a hypothetical product. A case was made for GLPD to be used in conjunction with an appropriate clustering/grouping tool to complement expert knowledge, with aggregated knowledge from life cycle data. Finally, an example of grouping, as enabled by GLPD was given, followed by possible interpretations.

CHAPTER 4

Schooling Genetic Algorithms

This dissertation seeks to develop optimization models dealing with problems where grouping, evolvability, and sustainability are key characteristics of the solution. These characteristics are prevalent in smart system LCE and design. Considering the nature of the described problems, metaheuristics, with the emphasis here being put on evolutionary methods, provide a foundation to solve the described problems. As a result, an enhanced metaheuristic search approach that uses biological inspiration and grouping by exploiting fish schooling group dynamics is created. The method is termed “schooling genetic algorithms”.

Schooling genetic algorithm (SGA) are GA-based models that enable process and operator adaptability by mimicking fish schooling. SGA represents an adaptive type of metaheuristic where operators behave differently depending on the perceived immediate area of the search domain in the context of fish schooling dynamics. SGA was built from concepts with multiple objectives in mind, notably the ability to “naturally” group candidate solutions based on some type of similarity, the ability to “intuitively” ungroup clusters of candidate solutions based on the local perception of the immediate environment, and the ability to exploit the evolvability of a subpopulation to possibly predict the performance trend of that subpopulation. This chapter serves as a detailed introduction to SGA.

4.1 Introduction

Genetic algorithms (Holland, 1975) are global optimization techniques inspired by the mechanisms of natural evolution. GAs are useful both as search methods for solving problems and for modeling evolutionary systems. GAs operate on a population of individuals (or potential solutions to a problem) and within a domain without an explicit mathematical description. GAs

work by applying the principle of survival of the fittest to achieve an optimal (or strong) solution.

The successful application of GA mainly depends on the population size or the diversity of individual solutions in the search space, and on the devising of the crossover and the mutation operators. If the GA cannot hold its diversity well before the global optimum is reached (ideally), it may prematurely converge to what is known as a local optimum. Such a phenomenon is known as a genetic drift, which is responsible for reducing the genetic variation within a population. Though maintaining diversity is the predominant concern of GA, it also increases the computational cost of GA. Various techniques have been attempted and used to find a balance between the population diversity and the performance of GA (exploration and exploitation).

An approach that has proved successful at increasing the diversity and exploiting its benefits is the use of distributed subpopulations. It is a “divide and conquer” approach that allows tackling increasingly complicated cost functions emanating from complex simulations common for engineering design problem. For solving complex problems, parallel GAs are used. Combining GAs with other alternatives has often proved to yield better results than traditional GAs, which uses a single large panmictic population. In parallel GA, it can reasonably be argued that having multiple subpopulations helps preserving the genetic diversity, since each subpopulation can potentially follow a different search trajectory through the search space. Two parallel GA approaches which have proved to be successful are known as the Island Genetic Algorithm (IGA) (Cantú-Paz, 2000), and the Niching Genetic Algorithms (NGA) (Mahfoud, 1995). The remainder of the chapter will be structured as follow. Since SGA represents an inherent form of parallel genetic algorithms, parallel GA (notably NGA and IGA) are described.

Using biology, the context of SGA will be situated followed by a detailed explanation of the SGA procedure and life cycle.

4.2 Parallel Genetic Algorithms

In ecology, a niche is defined as the sum total of an organism's use of the living and nonliving resources in its environment. Mahfoud and Watson both suggest an evolutionary classification method based on the way multiple niches are found in a GA (Mahfoud, 1995). In GA, niching refers to the notion that competing individuals or species cannot coexist in the same local environment. Both spatial and temporal (sequential) niching approaches have been used in the literature. Spatial methods can be further categorized depending on whether they use sharing, crowding, or clearing methods, whereas temporal or sequential niching methods find multiple niches iteratively or temporally. By maintaining subpopulations in so-called "niches", NGA prevents genetic drift and forces parallel convergence within the solution domain. The two main objectives of niching algorithms are (i) a converge to multiple, highly fit, and significantly different solutions, and (ii) a slowdown convergence in cases where only one solution is required (Mengshoel & Goldberg, 2008).

IGA operates differently as it does not avoid genetic drift, but isolates the subpopulations that are only allowed to exchange information via a migration operator. The migration operator acts as a fitness-based probabilistic selection operator for migration selection and replacement. In IGA, a third operator (besides mutation and crossover), known as the migration operator, is responsible for the communication between these "islands" of chromosomes. The communication occurs infrequently. The separation into subpopulations aims at preventing premature convergence by acting as a non-dominating strategy for the population as a whole. Such a scheme can lead to the finding of multiple solutions to a problem. Therefore, IGA has

been proven to be more suitable for linearly separable problems, although also slower (implementation) since it tends to perform better on larger total population sizes (Whitley, Rana, & Heckendorn, 1998). Although IGA is parametric (requires at least the provision of the given number of islands as a parameter in order to get the full benefits of using subpopulations), its implementation arbitrarily assigns solution candidates to islands.

Although both reviewed parallel strategies (IGA and NGA) involve groups of genes, which are usually sent to separate processors to evolve apart from the rest of the population (common characteristics of parallel GAs), the approach used in each case is what makes them different and unique. However, maintaining all islands and/or niches during the search process does not take advantage of the topology (attractors) of the solution domain. Further, both methods despite being inherently parallel are not using mutation and crossover in an advantageous way. The basic idea behind mutation is to reintroduce divergence into a converging population through exploration, whereas crossover aims at improving a population by exploiting its strengths. Consider the objectives of both mutation and crossover in traditional GAs. It is possible to make a parallel GA approach more adaptive by looking into a proper balancing of the use of GA operators (Luke & Spector; Spears, 1992). SGA is a suggested approach.

In Chapter 3, Figures 3.3 and 3.7 help demonstrate the connection of such a method in the LCE domain. In Figure 3.3, the impact of parameters on product performance showed that the performance of a system depends on the type of interaction between the design, operational, and environmental parameters. Determining the nature of the relationship between good performance and LCE parameters is a complex task requiring consideration of individual and group characteristics and accounting for interactions between systems. As SGA is now further

described and the suitability of schooling behavior to search heuristics is assessed, efforts will be made to answer the second question.

4.3 SGA Overview

Since SGA mimics the way fish live in community, a summary of schooling dynamics is necessary.

4.3.1 Fish school. A group of fish (shoal) that stay together not only for social reasons, but also for coordinated group swimming are said to be schooling (Bonabeau & Dagorn, 1995). Fish derive many benefits from schooling behavior including defense against predators (through better predator detection and diminished chance of individual capture), and enhanced foraging success. Schooling fish are usually of the same species and tend to have similar size. Schooling fish are capable of undertaking complicated maneuvers (Moyle & Joseph J. Cech, 2004). Fish school is a classic example of emergence, where there are properties that are possessed by the school but not by the individual fish (Parrish, Viscido, & Grunbaum, 2002). The emergent properties give an evolutionary advantage to members of the school, which individual fish do not receive.

Fish schools, ant pheromone trail networks, bird flocks, or aggregation of cockroaches are some typical examples of collective behavior of animals that have been accurately described in terms of individuals following simple sets of rules. Fish schools for instance are known to come in many different shapes and sizes: predator avoiding vacuoles, stationary swarms, flash expansions, herds and balls, hourglasses and vortices; highly aligned cruising parabolas (Parrish et al., 2002; B. L. J. Partridge, 1982; B. L. J. Partridge, Johansson, & Kalisk, 1983), and the principles that give rise to their collective behavior have already been more or less successfully

explained (Ballerini, Cabibbo, Candelier, Cavagna, Cisbani, Giardina, Vincent, et al., 2008; Inada, 2000; Pitcher, 2001; Sumpter, 2006).

4.3.2 Terminology and taxonomies. SGA intends to exploit fish schooling dynamics to enhance metaheuristic search. SGA is not a simulator, as its key concern is not to accurately mimic the explicit actions of schools of fish. SGA is an optimization technique based on an evolutionary algorithm: GA. An SGA fish is a GA string of encoded genes also known as a candidate solution. A local cluster, group, or subpopulation of fish is a school of fish and represents fishes that share some similarities. The domain of definition of an SGA problem is its search domain or the sea. In SGA, food and predators are represented by attractive and unattractive locations in the search domain respectively. SGA incorporates a technique for food depletion to encourage exploration. Figure 4.1 gives an example of food vs. predator for a maximization problem.

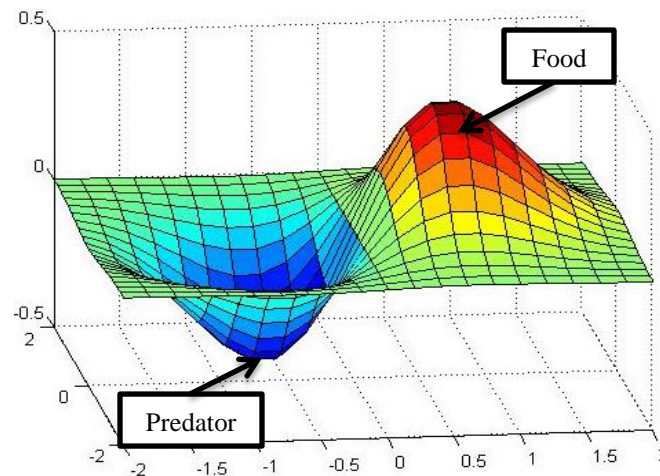


Figure 4.1. Food vs. Predator

Using this terminology, SGA can then be defined as an enhanced metaheuristic search technique in which the fishes discover and eat food, and spawn new fish whenever the conditions are favorable while avoiding, by escaping means, existing predators. SGA does so by modifying

traditional GA to account for schooling. It uses a model's parameters to control the different methods/functions/processes available within the model. As a process mimicking natural methods, SGA then accounts for the way a model's parameter either changes independently, or as a consequence of the variation in another model's parameter. Figure 4.2 shows the taxonomy of search techniques.

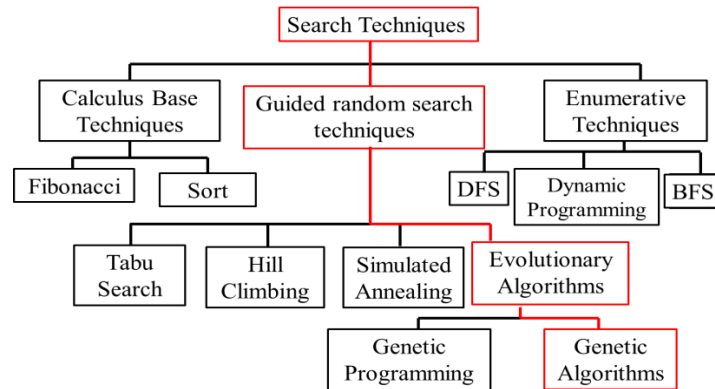


Figure 4.2. Taxonomy of search techniques

In SGA, schools are sensitive to their surroundings. Schools use their perception of the environment to guide the search through the solution domain. SGA has an adaptive parameter control mechanism, meaning that there is some form of feedback from the search that is used to determine the direction and/or magnitude of the change to the strategy parameter. Figure 4.3 shows the taxonomy of parameter setting. SGA falls under the red labeled (adaptive) category.

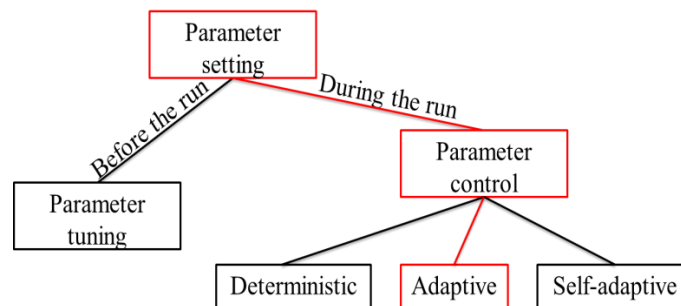


Figure 4.3. Taxonomy of parameter setting in evolutionary algorithms (Michalewicz & Fogel, 2004)

The next section addresses the procedural aspect of SGA. It explains the fish school concepts and details the implementation of those concepts.

4.4 SGA Procedure

The fish in the presented algorithm randomly but collectively “swim” and discover new places to feed. Each time a new feeding ground is found; the school consumes it, and then moves on to look for another (not yet fed on) location to exploit. The act of leaving a location after feeding from it is to prevent any school, for time to time, from being trapped in an optimum (whether local or global) solution. The behavior of depleting the food when feeding is also intended for the schools to explore new areas of the sea in which the global solution may be discovered.

During the schooling process, schools sometimes encounter predators and are forced to escape using various strategies. SGA mimics such a behavior by first defining a predator as any region of the search domain where solutions are relatively worse than the currently obtained best solutions. A predator avoidance mechanism built into each school allows the fish of that school to swiftly escape predators while looking for feeding grounds.

The sea can host many schools at a time. With SGA, within school and between schools’ interactions are behavior based. Applying genetic operators such as mutation and crossover carries out the interactions between fishes and between schools. Due to the dynamic nature of the interactions, the mutation rate (number of offspring to result from the mutation process), the mutation magnitude (defined here as the length of the phenotype to be affected by the mutation process), and the crossover rate (number of offspring to result from the crossover process) change during the run of an SGA algorithm. Their values change according to the size of a school, the overall average fitness of that school, and the relative perception of the local area of

the sea by a fish school. Figure 4.4 shows a side-by-side comparison of traditional GA and SGA. The “Food Foraging”, “Predator Avoidance”, and “School Maintenance” are processes belonging to SGA. “School Division” and “School Formation” are simple collateral effects of the SGA processes. The collateral effects are not built into SGA, but are the consequences of the grouping mechanism built into SGA. The SGA processes are the fish school dynamics SGA uses to enhance GA heuristics.

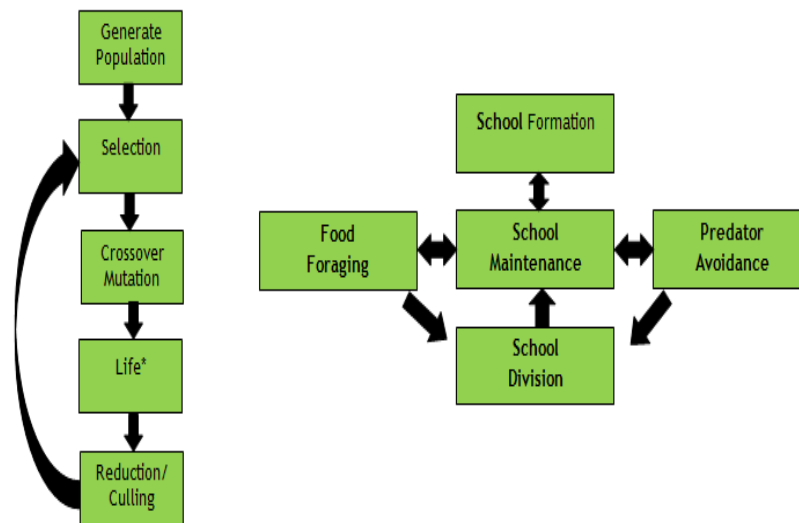


Figure 4.4. Genetic Algorithms vs. Schooling Genetic Algorithms (* often omitted)

4.5 SGA Modeling

The most notable features of SGA are its ability to “naturally” group subpopulations, and to exploit the evolvability characteristics of a population when assessing its performance. Both grouping and evolvability features rely on the perception of the local search space by fish schools.

4.5.1 School merging and splitting. Fish schools have various shapes and sizes, and those shapes and sizes change depending on the environment (proximity of food or predator

etc.). Couzin, Krause, James, Ruxton, and Franks (2002) proposed a model in which individual animals living in communities follow three simple rules of thumb: (1) move away from very nearby neighbors; (2) adopt the same direction as those that are close by and (3) avoid becoming isolated. A number of grouping techniques might be employed. In order to make SGA grouping simple, an unsupervised density based clustering algorithm is created for use in this research. The algorithm has an $O(n^2)$ average runtime. The developed clustering approach named GEMAC (Geometrically Expanded Membership for Automated Clustering), allows SGA to group fish in schools based on: (1) the relative proximity of other fishes to any given fish within the school, and (2) on the relative geometric proximity of shared near neighbors. The density within the aggregations of fish schools is nonhomogeneous, as fishes are packed more tightly at the border than the center of the shoal (Ballerini, Cabibbo, Candelier, Cavagna, Cisbani, Giardina, Orlandi, et al., 2008). The change in density is responsible for the oblong shape frequently observable with fish schools (Hemelrijk & Hildenbrandt, 2008; B. L. Partridge, 1980). The benefit of the oblong shape is considered to be the protection against predators (Hemelrijk & Hildenbrandt, 2008, 2012; Hemelrijk, Hildenbrandt, Reinders, & Stamhuis, 2010). GEMAC was not designed to exactly replicate the density distribution of fish schools, but to mimic the change of density along a cluster.

Let $\{x_1, x_2, \dots, x_N\}$ be a set of data points in an L dimensional Euclidean vector space. It is required that these N points be clustered in K groups (K unknown) where each group must fulfill the requirement of changing density from the head to the tail of the group. Let us consider the subset $\{x_1, x_2, \dots, x_M\}$ as a cluster of points from the previous set. To build such a cluster with GEMAC, points must be added one by one to the cluster, starting with x_1 and ending with x_N .

Let x_i, x_j, x_k , and x_l be four points added to the precedent cluster in that exact order. The distances formed by the pairs (x_i, x_j) , (x_j, x_k) , and (x_k, x_l) should have values that are either the same or decrease in a geometrically scaled down fashion. The highest distance between two consecutively added points to the cluster should not exceed the modal distance calculated between the first vector to the cluster and a randomly sampled set of points within the search domain.

4.5.1.1 Computational aspect of GEMAC. The clustering algorithm using the above grouping concept is carried out in the following manner.

Step 1: Compute an $N \times N$ proximity matrix of the set $\{x_1, x_2, \dots, x_N\}$. This operation has a $O(n^2)$ runtime complexity.

Step 2: Randomly sample a set number of entries from each row of the proximity matrix. Sample the same number of entry from each row, and calculate the row-wise mean of all the samples entries. A mean is the modal distance to be used for the corresponding row. Geometrically scale all modal distances by dividing them by an *a priori* set value higher than 1. Values between two and three seem to yield excellent results.

Step 3: For each vector, starting from the first vector, assign to the same cluster all points located within the calculated modal distance for that vector.

Step 4: For each vector assigned in step 3, use the same modal distance geometrically scaled one more time, and assign to the cluster of the starting vector, all points located within the newly calculated modal distance for that vector.

Step 5: For each vector left unassigned, repeat step 3 and step 4 until a vector either finds a cluster, or is left unassigned to form its own cluster. Step 3 thru step 5 represents an overall worst case complexity of $O(n^2)$.

Rather than using the Euclidean distance, the taxicab distance (Krause, 1987) is used to speed up the algorithm. Further, making GEMAC simple also has the benefit of minimizing SGA overhead beyond the GA.

4.5.1.2 Clustering in action with GEMAC. Figure 4.5 represents the partitioning that was used for testing GEMAC.

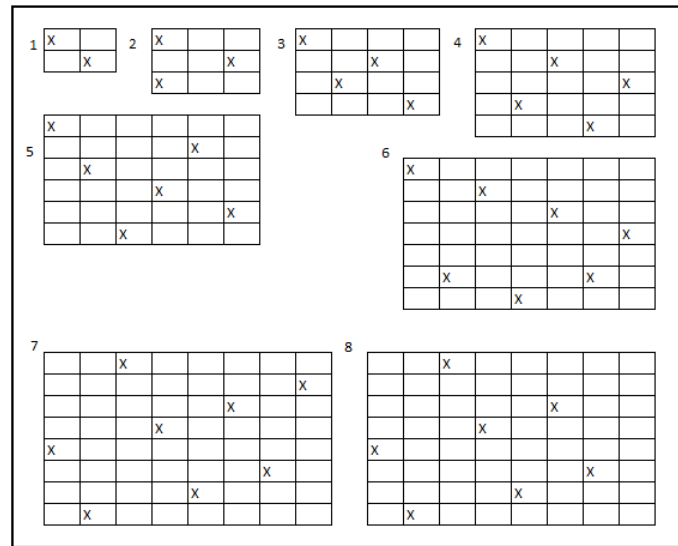


Figure 4.5. Domains with predetermined cluster centers

Two dimensional data with values x_i across each dimension set such that $x_i \in [0, 1000]$. Three test results are reported here. Each test was performed by generating, using a normal distribution ($\mu = \text{cluster}_{center\ coordinate}$, $\sigma = \frac{\text{cluster cell range}}{n}$ where $n = 3, 4, 5$), a number $m \in [50, 100]$ of points around each of the centers (represented with an X in Figure 4.5) of all the domains. The resulting data points for each domain were passed to a GEMAC routine for automated clustering, and the output of GEMAC plotted. In Test 1, points were generated around the designated cluster centers with a standard deviation $\sigma = \frac{\text{cluster cell range}}{3}$. Test 2 (respectively Test 3) used a standard deviation $\sigma = \frac{\text{cluster cell range}}{4}$ (respectively $\sigma = \frac{\text{cluster cell range}}{5}$) to test

how GEMAC would perform with respect to a changing density. 12% of each row was sampled to determine the modal distance as explained in step 2 above. The value $f = 2.7183$ was used as the scaling factor for the modal distances. Figure 4.6 thru 4.8 show the output of GEMAC for Test 1 thru 3.

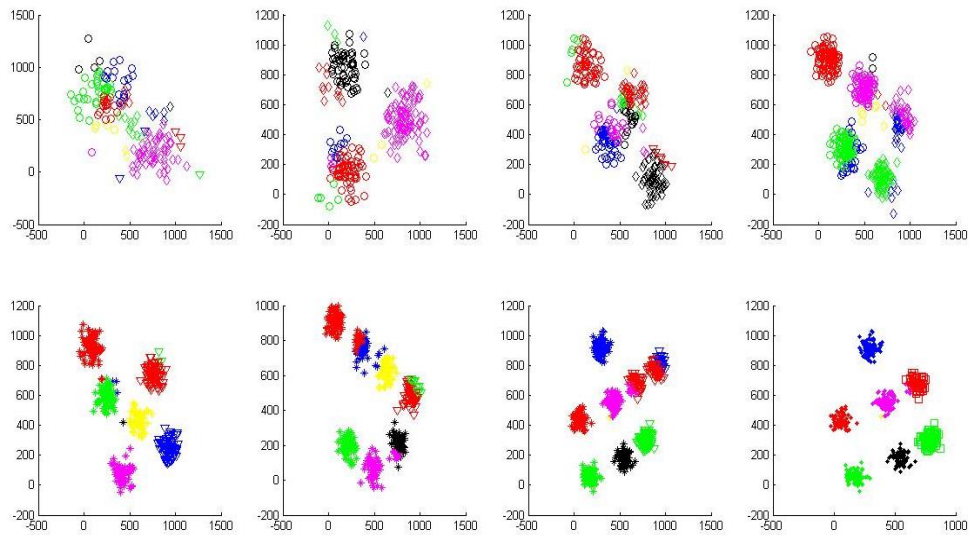


Figure 4.6. GEMAC output for Test 1

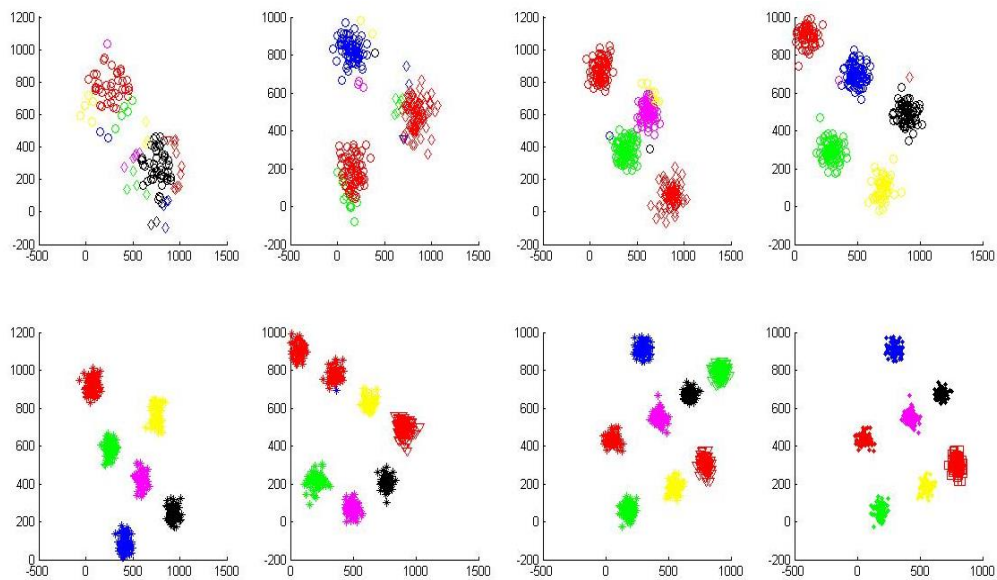


Figure 4.7. GEMAC output for Test 2

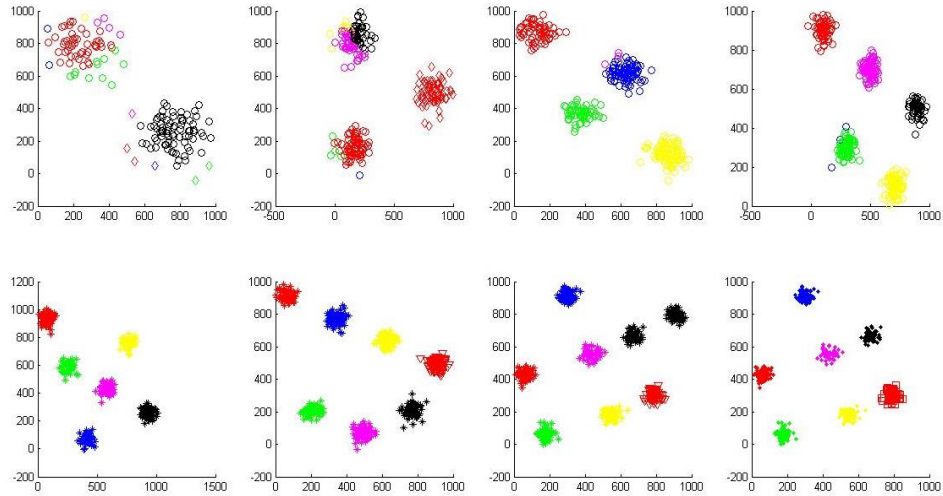


Figure 4.8. GEMAC output for Test 3

Table 4.1 summarizes the results of all eight experiments run to test the GEMAC.

Table 4.1

GEMAC clustering output summary

Number of given cluster centers	Number of cluster found per case and per test		
	Test 1	Test 2	Test 3
2	16	13	10
3	12	9	7
4	13	7	5
5	11	7	6
6	9	6	6
7	8	7	7
8	9	8	7
7	8	7	7

Since the role of GEMAC is solely to group, split and/or merge schools in an unsupervised fashion, no clustering validity checking approaches based on either internal and external criteria is presented or discussed, a simple comparison table is provided. Since GEMAC was created to meet a specific clustering need and a particular problem, the final partitions of the tested data set does not require some sort of comparative evaluation as it is the case in most applications (Milligan & Cooper, 1985; Pal & Biswas, 1997; Ramze Rezaee, Lelieveldt, & Reiber, 1998).

4.5.2 Behavior setup. Figure 4.9 shows a simple view of a school behavior assignment. Using the performance of a school center of mass (P_G), the averaged performance value of all the schools within the sea (P_{Avg}), and some cutoffs (l_{off} , h_{off}), the behavior of any school within a sea can be defined: a low value for P_G with respect to P_{Avg} would be characterized as the presence of a predator. All other fish school behaviors will refer to Figure 4.9 which also has a decision making node (selection statement) called “*Is Food still available?*” that also leads to the predator avoidance mode.

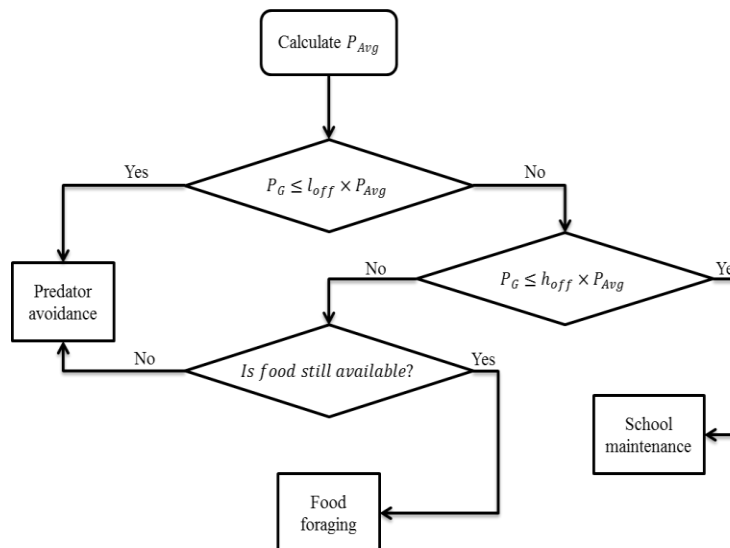


Figure 4.9. Fish school behavior assignment

The “*Is food still available?*” selection statement of Figure 4.9 connects to the predator avoidance in case of no food availability. The purpose of the selection statement is to mimic the depletion of food within any given area of the sea. By translating the time spent (generation wise) by a school of fish in an area of the sea (whether attractive or not) into the food being depleted; any school will be pushed away to explore other possibilities after some time.

4.5.3 Predator avoidance. Predator avoidance occurs when the school enters a dangerous or unattractive area. Fish swiftly move in an attempt to escape the detected predator. Such an operation sometimes results in school division, as the fishes will quickly swim in diverse directions. Figure 4.10 shows how a predator avoidance maneuver is performed. To create the appearance of motion of a school fleeing from a predator, SGA proceeds via two phases. First, some mutation operations are randomly performed on the fish of the school. Then the fish resulting from the mutation are used as the first parents for some crossover operations (during the second phase) with the members of the school. To keep SGA as simple as possible, both the proportion of crossover operations and the number of mutation operations could be equally set to half the proportion of fish allocated to the school by SGA.

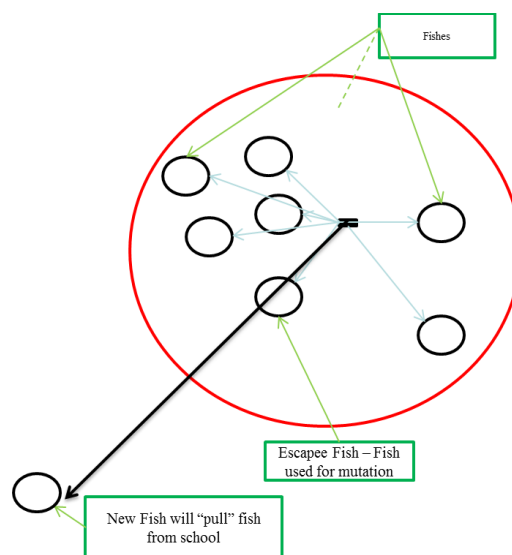


Figure 4.10. Predator avoidance maneuver.

To be able to assess the perception of a predator requires an assumption: the presence of a predator is characterized by a portion of the sea perceived as dangerous/unattractive by a fish school. This means that the fishes might perform worse than expected if performing a depth search within the local area. Therefore, to locally perceive an environment as one that requires a predator avoidance behavior, the SGA uses the fitness of the center of mass (CM) of that school and compares it to the average fitness (sea wise) of all the other schools. The center of mass in return, is assessed as a fitness-weighted combination of all the fish within a school. Proceeding such a way to get a school center of mass makes the center of mass biased towards the strongest fish of the school. The approach is simple and forces any underperforming school to keep exploring, whereas it allows the better performing schools to capitalize by exploiting performance. The center of mass CM of a school of n fish, where fish in the school are represented by data points X_i ($X_i = \langle x_1 \dots x_L \rangle$), and have fitness values f_i should be calculated as:

$$CM = \frac{\sum_i^n X_i \times f_i}{\sum_i^n f_i}$$

The choice of the “escapee” fish, picked during the first phase of the predator avoidance maneuver, is explained below. The farther from the school CM, and the stronger a fish is, the higher is the selection likelihood of that fish as the escapee fish. During the second phase, the second parents are picked probabilistically, proportionally to their fitness values, using a simple fitness proportionate selection. Let p_i represent the probability of selection of fish i as an escapee fish, f_i represent the fitness of fish i within a school of N fish, and d_i the distance of fish i to the center of mass of the school. The probability value of p_i should be calculated as:

$$p_i = \frac{d_i f_i}{\sum_{i=1}^N d_i f_i}$$

During the second phase, the parents are picked probabilistically using a simple fitness proportionate selection.

In reality, a school can either move as one entity further away from a predator or split as it moves away. Since the flee forces the fish to explore new areas of the sea, predator avoidance can result in the split of a school. SGA mimics the predator avoidance mechanism by giving high fitness fish locate on the outside of a school, a higher probability of pulling the school during a predator avoidance maneuver. Since GEMAC operates on a proximity basis for grouping, such an implementation of predator avoidance may yield a school split, which is a desirable intended side effect.

4.5.4 Food foraging. The food foraging behavior occurs when a fish school finds a relative safe or attractive area. Fish foraging assumes that the proximity of food would cause a school to have its average performance higher than the whole sea average performance. Therefore, the food foraging behavior would manifest when a part of the search domain is perceived as attractive by a school.

Looking back at Figure 4.9, a school in food foraging mode is already performing well compared to the other schools within the sea. For that reason, the mutation rate of a school in food foraging mode is low when compared to that school crossover rate. The crossover operator is a process that takes more than one parent (two in SGA) solutions to produce a child solution. Crossover is analogous to reproduction and capitalizes on the strengths/fitness of the parents to generate a hopefully more fit child.

The choice of the fish to use for either mutation or crossover during the food foraging mode is conditioned by the fitness of the fish. The higher the fitness value of a fish is, the higher is the selection likelihood of that fish as a parent for crossover. This is a typical fitness

proportionate situation. Let p_i represent the probability of selection of fish i , and f_i represent the fitness of fish i within a school of N fish. The probability value of p_i should be calculated as:

$$p_i = \frac{f_i}{\sum_{i=1}^N f_i}$$

Although the choice of the fish to use for mutation during the food foraging mode solely depends on the fitness of the fish from the school, the mutation magnitude is not and will be small. The mutation magnitude is defined as the length of the phenotype to be affected by the mutation operator. The purpose of mutation during the food foraging mode is to mimic a fish school spreading over an area to feed. Because of the use of mutation operations to simulate the feeding process, food foraging can end up with a school split. Such a split as a result of foraging is not very likely.

The implementation of the food depletion, as listed on Figure 4.9, relies on the periodic use of the GEMAC clustering and of a recency Tabu list. Performing school clustering periodically with GEMAC enables schools in a given behavioral mode to stay in that mode for a given (short) period of time. The recency-based tabu list can serve many purposes, with one of them providing medium-term knowledge of the search history. During the metaheuristic search process, all known better solutions are stored in a Tabu list. The solutions are updated or removed to the list on a first-entered first-removed basis. While a solution is still in the Tabu list, no school is allowed within a given proximity. Therefore, the Tabu list also guides the search process by guaranteeing that a given part of the search domain, once explored for food by any school, will not be visited again until it is removed from the list. The list size determines the duration of a location in tabu status.

4.5.5 School maintenance. School maintenance occurs when a fish school is neither looking for food, nor trying to escape predators. This behavior, as shown on Figure 4.9, is the default behavior for any school. In school maintenance mode, both GA operators (crossover and mutation) are balanced, as a school in this mode represents a school that is both exploring and exploiting. Unlike the procedure for predator avoidance, school maintenance uses all the fish of the school to sample the candidate solutions for both crossover and mutation. A maintaining school is a wandering school.

When choosing the candidate parents, whether for crossover or mutation, SGA uses roulette-wheel selection (SCX), also known as fitness proportionate selection. SCX allows SGA to select fishes based on their fitness, with the probability of a fish being selected increasing with the fitness of the fish greater or less than its competitor's fitness. In other terms, if f_i is the fitness of fish i , and school i has N fishes, then its probability p_i of being selected among the other fishes of the school is:

$$p_i = \frac{f_i}{\sum_{i=1}^N f_i}$$

When applying either crossover or mutation to a fish school, a proportionate measure is used to guarantee the survival of the more fit schools. The number of crossover and/or mutation operations a school can perform depends on a proportion allocated to the school based of the collective performance of its fish. The higher the number of high fitness fish in a school, the higher its proportional value. Assuming there are N schools within the sea, and that each of those schools i possesses n_i fishes where the performance of fish j is denoted f_j , then the computation of the number C_i to breed by school i will be calculated as follow:

$$C_i = \frac{\sum_{j=1}^{n_i} f_j}{\sum_{i=1}^N \sum_{j=1}^{n_i} f_j}$$

4.5.6 SGA Life Cycle. The SGA life cycle represents the steps required for implementing an SGA algorithm based on the given concepts. The implementation of the life cycle of SGA alternates between groupings and the application of GA operators based on the local perception of the sea by the diverse fish schools.

The initial number of schools could either be set manually and the grouping performed using a clustering algorithm such as the *k*-means algorithm (Hamerly, 2010; Nock & Nielsen, 2006; X. Wu et al., 2008). However, the GEMAC algorithm was built so that anything related to groupings (merge and split) can be unsupervised and nonparametric. Next, all the concepts talked about will be tested for the ability of SGA to converge to a global optimum within an unconstrained domain, and to perceive and avoid deceptive features of the search domain.

Algorithm 1. Schooling Genetic Algorithm High Level Metaheuristic

```

Set the parameters, initialize the population
while termination condition not met do
  Organize fishes in schools
  Set schools' statuses based on performances of fishes
  Foreach school within the sea:
    If school's status is foraging then look for food //Crossover rate > Mutation rate
    If status is predator avoidance then escape //Mutation precedes Crossover
    If status is maintenance then (explore and exploit) //Crossover rate = Mutation rate
  endforeach
  Proceed with reduction to keep overall population size constant
endwhile

```

4.6 Summary

Within this chapter, SGA, a new metaheuristic created to work on problems where grouping, evolvability, and sustainability are characteristics of the solution was developed, explained, and discussed. Increasingly complicated cost functions emanating from complex simulations use life cycle data and could benefit from SGA. Such cost functions are common for engineering design problem.

SGA mimics fish schooling and uses GAs that it extends with some new concepts. SGA develops and implements the concepts of predator avoidance, food foraging, school maintenance, and food depletion/replenishment. SGA uses a new density based clustering approach named GEMAC. GEMAC is used within SGA for all tasks related to grouping and splitting in fish schooling. To finish, a high level view of SGA was given.

Chapter 5

Applying Schooling Genetic Algorithms to Generalized Life cycle Product Design

5.1 Introduction

Schooling genetic algorithms (SGAs) are GA-based metaheuristics built using fish schooling mechanisms, and designed for problems where evolvability, grouping, and sustainability are characteristics of the solution. Generalized life cycle product design (GLPD) is a product life cycle engineering (PLE) based approach for product design that uses life cycle engineering (LCE) data characterization to assist system engineers with their design process. In this chapter, the applicability of SGA to product design, using GLPD, is investigated. Using simulated life cycle engineering data, different case scenarios are envisioned to assess SGA as an analytic tool for better product design.

5.2 Problem Definition

Traditional SE, when designing for the life cycle, not only transforms a need into a system configuration, but also strives to ensure design compatibility with related physical and functional requirements. Traditional SE tends to emphasize design optimization into fixed configuration, along with system decomposition to facilitate system analysis, and the central role of systems engineer for design and sustainment. Such an approach does not capitalize on the ongoing trend of using analogies to biological systems to develop solutions for engineering problems, also called biologically inspired design. Therefore, based on the shift in product nature, there is a need to characterize and extend product life cycle engineering (PLE), to incorporate evolvability (modularity, interoperability, and software level configurability), grouping (system efficiency due to the economy of scale), and sustainability (ability to continuously operate with minimal intervention).

In an attempt to meet and characterize the need of extending PLE, SGA will be applied to the generalized life cycle product design (GLPD) model. Using simulated PLE-data generated for a product, a product's DNA, as displayed in Figure 5.1, was created and was used as an input to an SGA algorithm. The objective was to find out the potential benefits of using SGA and GLPD in the design process of a system/product. Within the designed SGA test bed created for this chapter, values for n , m , and p representing the number of genes to encode the design, operational, and environmental parameters respectively, varied from 0 to 2.

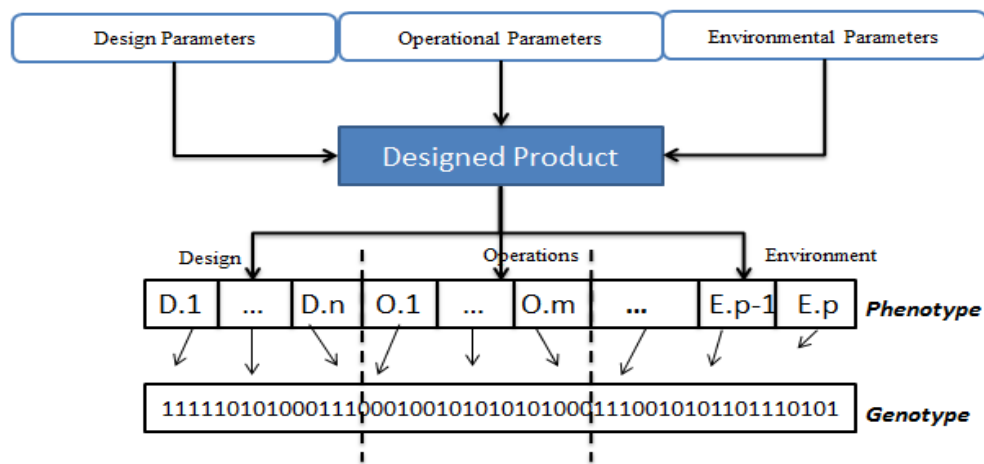


Figure 5.1. Transformation of a designed product into a GA entity

Being able to assess the effects of grouping on performance when running SGA is important. Such an assessment assists in interpreting the solutions returned by SGA, and possibly helps objectively mapping the fitness values of the final solutions. A solution quality assessment indicator was defined as the trait performance indicator (TPI). The TPI was defined such that its value would indicate how important (0.0 to 1.0) any given phenotype is to the observed performance values. Considering all the final solutions returned by the SGA implementation, the wider the spread over the range of permissible values for a given phenotype, the higher the importance of that phenotype. Let BP be the set of all the final solutions returned by the SGA, and P^j any element of BP . P^j is a solution candidate. Let P_i^j designate a trait i of the solution P^j .

Ideally, each trait has a range of values (constraints on trait) that fall between a minimum and a maximum. Let E designate the set of minima and maxima for all the traits of the phenotype, with E_i^m / E_i^M representing the minimum/maximum value for trait i . For a phenotype subset of n traits, the TPI values are computed in a way similar to a relative error, giving an indication of how important the selected set of traits is to the observed performance, relatively to the whole phenotype. The TPI value for trait i (TPI_i) is computed as:

$$TPI_i = \frac{\max\{P_i^j \in BP\} - \min\{P_i^j \in BP\}}{E_i^M - E_i^m}$$

The TPI values account for all the solutions returned by the SGA. TPI values should inform the systems engineer about a stricter constraint that can be imposed on a given parameter without sacrificing the system overall performance. The possible implications of a TPI value are as follow:

- The corresponding or involved component of the system/product can be changed to another less accurate, sensitive, or performing without hurting the system performance.
- The corresponding or involved attribute can be pushed from design to operational or vice versa, without hurting the system performance.

5.3 Applying SGA to GLPD

When applying SGA to GLPD, the SGA concepts take the following meanings within the LCE context:

- A fish is an instance of a product/system configuration represented by the life cycle parameters that will yield the predicted performance if designed.
- A school is a set of product/system configurations sharing some similarities

- A school center of mass (CM) is the central configuration for a cluster of systems, that is, it represents the average best configuration of a set of given configurations. Unlike a fish's performance value that only informs about the performance of one product, a CM on the other hand should inform about the average best of a set of configurations.
- A SGA's tabu list represents the set of best solutions found.
- A TPI value provides further insights into the SGA's group performance values that were recorded during the search process into the SGA's tabu list. Assuming that the SGA is able to return the absolute best solution for a problem, the TPI values would help the designer capture the lower and upper specification limits of the LCE parameters involved.

Using probability distributions, constrained continuous random values were generated that represent the design, operational of a hypothetical product. Considering the immutable character of environmental parameters, their values were set (discrete values) and for any given product, could not change during the simulation. The proportion of the number of products available for each environment over the number of products that is available overall was set and kept constant, via population reduction, throughout each simulation. Simulations were carried using one or more parameters of each LCE type. LCE data was generated to test GLPD with SGA for the following five scenarios (explained in following sections):

- Environment driving design for performance
- Environment driving both design and operation for performance
- Environment and design both driving operations for performance
- Environment and design only driving performance
- Environment, design, and operations all driving performance

5.4 Experimental Design

For the experiments/simulations, each candidate solution had n design genes/parameters, m operational genes/parameters, and p environmental genes/parameters. Therefore, each candidate solution was an N -dimensional data point of the search space, where $N = n + m + p$. Each dimension or independent product performance parameter was coded with a 10-bit string value (actual decimal values falling within the range of -500 to 500). The size of the population was set and maintained at 120 fish. The proportion for each environment was 55% for environment A, and 45% for environment B. The stopping criteria was set to be either the maximum number of generations (set to 1000), or the lack of improvement of the average fitness value of the population for three consecutive generations, whichever came first. Each experiment was repeated three times and the recorded assessment criteria were:

- the number of generations it took for SGA to converge (if convergence occurs)
- the distribution of the behavioral states
- the best solutions obtained
- the best school, and
- the TPI values of each dimension (allele) of the product.

GEMAC was used to handle all SGA grouping operations. Each simulation started with a population of fish randomly (uniform distribution) distributed across the search domain (design and operational parameters). This step was followed by the environmental parameter value(s) being assigned, using a uniform distribution. This process was adopted to allow schools to form in various proportions from each environment, since that GEMAC, the non-parametric clustering algorithm that was used for school formation, is a proximity-based clustering method.

Table 5.1 shows the factors, assessment criteria, and methods that were used to assess the results of the experiments. As far as the relationship is concerned (first factor), five were tested as explained within the previous section. The number of parameters/alleles ranged from three to six. Besides the simple case of a linear and polynomial underlying shape (not listed within the table), the Ackley, Griewank, and Schwefel functions were used to characterize the underlying shape of the search domain. On the assessment criteria side, the number of generations to convergence, along with the TPI values, the best schools and distributions of behavioral states were used. For some experiments, SGA performance was also compared to island GA (IGA) and parallel GA (PGA).

Table 5.1

Factors, assessment criteria, and methods for SGA.

Factors	Assessment criteria
Relationship { <i>LCE Characterization</i> }	Number of generations to convergence
Size {Number of parameters/alleles }	Quality of solutions obtained
Underlying shape {Ackley, Griewank, Linear, Polynomial, Schwefel }	TPI values for each LCE parameter type
	Best schools
	Distribution of behavioral states
Methods	
Schooling Genetic Algorithms (SGA) vs. {PGA, IGA}	

5.4.1 Environment driving design for performance. The objective of the experiments carried here was to determine the ability of SGA to effectively use grouping to converge on separate environments, and to check the ability of SGA to characterize the life cycle parameters' relationship. The relation used for the experiments was linear as follow:

$$f([D_1, O_1, E_1]) = \begin{cases} |1 - |\frac{D_1}{max_{D_1}} - 0.7|| & \text{if } E_1 \text{ is A} \\ |1 - |\frac{D_1}{max_{D_1}} - 0.3|| & \text{otherwise} \end{cases} ; -500 \leq D_1, O_1 \leq 500$$

Using a linear relationship, the expected characterization is vertical lines within the (D, O) plane. The lines should contain the best solutions of the problem. This test is for the case of a product with a simple (linear) performance function that depends on just two life cycle attributes.

5.4.2 Environment driving both design and operations for performance. The objective of the experiments carried here was to determine the ability of SGA to effectively use grouping to converge on separate environments, and to check the ability of SGA to characterize the life cycle parameters' relationship. The relation used for the experiments was polynomial as follow:

$$f([D_1, O_1, E_1]) = \begin{cases} 1 - ||\frac{D_1}{max_{D_1}}| - 0.7| \times ||\frac{O_1}{max_{O_1}}| - 0.3| & \text{if } E_1 \text{ is A} \\ 1 - ||\frac{D_1}{max_{D_1}}| - 0.3| \times ||\frac{O_1}{max_{O_1}}| - 0.7| & \text{otherwise} \end{cases} ;$$

$$-500 \leq D_1, O_1 \leq 500$$

Using a polynomial relationship as described above, the expected characterization is interceptions of vertical and horizontal lines within the (D, O) plane. The best solutions of the problem should be located around the interception points of the lines. This test is for the case of a product with a performance function that depends on three life cycle attributes.

5.4.3 Environment and design both driving operations for performance. The objective of the experiments carried here was to determine the ability of SGA to effectively use grouping to

converge on separate environments, and to check the ability of SGA to characterize the life cycle parameters' relationship. The relation used for the experiments was linear as follow:

$$f([D_1, O_1, E_1]) = \begin{cases} 1 - \left| \frac{O_1}{\max_{D_1}} - 0.7 \right|; E_1 \text{ is A, } D_1 < 0 \\ 1 - \left| \frac{O_1}{\max_{D_1}} - 0.3 \right|; E_1 \text{ is A, } D_1 \geq 0 \\ 1 - \left| \frac{O_1}{\max_{D_1}} - 0.9 \right|; E_1 \text{ is B, } D_1 < 0 \\ 1 - \left| \frac{O_1}{\max_{D_1}} - 0.1 \right|; E_1 \text{ is B, } D_1 \geq 0 \end{cases}$$

$$-500 \leq D_1, O_1 \leq 500$$

Using a linear relationship, the expected characterization is vertical lines within the (D, O) plane. However, the expected characterizing lines should only occupy half of the plane and should contain the best solutions of the problem. This test is for the case of a product with a performance function that depends on three life cycle attributes and that has two main modes of operations depending on the environment where the product is being used.

5.4.4 Environment and the design both driving performance. The objective of the experiments described in this section was to not only determine the ability of SGA to effectively use grouping to converge on separate environments, but also to check the ability of SGA to operate on “noisy” life cycle data. Similar to the previous experiments, the solution representation was picked to have three parameters, one of each kind. Two different sets of experiments were performed. They differed in the function that was used to represent the LCE relationship. This test was for the case of a product with a performance function that depends on many life cycle attributes and that has multiple modes of operations depending on the environment where the product is being used.

The Griewank function was used for the first experiment, and the Schwefel function was used for the second experiment. The Griewank function (Griewank, 1981) is a standard test functions for unconstrained global optimization, and it is used to test the convergence of optimization functions. It has many widespread local minima regularly distributed, to act as attractor to deceive the search process. The function that was used was a modified Griewank function to make the problem a maximization problem. The function is defined by:

$$f([D_1, O_1, E_1]) = -\left[\frac{D_1^2 + E_1^2}{4000} - \cos(D_1) \times \cos\left(\frac{E_1}{\sqrt{2}}\right) + 1\right];$$

$$-500 \leq D_1, O_1 \leq 500$$

For $x_i \in [-100, 100]$, Figure 5.2 shows the typical Griewank function plotted for $n=2$.

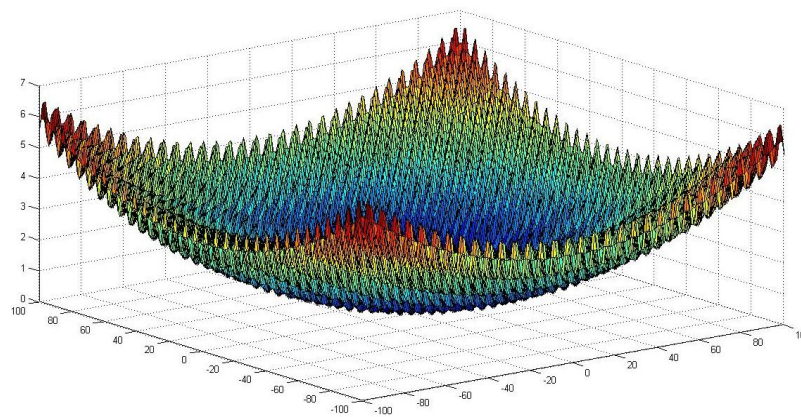


Figure 5.2. Griewank function in $[-100, 100]$ plotted using Matlab

The Schwefel function on the other hand is a deceptive function that has its global minimum geometrically distant, over the parameter space, from the next best local minima. Therefore, with the Schwefel function, the search algorithms are potentially prone to convergence in the wrong direction. The function that was used was a modified Schwefel function to make the problem a maximization problem. The function is defined by:

$$f([D_1, O_1, E_1]) = -[418.9829n - \sum_{i=1}^n |x_i| \times \sin(\sqrt{|x_i|})]$$

$$-500 \leq D_i, O_i \leq 500; 1 \leq i \leq 2; x_1 = D_1, x_2 = O_1, x_3 = E_1$$

For $x_i \in [-100, 100]$, Figure 5.3 shows a plot of the Schwefel's function.

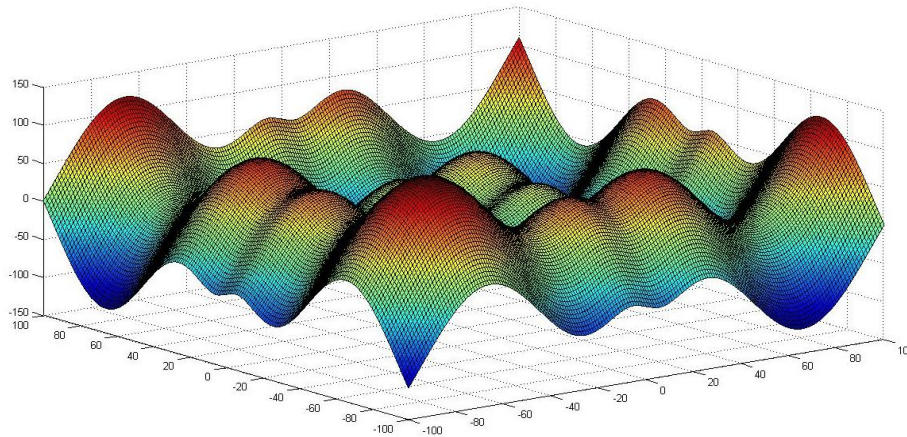


Figure 5.3. Schwefel function in $[-100, 100]$ plotted using Matlab

5.4.5 Environment, design and operations driving performance. The objective of the experiments was to determine the ability of SGA to effectively use grouping to converge on separate environments, and to check the ability of SGA to operate on “noisy” life cycle data. Unlike the previous experiments and setups, two experiments were carried here, each with a different representation of a solution. This test was for the case of a product with a performance function that depends on many life cycle attributes and that has multiple modes of operations depending on the environment where the product is being used. The supposed product required five parameters/genes for encoding: one for environment, and two of each of the other kind. The Ackley function was used to model the LCE relationship. The Ackley's function is a widely used multimodal test function. It has the following definition:

$$f([D_1, D_2, O_1, O_2, E_1]) = -[-20 \times e^{-.2 \sqrt{\frac{1}{n} \times \sum_{i=1}^n x_i}} - e^{\sqrt{\frac{1}{n} \times \sum_{i=1}^n \cos(2\pi x_i)}} + 20 + e^1]$$

$$-500 \leq D_i, O_i \leq 500; 1 \leq i \leq 2; x_1 = D_1, x_2 = D_2, x_3 = O_1, x_4 = O_2$$

For $x_i \in [-25, 25]$, Figure 5.4 shows a plot of the Ackley's function.

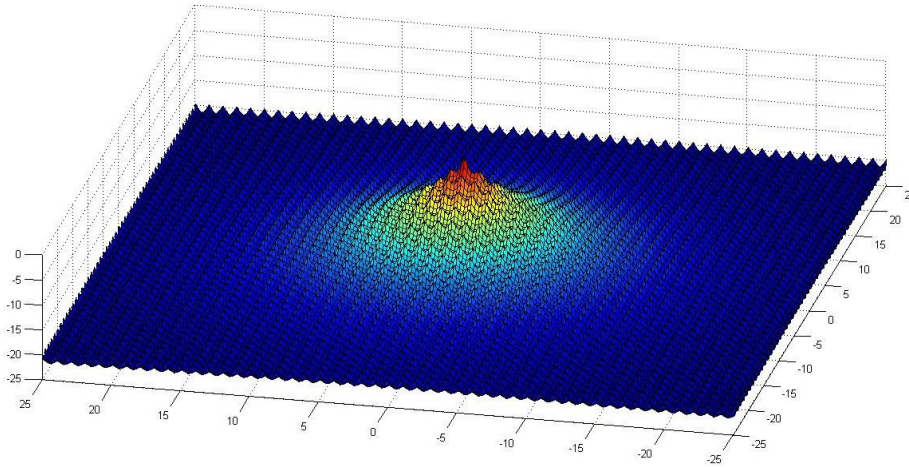


Figure 5.4. Ackley's function in $[-25, 25]$ plotted using Matlab

After testing with the Ackley's function, an extra parameter was added to the product definition to give it six performance parameters (two for each kind). This time, a modified Schwefel function was used to model the complex multidimensional LCE relationship. The modified Schwefel function that was used is defined by:

$$f([D_1, D_2, O_1, O_2, E_1, E_2]) = -[418.9829n - \sum_{i=1}^n |x_i| \times \sin(\sqrt{|x_i|})]$$

$$-500 \leq D_i, O_i \leq 500; 1 \leq i \leq 2; x_1 = D_1, x_2 = D_2$$

$$x_3 = O_1, x_4 = O_2, x_5 = E_1, x_6 = E_2$$

5.5 Results and Interpretation

For each set of experiments that were carried out, results were collected and an interpretation provided. The following subsections show the typical results for each set of experiment.

5.5.1 Environment driving design for performance. Table 5.2 represents a sample of the initial population after a first grouping using GEMAC. There were 22 schools total.

Table 5.2

Sample of initial population for experiment 1

Design	Operational	Environmental	School ID	Fitness
3.805499	-22.01624352	B	1	-1.791211
-30.3083	-20.23126913	B	2	-0.782837
-15.6684	5.927718548	A	3	-0.618067
-4.18381	-10.30136624	A	1	-0.785097
49.54511	-4.415981823	B	4	-0.86212
-28.8373	19.94448208	B	5	-2.05356
26.28639	35.06889827	B	6	-0.767593
10.33888	-6.631614192	A	1	-0.75937
-7.83431	10.97721849	A	3	-1.033247
-2.90412	14.73992981	B	7	-1.974045
-49.2247	-20.66500862	B	2	-1.100193
6.438105	9.207032362	A	8	-1.466603
-27.5468	20.13691809	A	5	-0.860644
-3.15706	3.092316367	A	3	-0.558884
11.26315	57.94968662	B	9	-0.767323
-9.65659	-24.69150235	B	1	-1.996565
11.7271	-58.62095676	A	10	-1.345763
-61.5013	30.50116642	A	11	-2.0623
81.21945	-2.676410108	A	12	-3.063364

Plotting the contents of the experiment's tabu list showed the SGA was able to capture the nature of the relationship between the LCE parameters. Figure 5.5 shows the plot of the tabu list contents. The best performance for environment A (red dotted line) and environment B (blue

dotted line) are both plotted. The contents of the tabu list tend to follow the line that the plot of the performance within (D, O) plan would return for each environment. The observed trend gives an idea of the embedded relationship between performance and LCE parameters.

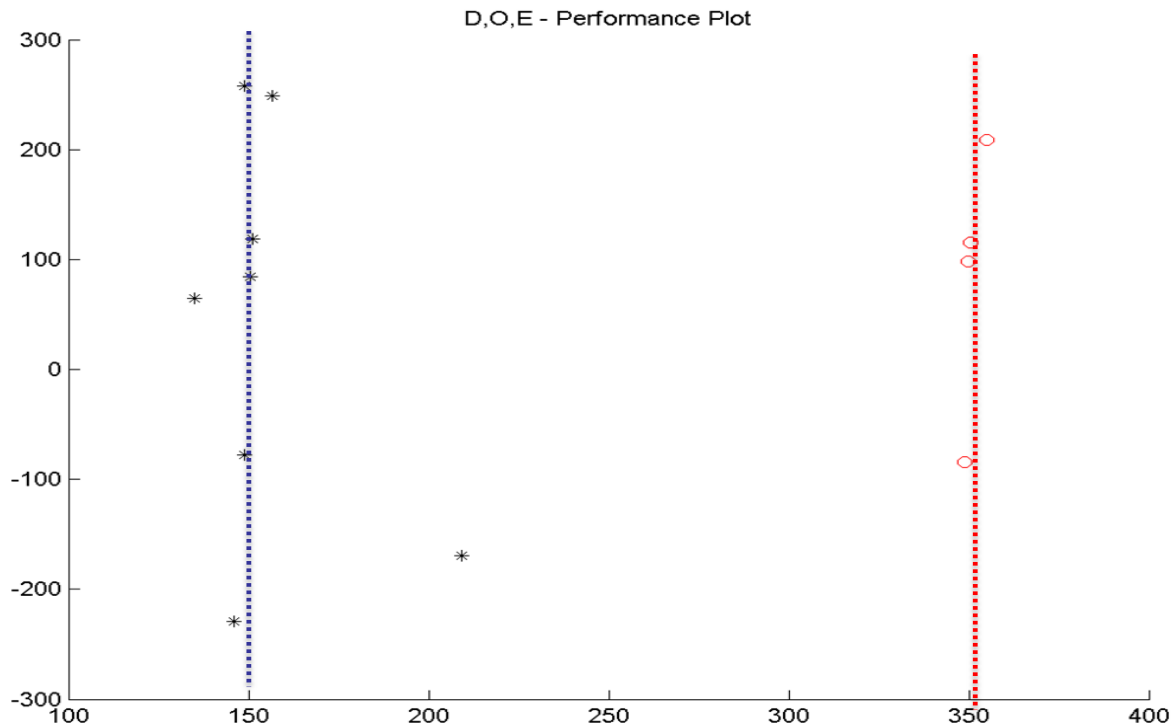


Figure 5.5. Tabu list contents plot for experiment set 1

Table 5.3 shows the contents of the SGA's tabu list for experiment 1. The tabu list is recency based with elements representing the best feeding locations recorded by SGA during the search process. Along with the locations, are listed the composition, environment wise, of the number of fish that made that school. The CM values represent the average best configurations available to the designer.

Table 5.3

Tabu list contents for experiment 1

Center of mass from tabu List			Tabu Performance	# fish from A	# fish from B	Total
Design	Operational	Environmental				
209.2005	-169.253392	A	0.881598961	1	0	1
156.6719	248.6509379	A	0.986656224	3	1	4
146.0206	-229.1916984	A	0.992041137	4	1	5
135.1326	64.83714354	A	0.970265136	19	7	26
148.7408	257.8771119	A	0.997481554	1	0	1
151.2075	118.6229351	A	0.99758492	1	0	1
148.7791	-77.94622939	A	0.997558292	43	0	43
150.5193	84.01986896	A	0.998961384	8	0	8
350.6195	115.1523153	B	0.998761058	0	9	9
354.9097	209.0342458	B	0.990180656	0	2	2
348.7395	-84.68739403	B	0.99747906	0	51	51
349.7957	98.50342012	B	0.999591313	0	19	19

The contents of the tabu list were used to calculate the TPI. The TPI value for the design parameter (0.2523) indicates that the recorded best performance was achieved for values of the design parameter occupying just 25.23% of the available range [-500, 500]. This means that the best performances recorded can still be achieved, even if the designers was to only consider that small range of values for the design parameters. The TPI value for the operational parameter was 0.8025, indicating a wider coverage of the permissible values by the operational parameters. The TPI value for the environment should always be 1 when the performance is to some extent environment dependent. Unlike Table 5.4 that shows a sample of the final population, Table 5.3 has more information because of the convergence of SGA.

Table 5.4

Sample of final population for experiment 1

Population				
Design	Operational	Environmental	School ID	Performance
350	-85.75016356	B	1	1
350	-85.75016356	B	1	1
350	-85.75016356	B	1	1
150	-79.03928056	A	2	1
150	-79.03928053	A	2	1
150	-79.03928051	A	2	1

Figure 5.6 shows a plot of the final results for one of the experiments. From Figure 5.6, it can be seen that as the population's average fitness improves, the number of schools decreases to a final count of 2, matching the number of environments.

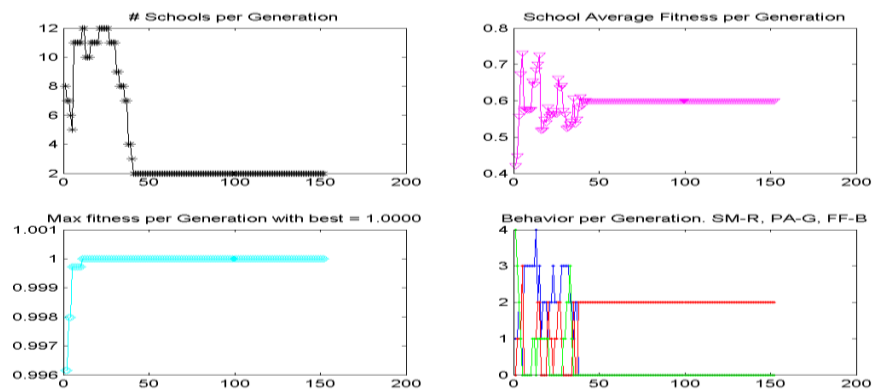


Figure 5.6. SGA results per generation for experiment 1

The lower right graph represents the number of schools that were on a given behavioral mode during a given generation. School maintenance (SM) is in red (R), predator avoidance (PA) is in green (G), and food foraging (FF) is in blue (B). Although all behavioral modes were in used early during the simulation, by the time the final count of 2 is reach, the improvements

per generation to the population's average fitness become too small and school maintenance become the predominant behavior for the schools. Table 5.2 shows the contents of the SGA's tabu list for experiment 1. Along with the locations, are listed the composition, environment wise, of the number of fish that made that school.

5.5.2 Environment driving both design and operations for performance. The simulation started with 120 fish initially grouped in 23 schools using GEMAC. Table 5.5 below represents a sample of the initial population.

Table 5.5

Sample of initial population for experiment 2

Design	Operational	Environmental	School ID	Fitness
41.45297	-13.7693545	A	1	-1.068152565
1.105899	22.2292995	B	2	-0.551974849
4.139539	7.56370167	B	2	-1.546313486
9.176969	-7.13024429	A	3	-0.591220148
33.95105	-5.21207269	A	1	-0.9252308
-31.0792	4.32645342	A	8	-1.67771339
-37.2721	-3.80320293	A	8	-1.768311975
-45.6622	-3.00839178	B	7	-1.630178013

As a contrast to Table 5.5, Table 5.6 displaying a sample of the final population has much less diversity. This is a result of the convergence of SGA.

Table 5.6

Sample of final population

Population				
Design	Operational	Environmental	School ID	Performance
-350	-150.408	B	1	1
-350	-150.408	B	1	1
-350	-150.408	B	1	1
-150	6.0132	A	2	1
-150	6.0132	A	2	1
-150	6.0132	A	2	1

The convergence of SGA is also observable from Figure 5.7 that shows a plot of the final results for one of the experiments. From Figure 5.7, it can also be seen that as the population's average fitness improves, the number of schools decreases to a final count of 2. The SM mode is the only school behavior that was used during the simulation by all schools. This could be explained by the nature of LCE parameters relationship that was used. An LCE relationship that changes linearly or smoothly will make effective perception of the environment hard to be achieved, causing the default behavior of fish to be used most of the time.

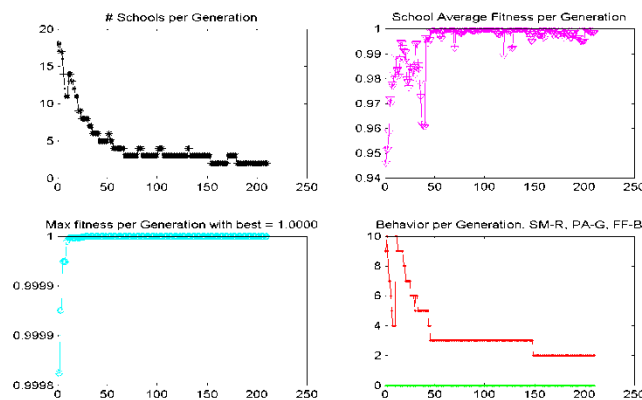


Figure 5.7. SGA results per generation for experiment set 2

Table 5.7 shows the contents of the tabu list at the end of the simulation. The tabu list is recency based with elements representing the best feeding locations recorded by SGA during the search process. Along with the locations, are listed the composition, school and environment.

Table 5.7

Tabu list contents for experiment 2

Center of mass from tabu List			Tabu	# fish	# fish	Total
Design	Operational	Environmental	Performance	from A	from B	
357.1424	220.0255	B	0.997999	0	0	2
157.1135	276.0899	A	0.997897	1	0	1
316.0363	146.6837	B	0.999549	0	3	3
354.0654	141.5273	B	0.999862	0	1	1
-152.593	-212.36	A	0.998572	7	0	7
189.1089	361.0956	A	0.998264	1	0	1
-349.138	150.9856	B	0.999997	0	15	15
-322.118	-150.408	B	0.999955	0	6	6
-342.429	-150.408	B	0.999988	0	27	27
347.8761	-150.408	B	0.999997	0	7	7
150.3152	-291.122	A	0.999926	26	0	26
-150.086	90.7473	A	0.999911	25	0	25
-149.999	6	A	0.999999	10	0	10
-350	0	B	1	0	1	1

The contents of the tabu list represent the average best solutions gathered by the SGA during the process. The values of the tabu list were used to calculate the TPI. The interpretation of those values is as follow: the TPI for the operational and the design parameters (0.6522 and 0.7071 respectively) indicates that the recorded best performance was achieved for values of the operational and the design parameter occupying just 65.22% and 70.71% respectively of the

available range [-500, 500]. Considering the relation used for LCE parameters, a TPI value of 75% $\left(\frac{350-(-350)}{500-(-500)}\right)$ or lesser was expected that would capture the lower and upper specification limits of each LCE parameter.

Plotting the contents of the experiment's tabu list showed that the tabu list was able to capture the nature of the relationship between the LCE parameters. Figure 5.8 shows the plot. The best performance for environment A and environment B are both plotted: red dotted line and blue dotted line respectively. The contents of the Tabu list tend to follow the lines and line intersections for each environment, giving an idea of the performance trend hidden within the life cycle data.

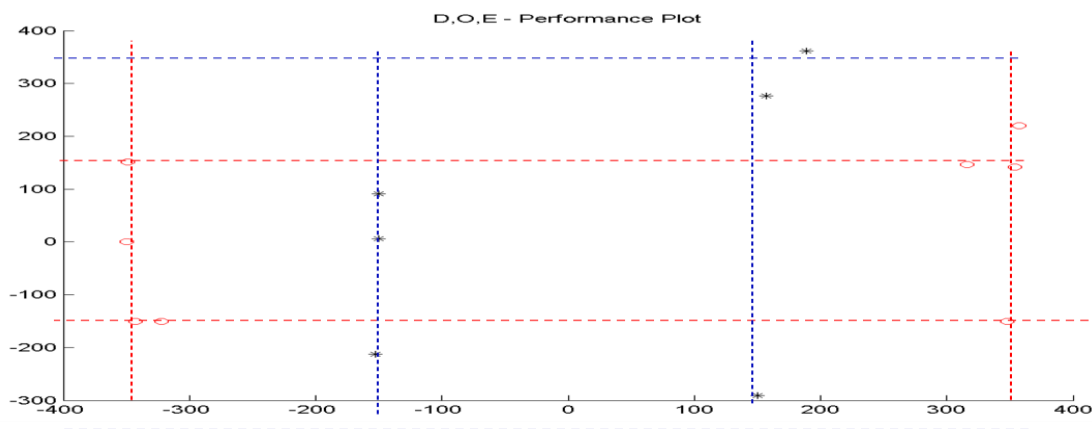


Figure 5.8. Tabu list contents plot for experiment set 2

5.5.3 Environment and design both driving operations for performance. Figure 5.9 shows a plot of the final results for one of the experiments. From Figure 5.9, it can be seen that as the population's average fitness improves, the number of schools decreases to a final count of 3. The SM mode dominated during most of the simulation, with PA being the first behavior to stop being used after a couple of generations.

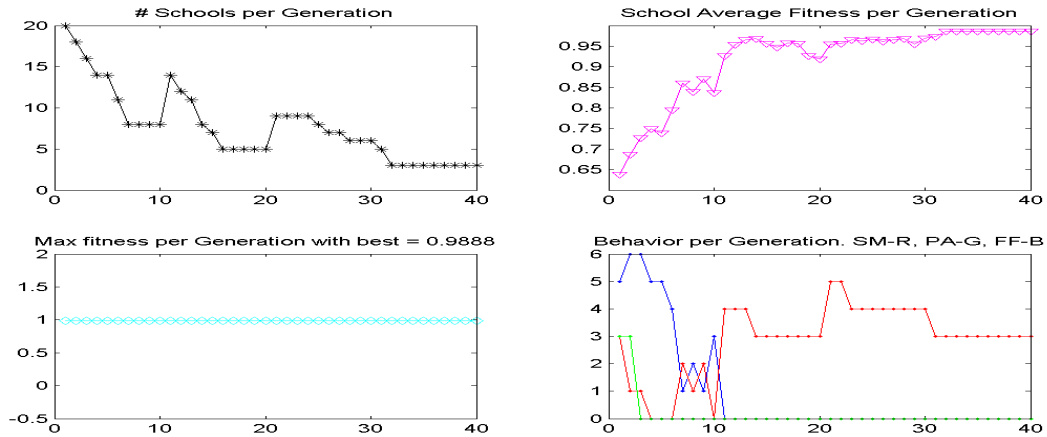


Figure 5.9. SGA results per generation for experiment set 3

Table 5.8 shows the contents of the SGA's tabu list for experiment 3. As a reminder, the tabu list is a recency based tabu list whose elements represent the best feeding locations recorded by SGA during the search process. Along with the locations are listed the composition, environment wise, of the number of fish that made that school.

The contents of the tabu list were used to calculate the TPI. The interpretation of the TPI values is straightforward. The TPI for the operational and the design parameters (0.7751 and 0.7021 respectively) indicates that the recorded best performance was achieved for values of the operational and the design parameter occupying up to 77.51% and 70.21% respectively of the available range [-500, 500].

Plotting the contents of the experiment's tabu list showed the tabu list was able to capture the nature of the relationship between the LCE parameters. Figure 5.10 shows the plot. The best performance for environment A (red dotted line) and environment B (blue dotted line) are both plotted. The contents of the tabu list tend to follow the half-lines for each environment, giving an idea of the performance trend hidden within the life cycle data.

Table 5.8

Tabu list contents for experiment set 3

Center of mass from tabu List			Tabu	# fish	# fish	Total
Design	Operational	Environmental	Performance	from A	from B	
59.29144	-190.786	B	0.918429	0	2	2
60.55729	-51.0816	A	0.997837	4	3	7
159.3046	50.94962	A	0.998101	2	0	2
50.0277	157.3688	B	0.985262	0	2	2
-271.822	-438.411	A	0.976821	1	0	1
65.7312	-166.77	B	0.96646	0	5	5
34.33504	-133.416	B	0.966832	0	5	5
262.0071	-55.5769	A	0.988846	1	0	1
406.3256	-131.402	B	0.962804	0	1	1
-75.1225	336.6956	B	0.973391	0	15	15
170.409	168.5702	B	0.96286	0	35	35
205.8378	167.9305	B	0.964139	0	31	31
430.2609	-55.5769	A	0.988846	11	0	11
414.0749	-55.5769	A	0.988846	14	0	14
394.8999	-55.5769	A	0.988846	9	0	9

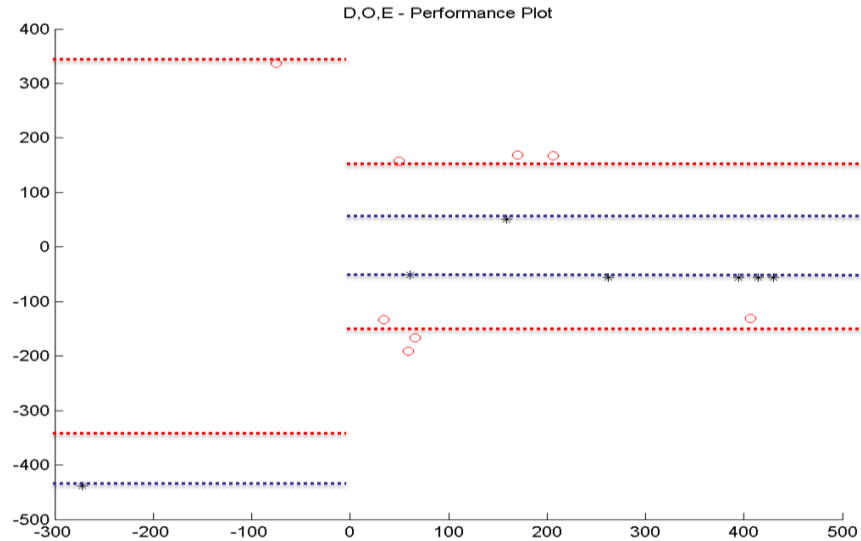


Figure 5.10. Tabu list contents plot for experiment set 3

So far, the plotting of the tabu list has succeeded in giving a trend of the underlying LCE parameters relationship. The success has been observed with both linear and polynomial relationship characterizing the LCE parameters. The setup of school behavior, based on the perception of the environment by fish schools, appears to work as expected: SGA is able to use grouping to prevent premature convergence of a population while learning relationships within the data.

5.5.4 Environment and the design both driving performance. As mentioned during the setup phase, two sets of experiments were carried out for this case.

5.5.4.1 Using Griewank to characterize LCE's relationship. The Griewank function was modified to turn the problem into a maximization problem with global best performance value being 0. The simulation started with 27 schools, and ended with 2 schools. Figure 5.11 shows a plot of the final results for one of the experiments.

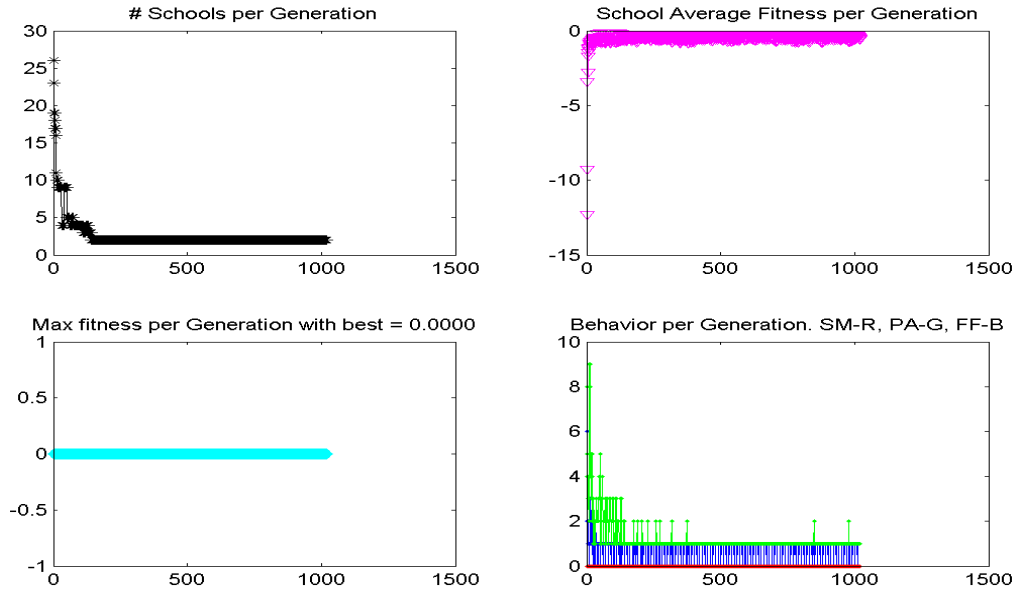


Figure 5.11. SGA results per generation for experiment set 4 – Griewank

From Figure 5.11, it can be seen that as the population's average fitness improves, the number of schools decreases to a final count of 2. The PA and FF modes of behavior dominated during the whole simulation. Although such domination was expected, its underlying cause cannot be determined with 100% confidence. The assignment of a school behavior based on the relation between the school CM's fitness and the current global average rather than past school history, or the regularly spaced sinusoidal-shaped subdomains making up the entire search domain are two possible reasons. With the later reason, the subdomains created by the Griewank function would then cause fish schools to perceive the environment predominantly as either food or predator. Table 5.9 shows the contents of the SGA's tabu list for experiment set 4. Along with the locations are listed the composition, school, and environment.

Table 5.9

Tabu list contents for experiment set 4 – Griewank

Center of mass from tabu List			Tabu	# fish	# fish	Total
Design	Operational	Environmental	Performance	from A	from B	
31.03806	-16.3634	A	-0.31139	0	0	1
34.53917	-89.5806	B	-0.85465	3	0	3
-22.2395	-61.805	B	-0.69388	2	0	2
44.07288	-24.9635	A	-0.4897	0	2	2
0	-67.032	A	0	0	1	1
0	73.32395	A	0	0	2	2
-0.61877	-89.0321	A	-0.18551	1	17	18
-0.4102	-108.575	A	-0.083	1	9	10
-0.04092	28.67091	A	-0.00084	0	8	8
1.87E-09	220.4507	A	0	0	4	4
0	141.3936	A	0	0	2	2
3.31E-09	115.2158	A	0	0	3	3
4.42E-10	-218.242	A	0	0	3	3
4.74E-09	-21.5506	A	0	0	4	4
0	-262.493	A	0	0	1	1
0	-131.247	A	0	0	2	2
-0.31273	162.7716	A	-0.04853	1	9	10
2.12E-09	45.25208	A	0	0	1	1
3.41E-09	90.50417	A	0	0	1	1

The contents of the tabu list were used to calculate the TPI values. The TPI for the operational and the design parameters (0.9547 and 0.1512 respectively) indicates that the recorded best performance was achieved for values of the operational and the design parameter occupying up to 95.47% and 15.12% respectively of the available range [-500, 500]. The big difference was not expected, considering the shape and the symmetry of the Griewank space.

Finally, plotting the contents of the experiment's Tabu list revealed a peculiar median line that happens to be one of the lines of local minima of the Griewank function. The nature of the relationship between the LCE parameters was more complex in this experiment than within the previous experiments. Figure 5.12 shows the plot. The best performance for environment A (red circle) and environment B (black stars).

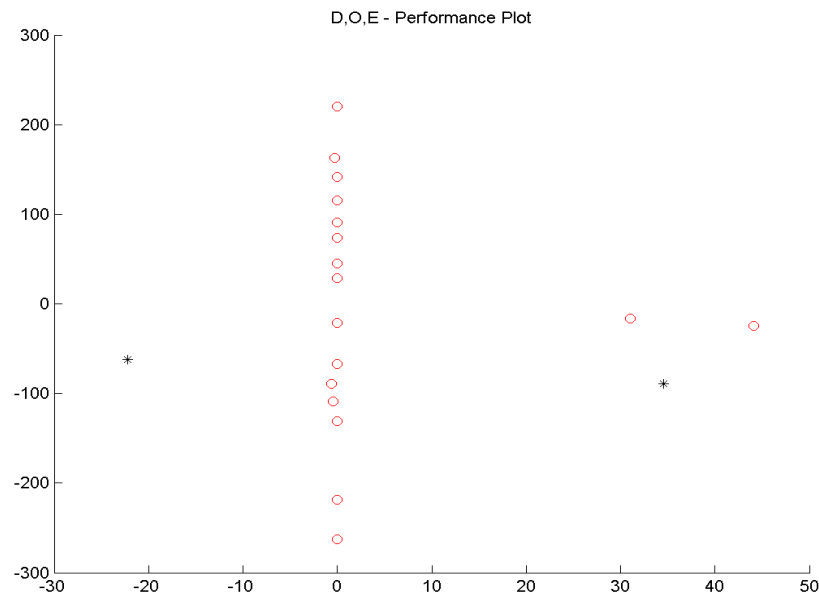


Figure 5.12. Tabu list contents plot for experiment set 4 – Griewank

No direct explanation can be given to explain the line trend observable from Figure 5.12. The same pattern was observed for three repetitions of the same simulations.

5.5.4.2 Using Schwefel to characterize LCE's relationship. The Schwefel function was modified to turn the problem into a maximization problem with global best performance value being 0. Figure 5.13 shows a plot of the final results for one of the experiments. From Figure 5.13, it can be seen that although the simulation started with 9 schools, as the population's average fitness improved, the number of schools decreases to a final count of 2 matching the number of environments. The PA mode of behavior dominated during the whole simulation. Such a behavior was not expected. The Schwefel function, like the Griewank function, creates deceptive attractors all over the search domain. However, both PA and FF are not used the same way in this experiment as they were in the previous experiment.

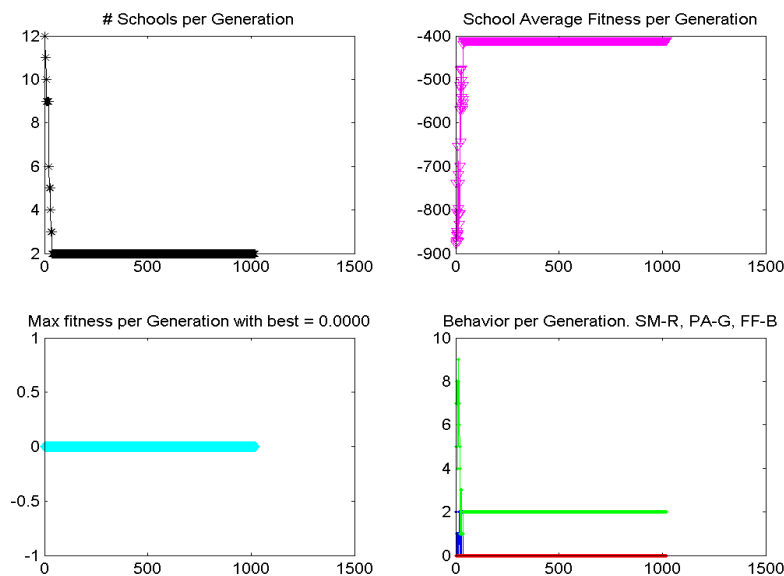


Figure 5.13. SGA results per generation for experiment set 4 – Schwefel

Table 5.10 shows the contents of the SGA's tabu list for experiment set 4 with Schwefel. Along with the locations are listed the composition, school and environment. Unlike other experiments, the tabu list has more heterogeneous schools (school made of fish from different environment). Within the LCE context, heterogeneous school would represent groups of diverse products that can scale out when used together.

Table 5.10

Tabu list contents for experiment set 4 – Schwefel

Center of mass from tabu List			Tabu	#fish	# fish	Total
Design	Operational	Environmental	Performance	From A	From B	
-250.589	200.4185	A	-668.096	4	0	4
52.03411	-177.402	B	-674.953	3	6	9
43.96935	-196.146	B	-628.541	2	6	8
190.3945	85.05112	A	-641.294	1	1	2
-46.2981	57.2122	B	-760.096	3	45	48
-48.6146	-214.007	B	-618.408	0	3	3
-66.2954	60.12544	A	-714.58	6	0	6
-53.8317	53.7223	A	-744.667	11	3	14
386.7243	59.9561	A	-496.749	1	0	1
-76.0	-60.9655	A	-727.712	5	3	8
193.3622	29.97805	A	-671.441	2	0	2
-78.8	52.95076	A	-752.547	10	0	10
-183	155.5399	A	-704.687	1	0	1
-212	-14.4036	A	-652.891	1	0	1
-185.958	19.295	A	-693.135	14	0	14
221.7683	4.757077	A	-672.518	1	0	1
-225.367	1.062059	B	-692.607	4	16	20
-187	2.3209	A	-668.605	36	0	36
-223	-1.00857	A	-681.406	32	0	32
-187.419	-24.846	A	-692.896	17	0	17
-381.054	-4.83064	A	-597.112	30	0	30
-412.277	-7.06338	A	-425.163	38	0	38
-423.489	-5.21758	B	-415.84	12	54	66

The contents of the tabu list were used to calculate the TPI values. The values for the design and the operational parameter were 0.8102 and 0.4144 respectively. The interpretation of those values follows the same logic as previously used.

Similar to the previous experiments, the final population lost its diversity because of the premature convergence the GA. Figure 5.14 shows the plot of the contents of the experiment's tabu list. The best performance for environment A is represented with red dotted circle, whereas environment B is represented with black stars. The plotting shows some areas of the design and operational spectrum where performance will favor one environment at the expenses of the other. Those are areas where only a concentration of best performing products from one type of environment is observed. The plot also shows area of the same spectrum where a product manufactured for environment A, is expected to perform equally well if moved to environment B. Those are areas where there is an overlap between a red dot and a star.

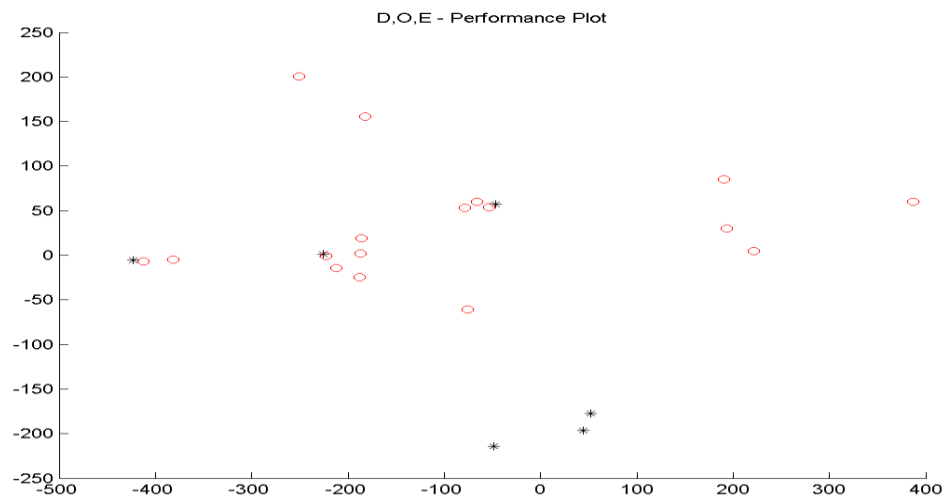


Figure 5.14. Tabu list contents plot for experiment set 4 – Schwefel

5.5.5 Environment, design and operations all driving performance. Performing experiments with more than three parameters and comparing the results obtained to other known

GA-based approaches make possible seeing the potential benefit of grouping on GA, as well as the possible use of SGA as a general optimization method. GA is a trial and error method to problem solving. GA is solution-oriented and makes no attempt to discover why a solution works; merely that it is a solution. SGA on the other hand was built to not only discover solutions that work, but also solutions that share some common characteristics so they can be grouped to make them scale. The purpose of the experiments carried out in this section was to determine whether the grouping feature built into SGA makes SGA a lesser metaheuristic performer when compared to other well-known GA derivatives. Two sets of experiments were carried out for this case.

5.5.5.1 Using Ackley to characterize LCE's relationship. The Ackley's function was modified to turn the problem into a maximization problem with global best performance value being 0. Figure 5.15 shows a plot of the final results for one of the experiments.

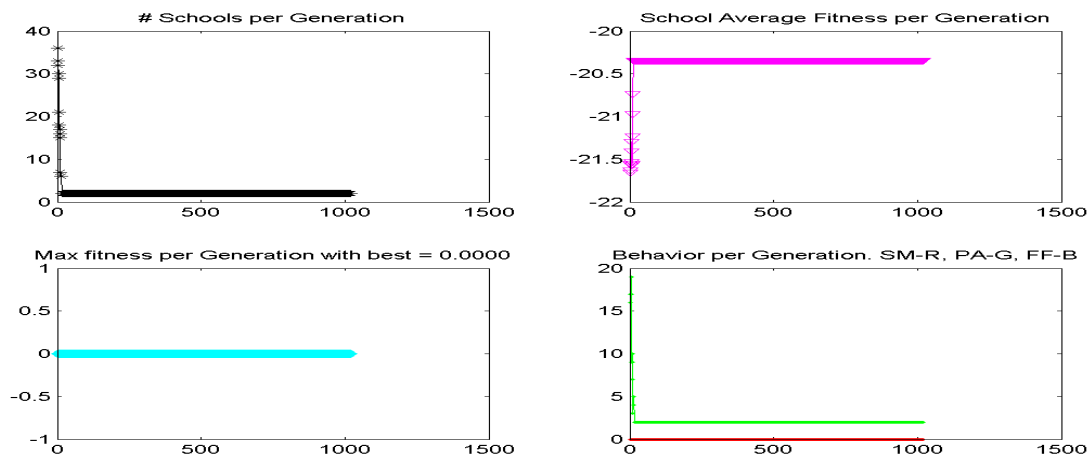


Figure 5.15. SGA results per generation for experiment set 5 – Ackley

From Figure 5.15, it can be seen that as the population's average fitness improves, the number of schools decreases to a final count of 2. The predator avoidance (PA) mode of behavior dominated during the whole simulation. Table 5.11 shows the contents of the SGA's

Tabu list for experiment set 5 with Ackley's function. Along with the locations are listed the composition, schools, and environment.

Table 5.11

Tabu list contents for experiment set 5 – Ackley

Center of mass from tabu List					Tabu	# fish	# fish	Tot
Design	Design	Operation.	Operation.	Environ.	Perform.	From A	From B	
258.00	-224.77	-236.931	-152.174	A	-20.8506	3	0	3
-0.9928	165.980	224.9858	-119.284	A	-20.7190	4	0	4
61.885	203.219	66.9298	-65.0451	B	-20.7005	0	5	5
342.91	-203.80	-240.03169	-185.0174	A	-20.5072	2	0	2
11.129	183.018	14.9800	16.9225	B	-20.2851	0	5	5
124.17	-227.93	-70.0579	64.9784	A	-20.4186	2	2	4
44.911	55.1262	236.0550	-65.902	B	-20.4305	0	1	1
115.00	-59.997	-37.170105	49.027457	B	-20.3397	0	1	1
115.00	-59.997	-37.1701	49.0274	A	-20.3396	2	1	3

The contents of the Tabu list were used to calculate the TPI. The TPI values for the design parameters were 0.3439 and 0.4312. The TPI values for the operational and environmental parameter were 0.4761 and 0.25 respectively. With low TPI values, a designer can change the specification limits on all LCE parameters. Such a change of specifications without impact of the observed performance can significantly lower the overall production cost of any product/system.

Once again, a convergence of the underlying GA was observed, as the final population completely lacked diversity. Therefore, the school mechanisms built into SGA appear to not prevent premature convergence. However, SGA returns more information besides the final solutions. SGA allows the engineer to get an idea on how the underlying life cycle processes a product goes through during its life affect its performance or operational efficiency.

The implementation of parallel GA (PGA) that was made had a total of five subpopulations evolving separately from one another. The number of five has no specific meaning. Each subpopulation had access to the entire search domain. Figure 5.16 shows the evolution per generation of the PGA implementation.

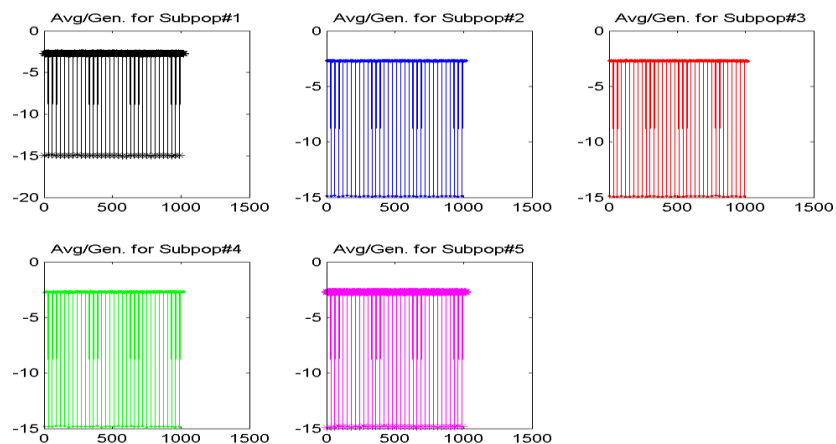


Figure 5.16. Average population fitness over time with PGA for experiment set 5 – Ackley

Table 5.12 represents a sample of the final population obtained after running PGA. Although no better results were achieved by PGA when compared to those recorded by SGA, the final population in PGA has a lot more diversity than that of SGA.

Table 5.12

Sample of final population for PGA experiment set 5 – Ackley

Population					
Design	Design	Operational	Operational	Environment	Performance
-164.799	-481.572	143.7725	-50.9452	B	-21.5891
-150.984	-129.014	45.69213	-48.8014	B	-21.0889
-152.78	42.47536	-0.91078	-143.042	B	-21.4305
134.262	-24.4091	-17.4822	3.941685	A	-21.9349
-10.1991	-158.589	19.10951	-50.6574	B	-21.7926
157.0182	180.3656	-10.7803	246.5109	A	-21.8312
-110.008	9.519213	-55.0603	157.9034	A	-21.167
54.69766	28.84379	87.06999	20.79368	A	-21.2958
-30.1471	265.754	-37.3501	-48.7556	A	-21.6995
-66.0542	165.0322	75.19613	104.2058	B	-20.8368
32.89862	-79.0304	-54.6876	297.3046	A	-21.4126
-166.391	-3.13498	328.9591	-93.6539	A	-21.6443
48.71317	-351.099	128.035	-93.0879	B	-20.8908
-247.845	216.9395	3.951581	229.2697	A	-20.9307

The implementation of island GA (IGA) that was made had a total of five subpopulations evolving on separate islands from one another. However, a migration operator was available to IGA that every 30 generations, would allow the islands to exchange in a round-robin way, up to

5 of their best solutions as a way to re-introduce diversity within a given island. Figure 5.17 shows the evolution per generation of the IGA implementation.

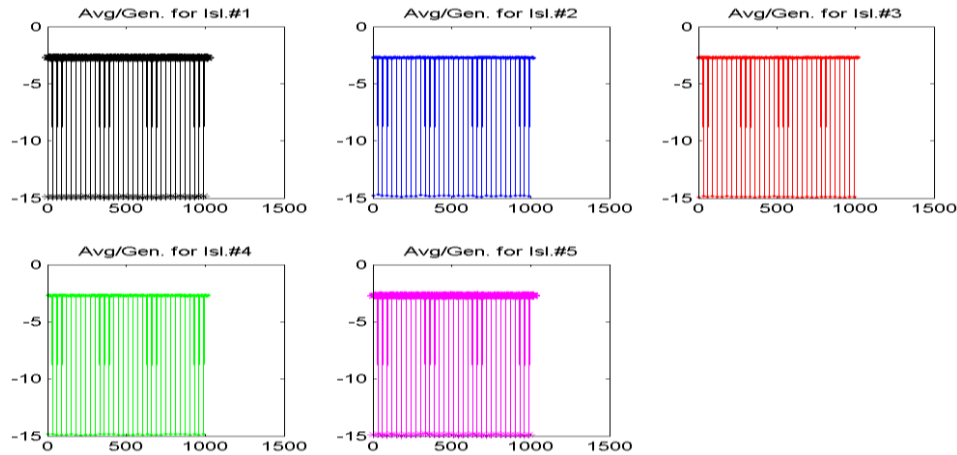


Figure 5.17. Average population fitness over time with IGA for experiment set 5 – Ackley

Table 5.13 represents a sample of the final population obtained after running IGA. Here also, no better results were achieved when compared to SGA. Also, unlike SGA, the final population in IGA has a lot more diversity.

Making an overall comparison based on the quality of the final population (final population average fitness), SGA appears to be performing slightly better than both PGA and IGA. Therefore, the grouping feature built into SGA seems to keep the quality of the solutions at worst at the quality level of PGA and SGA.

Table 5.13

Sample of final population for IGA experiment set 5 – Ackley

Population					
Design	Design	Operational	Operational	Environment	Performance
59.17157	149.4894	78.91444	-44.9946	A	-21.3225
-51.3367	67.59259	-58.8961	-185.948	B	-21.6164
-6.18678	-127.84	-0.44204	14.86595	A	-21.5407
-16.9806	-66.4336	377.3741	-265.759	A	-21.85
36.25633	149.859	-100.201	-98.6796	A	-21.5935
-124.87	-63.334	6.866562	-211.447	A	-21.7417
79.23545	-131.754	-107.063	-82.734	A	-21.4536
-37.0189	-45.1076	-44.081	-148.812	B	-20.588
-153.066	-290.224	62.93955	-140.625	A	-21.3334
38.14309	60.99853	407.2399	-113.165	A	-20.9871
-135.449	-44.9966	239.352	46.58049	B	-22.0175
149.4617	104.3652	152.6735	55.39732	A	-22.2333
-29.057	-43.0327	261.2078	341.5205	A	-21.3733
11.56537	-149.069	-154.211	-148.015	A	-21.3592
106.5239	128.9473	-398.941	-297.036	A	-21.1254
-48.8613	233.3871	28.19601	-116.926	A	-21.3975

5.5.5.2 Using Schwefel to characterize LCE's relationship. The Schwefel function was modified to turn the problem into a maximization problem with global best performance value

being 0 with six parameters. Figure 5.18 shows a plot of the final results for one of the experiments.

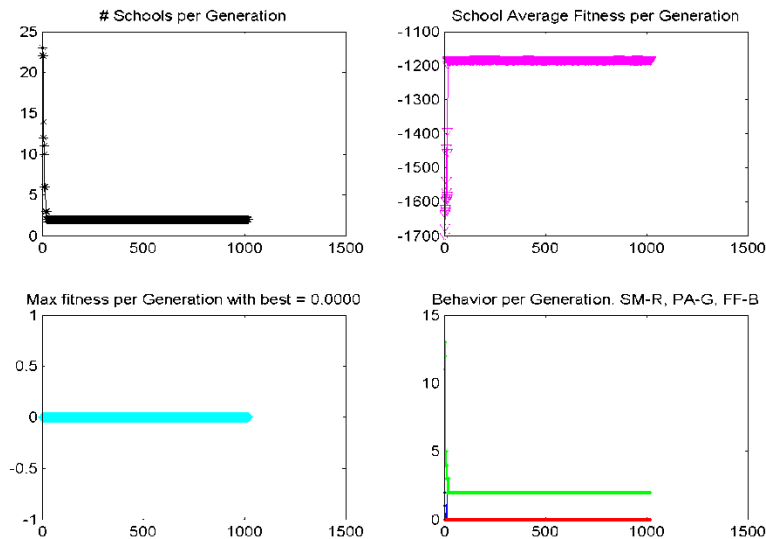


Figure 5.18. SGA results per generation for experiment set 5 – Schwefel

From Figure 5.18, it can be seen that the simulation started with 23 schools, and as the population's average fitness improved, the number of schools decreases to a final count of 2. The PA mode of behavior dominated during the whole simulation. Table 5.14 shows the contents of the SGA's Tabu list for experiment set 5 with Schwefel. Along with the locations are listed the composition, school, and environment.

The contents of the tabu list were used to calculate the TPI. The values found were 0.1205 and 0.5988 for the design parameters, and 0.4281 and 0.4255 for the operational parameters. The interpretation of those values follows the same logic as previously used.

Table 5.14

Tabu list contents for experiment set 5 – Schwefel

Center of mass from tabu						Tabu	# fish	# fish	Tot
Design	Design	Oper.	Oper.	Envmt	Envmt	Perform.	From A	From B	
210.499	-44.3764	-211.401	-58.9357	-421	-421	-1215.6	12	0	12
203.901	-54.981	-186.05	-57.551	-421	-421	-1211.7	12	0	12
128.126	391.989	216.716	74.3986	421	421	-1240.4	0	1	1
214.337	4.01311	-170.286	203.008	-421	-421	-1210.0	3	0	3
200.08	-187.992	216.413	100.685	421	421	-1187.2	0	2	2
93.820	-8.093	-203.508	366.578	-421	-421	-1386.1	3	0	3
193.668	-206.812	174.553	133.469	421	421	-1290.0	0	47	47
190.887	-192.707	194.522	117.762	421	421	-1232.7	2	44	46
210.410	-75.729	7.47638	189.982	-421	-421	-1247.9	1	1	2
199.587	-190.474	210.934	105.087	-421	-421	-1185.6	23	2	25

Here also, the SGA converged. Table 5.15 shows a sample of the final population.

Table 5.15

Sample of final population for SGA in experiment set 5 – Schwefel

Population							
Design	Design	Oper.	Oper.	Envmt.	Envmt.	School ID	Performance
199.726	-189.773	212.4815	103.8441	-421	-421	1	-1185.686114
199.726	-189.773	212.4815	103.8441	-421	-421	1	-1182.902624
199.726	-189.773	212.4815	103.8441	-421	-421	1	-1182.897779
199.726	-189.773	212.4815	103.8441	421	421	1	-1181.014161
199.726	-189.773	212.4815	103.8441	421	421	2	-1183.284507
199.726	-189.773	212.4815	103.8441	421	421	2	-1181.469692

The same implementation of parallel GA (PGA) was modified to use a phenotype of length six and was run with subpopulations evolving separately from one another. Figure 5.19 shows the evolution per generation of the PGA implementation.

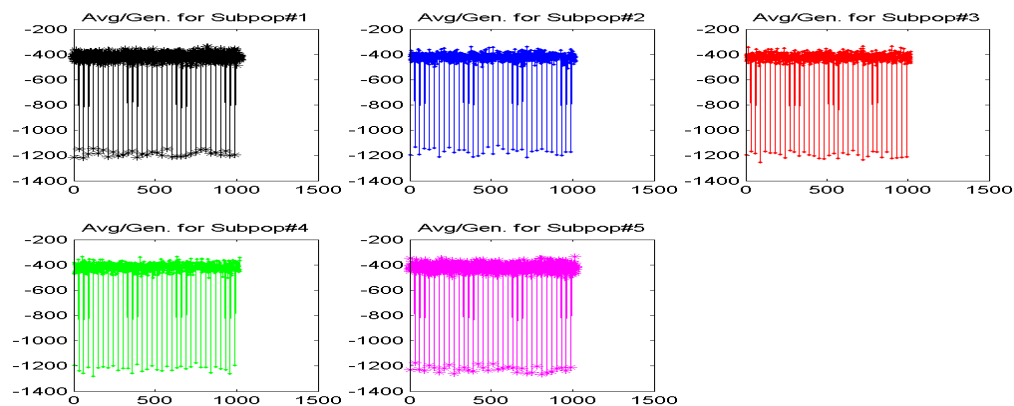


Figure 5.19. Average population fitness over time with PGA for experiment set 5 – Schwefel

Table 5.16 represents a sample of the final population obtained after running PGA. Unlike SGA, the final population in PGA did not converge and therefore, has a lot more diversity.

Table 5.16

Sample of final population for PGA experiment set 5 – Schwefel

Population						
Design	Design	Operation	Operation	Environment	Environment	Performance
-110.764	110.6206	114.1634	-122.78	421	421	-2103.88
-25.7738	-152.613	168.5419	-135.198	421	421	-1773.22
-234.052	-91.0003	123.9331	163.4755	421	421	-1680.41
-149.203	-41.7851	150.3702	-21.3514	-421	-421	-1786.03
-46.3813	-211.623	100.9683	192.426	-421	-421	-1331.79
-180.18	105.9761	-330.7	-46.8641	421	421	-1799.66
-117.068	10.2877	115.6455	81.87175	-421	-421	-1874.06
235.6803	-159.186	-163.512	218.4388	421	421	-1375.04
-37.0359	281.4288	-71.315	91.45364	421	421	-1883.17
-85.9352	44.27344	193.3139	33.32065	421	421	-1474.88
-320.59	141.7975	155.5812	116.4146	421	421	-2150.88
18.55694	-31.4216	37.39198	-144.287	-421	-421	-1794.91
-85.6409	112.4326	245.2906	64.79873	421	421	-1690.4
225.2474	77.72268	1.193966	27.54271	-421	-421	-1509.05
4.131756	-450.11	-114.329	-130.813	-421	-421	-1584.54
223.6794	-55.4336	78.82555	21.12316	-421	-421	-1452.28

The same implementation of island GA (IGA) was modified to use a phenotype of length six and was run with subpopulations evolving separately from one another. Figure 5.20 shows the evolution per generation of the IGA implementation.

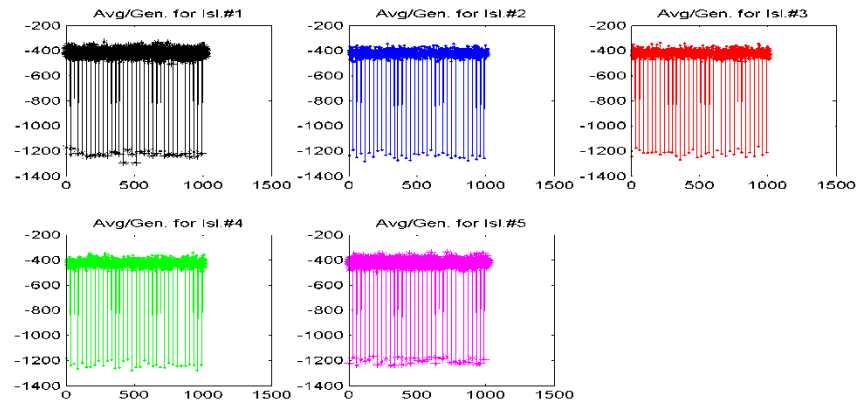


Figure 5.20. Average population fitness over time with IGA for experiment set 5 – Schwefel

Table 5.17 represents a sample of the final population obtained after running IGA. Unlike SGA, the final population in IGA has a lot more diversity.

Once again, making an overall comparison based on the quality of the final population (final population average fitness), SGA performance appears to be just as bad as either PGA or IGA. Therefore, the grouping feature built into SGA seems not to be affecting the ability of GA to successfully perform metaheuristic search.

Table 5.17

Sample of final population for IGA experiment set 5 – Schwefel

Population						
Design	Design	Operation	Operation	Environment	Environment	Performance
75.329	-167.453	-262.074	300.293	-421	-421	-1984.768
-33.249	154.567	341.833	-63.650	421	421	-1770.517
83.422	-122.429	294.779	-45.117	421	421	-2048.165
-186.50	175.472	-61.005	206.677	-421	-421	-1138.464
-18.631	-8.129	-133.941	98.169	421	421	-1848.615
-364.10	107.688	443.436	101.915	421	421	-1386.967
-138.43	119.805	-320.32	224.222	-421	-421	-2005.699
130.081	-346.562	-287.362	133.558	-421	-421	-2260.600
13.845	-251.76	-202.456	-190.149	-421	-421	-1342.991
258.964	55.81715	-328.877	-61.453	-421	-421	-1875.343
52.933	228.1491	-420.826	121.588	-421	-421	-1204.704
40.352	-220.977	-66.664	98.310	-421	-421	-1491.028
29.213	58.4483	-39.935	-114.615	-421	-421	-1749.626
493.048	385.762	-107.189	5.968	421	421	-1588.027

5.6 Summary

Within this chapter, schooling genetic algorithms (SGA) was applied to the GLPD methodology of continuous product design. Simulations of SGA were carried out according to a set of factors including the relationship between LCE parameters, the number of parameters, and

the shape of the search domain. The quality of the solutions returned by SGA was assessed using different criteria including the TPI values for each LCE parameter type, and the distribution of behavioral mode.

A first set of experiments (1-4) was successfully carried to better understand how SGA works on different types of environments, and how the SGA built-in grouping shapes both the search process and the interpretation of the returned results. As the nature of the LCE parameters' relationship gets more complex, SGA appears to return more heterogeneous groups when the density of deceptive attractors within the search domain is low (Schwefel function). Similarly, SGA appears to return more homogeneous groups when the density of deceptive attractors within the search domain is high (Griewank, Ackel function).

A last (5th) set of experiments that was carried to assess the impact of grouping on the GA search process by comparing SGA to both parallel GA (PGA) and island GA (IGA) proved conclusive. The grouping mechanism appears not to negatively impact SGA. However, further experiments with different type of LCE parameter relationships would be necessary to decide whether the metaheuristic search process of GA was improved as suggested by the results from one of the sets of experiments.

Overall, the SGA would converge even when GA, PGA and SGA will not. By returning the average best solutions that were found and stored within a recency list, SGA provides a view of the series of improvements a group of products has gone through. With some of the experiments the SGA was able, via the TPI values, to provide insights on better specification limit for LCE parameters.

Table 5.18 summarizes the results of all the experiments.

Table 5.18

Summary of experiments on SGA

SGA Summary		
Linear & Polynomial Performance	Griewank & Schwefel Performance	Ackley & Schwefel Performance
Convergence observed within population for both environments	Some delimited characterization of LCE parameters relationship for both environments	Performs better than IGA and PGA on optimization
Good characterization of LCE parameters relationship for both environments	Food foraging and predator avoidance were both frequently used	Tabu shows that best schools are small schools
School maintenance observed as the predominant behavior for schools	TPI values indicate where parameter range can be reduced	TPI values indicate where parameter range can be reduced
TPI values indicate where parameter range can be reduced	Final best school centers in tabu indicate the presence of multiple attractors within the search domain	Predator avoidance observed as the predominant behavior
	Center of mass capturing best solutions for both environments were found	Population average fitness quickly rise and stabilizes

CHAPTER 6

Genetic Social Networks

Genetic Social Networks (GSNs) are GA-based models that enable process and operator adaptability by mimicking social networks. In GSN, operators behave differently depending on the perceived immediate area of the search domain, and on social networking connections and dynamics. This chapter serves as an introduction to GSN.

6.1 Introduction

GAs have been successful at solving problems in a variety of engineering fields ranging from engineering cost control to the design and the implementation of systems (Hassan, Azubir, Nizam, Toha, & Ibrahim, 2012; Sarkar, Mandal, Saha, Mookherjee, & Sanyal, 2013; Toulabi, Shiroei, & Ranjbar, 2014). Applications of GAs to complex design optimization problems rely on population's diversity in order to generate solutions that are acceptable. GAs are a trial and error approach, and as such, GAs are solution-oriented, and problem-specific (do not attempt to generalize a solution to other problems). GAs require little knowledge to get started, and are typically good for problems where there are multiple chances to get the best solution. Such problems are common in engineering design.

However, when solving for a problem in general, and for a design problem in particular, solutions with given characteristics are sometimes desirable. Such characteristics include evolvability; have some system efficiency due to the economy of scale, or the ability to continuously operate with minimal intervention. Solving a problem by looking for solutions with the described characteristics is difficult for GAs. But building on the strengths of GAs, the motivation for using social networks comes from the dynamic nature of the component of social networks: people or organizations. Within social networks, individuals can join or leave a group

at any time. Social networks are resilient and scale naturally (A.-L. Barabási & Albert, 1999; A. L. Barabási et al., 2002). The adaptability and scaling features of social networks are important and desirable in life cycle engineering (LCE). By mimicking social network interactions, the main objective is to come up with LCE solutions possessing the same characteristics as those of social networks.

This chapter presents a new approach to GAs. The approach is carried out by adding social networking behavior to GAs. In order to assess the suitability of social networking behavior to search heuristics, a model is designed that mimics a population of individuals who are socially networked. The proposed model is called genetic social network (GSN). It is further described in the next sections.

6.2 GSN Overview

Social networks (SNs) are social structures made up of a set of social entities (such as individuals or organizations) and a set of interactional ties between these entities. The growing trend and rise in power of some social networks such as Facebook or Twitter have driven increased research. Social network's modeling serves at least two purposes. First, such modeling promotes understanding of social networks formation and evolution. Second, studying network-dependent social processes by simulation can be used to specify or anticipate the structure of social networks' interactions (A. L. Barabási et al., 2002; Eubank et al., 2004). GSN serves a third purpose: mimicking and applying high level social network concepts to problem solving.

A GSN is a hybrid model that combines social network dynamics with GAs. GSN applies nodal attribute models (NAMs) with an evolutionary aspect to traditional GAs. NAMs are social networks models where the probability of each link existing within the network depends only on nodal attributes, the local network structure being irrelevant (Toivonen et al., 2009). NAMs have

also been described by the term spatial models (Boguna, Romualdo, Diaz-Guilera, & Arenas, 2003; Wong, Pattison, & Robins, 2006), to refer to the fact that the attributes of each node determine its 'location' in a social or geographical space. NAMs represent one of the main two categories of models existing in physics-oriented network literature (Toivonen et al., 2009). The other category, network evolution models (NEMs), is characterized by the addition of new links (friends joining), based on the local network structure. NEMs focus on network evolution mechanisms, and are used to predict the outcome of a network growth based on specific network evolution mechanisms observed within that network. GSN uses a distance-based scheme (referred to as influential distance) to determine the membership of an individual to a specific group sample (group of friends) of a population. GSN applies NAMs by assigning probabilities to edges e_{ij} forming between two nodes (respectively nodes i and j) as a function of the attributes of nodes i and j only.

GSN adds social network concepts to GAs, by implementing single and dyadic social interactions of social groups with GA operators (crossover and mutation). GSN does this by viewing a whole GA population as a graph, and using both the fitness values and the strengths of the created links/bounds to assess whether a node (an individual from the population) is fit for mating. In GSN mutation is used to implement single social interactions, whereas dyadic social interactions are implemented via crossover.

To mimic social interactions concepts with groups, GSNs introduce the following notions:

- theme
- group
- term

- leader, and membership

Themes are candidate solutions, not part of the original population, which are either carefully or randomly selected by the GSN. Real life group themes could be: fashion, hip hop, soccer.... Themes are first set at the beginning of a GSN simulation, and updated periodically, at the beginning of each term. A group is organized around a single theme. The number of themes (therefore groups) to be used by GSN is a design decision and can be set to any arbitrary value. Terms correspond to the number of GA generations an individual gets to be the leader of a group before group themes are updated. Group themes are updated at the beginning of every term to keep up with the changes that have occurred within the population. as a result, the themes themselves evolve as the population evolves. Themes are used to select the leader of their groups. Leaders' selection occurs at the beginning of a term. Leaders are among the most fit and theme "knowledgeable" individuals of the population. An individual becomes a group leader when two conditions are met. First, of all the individuals making up the population, an individual must have the highest similarity (proximity) to any given theme. Second, the fitness value of that individual must be higher than an arbitrary threshold value set at the beginning of the simulation. The requirement for the highest similarity to the theme is to mimic the expertise of a group leader as the most knowledgeable individual of his group. The requirement on the fitness value of the leader is to only allow "strong" individuals as group leaders. The use of two requirements for a group leader also makes it impossible for any individual within a well-diversified population, to lead more than one group at a time. Finally, there is a membership concept characterizing the level of belonging of an individual to a group. In GSN, an individual's membership level to a given group is expressed as a rational number (0.0 to 1.0). The memberships of individuals to groups change constantly causing the population to dynamically

reorganize itself according to the formation of links between its selected leaders and the population. Those constant reorganizations are based on the ability of leaders to influence others. Such ability is represented in GSNs as a gravitational pull. The higher the leadership ability of an individual, the higher is the leader's ability to form links/bounds that yield dynamic mating scheme called proportionate breeding.

In GSN, the structures observed within the social network are explained by the interactions of individuals, with reference to their intrinsic properties (ability to influence peers). In social networks, links are created based on assumptions about the local mechanisms of tie formation, such as people meeting friends of friends, and thus forming connections with their network neighbors (triadic closure) (Granovetter, 1973). GSN uses a different approach by automatically creating links from all individuals (also referred to as solution or configuration when within the context of LCE problem solving) to the group leaders. The approach of GSN is simple, enables a global (population wise) linkage mechanism.

Figure 6.1 shows a graphical representation of a GSN with nine individuals and two themes. Since there are two themes, there are groups that are formed and therefore two leaders.

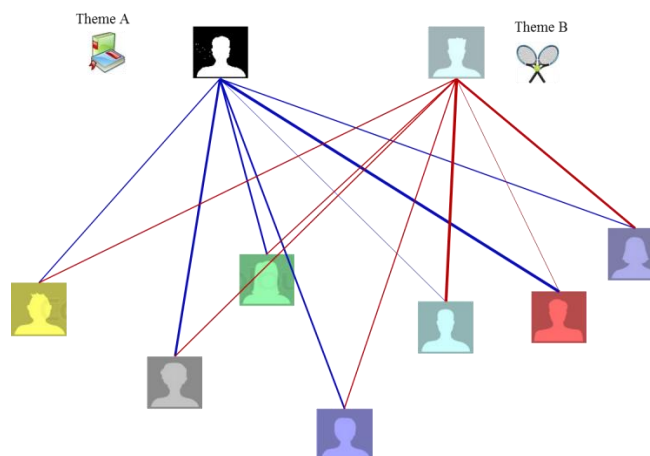


Figure 6.1. Genetic social networking

The two leaders are represented by the individuals with more than two links connected to them. Each group is also characterized by the color of the link. The thickness of the links represents the level of membership of the corresponding individuals to the given groups. Fit individuals with high level of membership to a group have a higher selection probability for mating and lower selection probability for culling.

6.2.1 Gravitational pull. Also known as the sphere of influence's value, this is a value quantifying the influence a leader has on the population. The gravitational pull is proportional to both a leader and a follower fitness values, and inversely proportional to the squared distance of the follower to the leader. A follower is any individual that an individual who is not a leader. The gravitational pull by the leader of group i on individual j , calculated at the beginning of each term, is given as:

$$G_{i,j} = \frac{F_{L,i} \times F_j}{d_{l-f}^2}$$

where $F_{L,i}$ is the fitness of the leader of group i , F_j is the fitness of the potential follower j , and d_{l-f} the Manhattan distance leader to potential follower.

6.2.2 Proportionate breeding. During the mating process, group level proportionate breeding is observed. Proportionate breeding stems from the fact that each generation, each group is allowed to gain a certain number of new members. GSN rewards groups based on their overall fitness and on their size. For example, large groups of individuals with fitness values around the overall population average can be given the same opportunities to gain new members as small groups of highly fit individuals.

Assuming a constant number of crossovers and mutations for each generation, a group breeding proportion is calculated based on its relative size and performance level. The proportion indicates the percentage of new members allowed for that term for that group. With N social

groups in the environment composed of G individuals, where the performance of individual j in group i , denoted $P_{j,i}$, is calculated as the product of individual j fitness by its membership level $M_{j,i}$ within group i . The allocation of the numbers of new members NM_i to group i is calculated as:

$$NM_i = \frac{\sum_{j=1}^G P_{j,i} - \sum_{j=1}^G F_{j,i} \times M_{i,j}}{\sum_{i=1}^N \sum_{j=1}^G P_{j,i} - \sum_{i=1}^N \sum_{j=1}^G F_{j,i} \times M_{i,j}}$$

The performance $P_{j,i}$ of individual j in group i is calculated as the product of their fitness by their membership level $M_{i,j}$ as a way of rewarding “committed” members. Proceeding this way, groups with high performing (high membership and fitness values) members have a higher probability of thriving compared to groups of either smaller size, or less high performing individuals. The next section is about the GSN procedure and includes a high level pseudocode description of GSN algorithms.

6.3 GSN Procedure

Themes $\{T_i, i \in \mathbb{N}\}$ are created first. The number of initial themes determines the upper limit for the number of groups that will ever exist at one point of time within the environment. Themes and individuals are encoded with the same phenotype. This way, it is possible to define and use a similarity measure between individuals and themes. Such a measure mimics the affinity of an individual to that given theme. A non-exhaustive list of usable distance and similarity measures is available in (Cha, 2007). Assuming that the Sørensen distance $S_{i,j}$ represents the similarities between individuals I_j and the existing T_i themes, the leader of group i would be the individual meeting the following two requirements:

$$\underset{I_j}{\operatorname{argmax}} \left\{ S_{i,j} = \frac{\sum_{j=1}^N |I_j - T_i|}{\sum_{j=1}^N (I_j + T_i)} \right\}, \text{ and } F_j \geq m_F$$

where F_j is the fitness value of individual j , and m_F the previously described threshold value. To be problem independent, m_F could be set as a given percentage of the current best fitness value. If no individual meet both requirements for a given theme during any term, the GSN will then only run with the other groups during that term. The group without a leader will be considered as dormant. Such situations are likely in social networks where a group can be inactive just because of lack of membership or of appropriate leadership. If more than one individual is eligible to the leadership role, then the first found candidate is selected.

Following the emergence of leaders due to their affinity level (determined by their relatively strong similarities to the existing themes), links from leaders to followers are formed based on the influences of group leaders. Choosing leaders instead of the themes as the attraction is because a leader, unlike a theme, already has the obligation to be a “good” candidate solution. Therefore, relying on leaders for the attraction is likely to yield a higher performance for the followers. Every individual is modeled as belonging to all groups, with a given level of membership. Fuzzy membership is used to represent levels of membership to groups.

Figure 6.2 shows a high level diagram of GSN. Similar to SGA, GSN is adaptive and does not follow the systematic execution flow of GA operations. A group that is expanding will need to do more exploration (using the GA mutation operator) in order to become more diverse, whereas a group that is specializing will capitalize on the strengths of its members only.

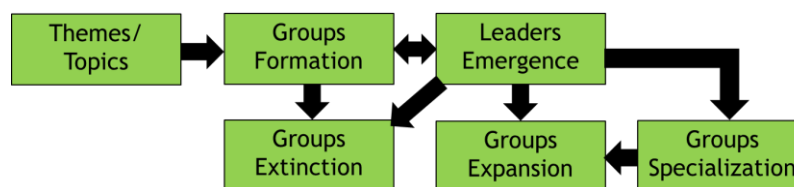


Figure 6.2. GSN high level diagram

6.4 GSN Modeling

The modeling of GSN involves the different processes of GSN along with the underlying concepts. GSN most notable features are its ability to group individuals based on their common interests, and to exploit the grouping. The grouping feature relies on the ability of individuals to create and maintain links.

6.4.1 Joining and leaving groups. Figure 6.1 showed an example of social network with two themes. Within the network, each group is a subset of the social network. The GSN algorithm is implemented so that each group has exactly one leader. Since each group is based on a given theme set at the very beginning of the experiment, individuals join groups based on their level of affinity with the leader of that group. Because leaders, rather than themes, are used to attract members, the GSN always assigns a membership value of 1.0 to all the leaders. The membership level of the other members to the group, after they join, is determined by the pull of the leader of the group.

Let us have the Sørensen distance $S_{i,j}$ represent the similarity between individuals I_j and the existing T_i themes. Let $M_{i,j}$ represent the membership level of individual I_j to group i , F_j the fitness value of individual I_j , and $T_i(t + 1)$ the value of theme T_i at term $t + 1$. Let α_j represent the normalization factor of the gravitational pull that is used to ensure that $M_{i,j}$ is within the (0.0 1.0) range, with the membership of a leader being 1.0. Finally, let $F_{L,i}$ represent the fitness value of the leader of group i , and d_{L-j} the distance from group leader to follower j . The following are the expressions of some of the dynamic concepts related to joining and leaving groups:

$$M_{i,j} = \alpha_j \times G_{i,j}$$

$$T_i(t + 1) = T_i(t) + \frac{\sum_{j=1}^N I_j \times F_j \times M_{i,j}}{\sum_{j=1}^N F_j \times M_{i,j}}$$

6.4.2 Fuzzy membership. The current unique purpose of the memberships is to drive the search process. Using fuzzy membership, each individual may be represented as a member of all of the existing groups, but with different levels of membership. Only two types of membership exist within any group: a leader, and a follower. A member of a group, leader excluded, has a level of involvement in the group that depends on their membership level.

In GSN, it is assumed that brand new members (whether created by crossover of mutation) to a group would join because of the attributes of their friends. For that reason, the level of membership is an inheritable characteristic. Therefore, pending a reassessment of themes at the end of a leadership term, the level of membership of offspring (new members to a group) is calculated differently depending on whether the offspring is a result of a crossover or a mutation. Such an approach is used to account for the differences between the crossover and the mutation operators: the former requires at least two parents from which a child would inherit characteristics, whereas the later requires one. Considering crossover, the new member's level of membership will then be a fuzzy AND of the parents' level of membership. This way, the offspring only gets the best from their parents. Considering mutation, the new member's level of membership will be a fuzzy OR of the single parent with random values (0.0 1.0). This way, the offspring gets a chance of exploring the network as a whole. Both cases are summarized below:

$$\text{level of membership for crossover} = M_{i,j} = \min(M_{i,j}^{Parent1}, M_{i,j}^{Parent2})$$

$$\text{level of membership for mutation} = M_{i,j} = \max(M_{i,j}^{Parent}, rand)$$

The application of GA operators is based on groups. But since individuals belong to all groups with various levels of memberships, each individual gets a chance to participate to

crossover and mutation within all the available groups. Proceeding further, when choosing the candidate parents, GSN uses weighted roulette-wheel selection (SCX), a form of fitness proportionate selection. Except that performance, instead of fit ness, is used within the selection process. The weighted SCX takes into consideration the level of membership of individuals within the group when computing the chances of being selected. Let G be the population size of the GSN. The probability of individual selection (for either crossover or mutation) $p_{j,i}$ of individual j from group i is given as:

$$p_{j,i} = \frac{P_{j,i}}{\sum_j P_{j,i}} = \frac{M_{i,j} \times F_{j,i}}{\sum_{i=1}^G M_{i,j} \times F_{j,i}}$$

The GSN life cycle represents the steps required for implementing a GSN algorithm based on the given concepts. The life cycle of GSN is rather short. A high level overview of the GSN is given next.

Algorithm 2. Genetic Social Network High Level Metaheuristic

Set the parameters, initialize the population
while termination condition not met do
 Set/update the themes for each group
 Organize individuals in groups based on group theme
 Assign Leadership role based on parameters' values
 Reset all term counters for newly elected leaders
 while term counter still running
 foreach group within the world:
 Calculate the number of new members that are allowed
 Apply GA operators based on the number of new members and group dynamics
 Increment term counter for leaders
 endforeach
 Proceed with culling via reduction to keep overall population size constant
endwhile
endwhile

6.5 Summary

Within this chapter, genetic social networking (GSN), a new metaheuristic created to work on problems where scaling and sustainability are characteristics of the solution was developed, explained, and discussed. GSN mimics social networking and uses GAs that it extends with some new concepts. GSN develops and implements the concepts of group joining and leaving, and of group fuzzy membership. GSN defines and uses the concepts of group, term, theme, membership, and leader to mimic and track dyadic links within a social network. To finish, a high level overview of GSN was provided.

Chapter 7

Applying Genetic Social Networks to Generalized life cycle product design

7.1 Introduction

Genetic social networks (GSNs) are GA-based metaheuristics emulating social networking dynamics, and designed for problems where evolvability, grouping, and sustainability are characteristics of the solution. Generalized life cycle product design (GLPD) is a PLE-based approach for product design that uses LCE-data characterization to assist system engineers with their design process. The product design process is complex and costly. It relies on expert knowledge. In this chapter, the applicability of GSN to product design, using GLPD, is investigated. Using simulated life cycle engineering data, different case scenarios are envisaged to assess GSN as an analytic tool for better product design.

7.2 Applying GSN to GLPD

The GLPD approach is proposed as an attempt to define a life cycle based sustainable design process. The aim of the GLPD is to capitalize on biologically and sociologically inspired design to create product configurations that can evolve, group, and sustain. Figure 3.6 shows the GLPD approach. Considering the highly scalable nature of social networks, structures that GSNs mimic, how do we use GSN to incorporate evolvability (evolution in design and operational parameters of products and product's parts), grouping (based on environmental parameters related to a product), and sustainability (ability of the system to maintain and improve itself) into a product attributes?

In an attempt to answer the question, GSN will be applied to the GLPD representation. Using simulated PLE-data generated for a product, a product's DNA, as displayed in Figure 7.1, was created and was used as an input to a GSN algorithm.

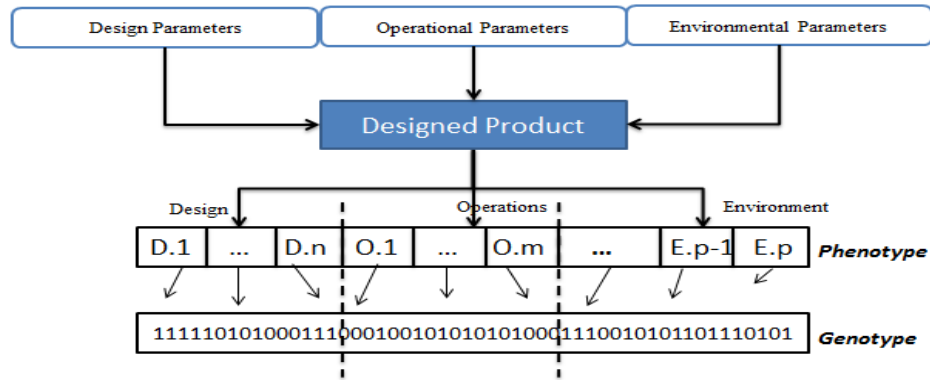


Figure 7.1. Transformation of a designed product into a GA entity

The GSN concepts, within the LCE context, take the following meanings:

- A theme is one of the early better configurations the systems engineer came up with for a product/system. It is a conceptual configuration.
- A leader is a current good configuration for a product/system. Defined this way, the role of a leader is to coordinate the improvements of a group based on constantly updated life cycle parameters. A leader is a prototype or most closely configured fielded system.
- A group is a set of configurations with common characteristics. Since all solutions (product/system configuration) are members of all existing groups with some level of membership, then the notion of group makes better sense within a meaningful definition of membership.
- A membership level is assigned as a prerogative of a group leader. Therefore, the higher the membership of a candidate solution is, the lesser is the variability between that solution the leader of the group.

7.3 GSN Approach to the Problem

The objective is to find out the potential benefits of using GSN and GLPD in the design process of a system/product. Using the PLE-data generated by a product, a product DNA, as displayed in Figure 7.1 was created and was used as an input to a GSN algorithm. Within the

designed GSN test bed, values for n , m , and p varied from 0 to 2. When applying a fuzzy or GA operator to a solution, the environmental phenotypic value is given special care so both its permissible values (discrete only), and proportion are preserved in between generations.

Using probability distributions, constrained random values were generated that represented the design, operational and environmental parameters of a hypothetical product. Simulations were then carried using one or more parameters of each type. LCE data was generated to test GLPD with GSN for the following scenarios described in the following sections:

- Environment driving design for performance
- Environment driving both design and operation for performance
- Environment and design both driving operations for performance
- Environment and design only driving performance
- Environment, design, and operations all driving performance

7.4 Experimental Design

For the experiments/simulations, each candidate solution was an n -dimensional vector of the search space, where n is the sum of the number of parameter (of each kind) used. Each dimension or independent product performance parameter was coded with a 10-bit string value (actual decimal values falling within the range of -500 to 500). The size of the population was set and maintained at 120 individuals. The proportion for each environment was 55% for environment A, and 45% for environment B. The stopping criteria was set to be either the maximum number of generations (set to 1000), or the lack of improvement of the average fitness value of the population for three consecutive generations, whichever came first. Each experiment was repeated three times and the recorded assessment criteria were: (1) the number of

generations it took for GSN to converge (if convergence occurs), (2) the solutions returned, and (3) the quality of the final themes. Each simulation started with a population of individuals randomly distributed (normally) across the search domain, followed by the environmental parameter value(s) being assigned, using a uniform distribution, to each individual.

Table 7.1 shows the factors, assessment criteria, and methods that were used to assess the quality of the experiments that were carried out. As far as the relationship is concerned (first factor), five were tested as explained within the previous section. The number of parameters or alleles changed from three to six. Linear, polynomial, Ackley, Griewank, and Schwefel functions were all used to characterize the underlying shape of the search domain. Different assessment criteria were used. For some experiments, GSN performance was also compared to island GA (IGA) and parallel GA (PGA). Table 7.1 shows all the factors, assessment criteria, and methods that were used to test GSN.

Table 7.1

Factors, assessment criteria, and methods

Factors	Assessment criteria
Relationship {LCE characterization}	Number of generations to convergence
Size {Number of parameters/alleles }	Quality of solutions obtained
Underlying shape {Ackley, Griewank, Linear, Polynomial, Schwefel }	Quality of final themes
Methods	
Genetic Social Network (GSN) vs. {PGA, IGA, SGA }	

7.4.1 Environment driving design for performance. The objective of the experiments carried here was to determine the ability of GSN to effectively use grouping to converge on separate environments, and to check the ability of GSN to effectively characterize the life cycle parameters' relationship. The relation used for the experiments was linear as follow:

$$f([D_1, O_1, E_1]) = \begin{cases} |1 - |\frac{D_1}{max_{D_1}} - 0.7|| & \text{if } E_1 \text{ is A} \\ |1 - |\frac{D_1}{max_{D_1}} - 0.3|| & \text{otherwise} \end{cases} ; -500 \leq D_1, O_1 \leq 500$$

7.4.2 Environment driving both design and operations for performance. The objective of the experiments was to determine the ability of GSN to effectively use grouping to converge on separate environments, and to check the ability of GSN to characterize the life cycle parameters' relationship. The relation used for the experiments was polynomial as follow:

$$f([D_1, O_1, E_1]) = \begin{cases} 1 - ||\frac{D_1}{max_{D_1}} - 0.7| \times ||\frac{O_1}{max_{O_1}} - 0.3| & \text{if } E_1 \text{ is A} \\ 1 - ||\frac{D_1}{max_{D_1}} - 0.3| \times ||\frac{O_1}{max_{O_1}} - 0.7| & \text{otherwise} \end{cases} ; -500 \leq D_1, O_1 \leq 500$$

7.4.3 Environment and design both driving operations for performance. The objective of the experiments carried here was to determine the ability of GSN to effectively use grouping to converge on separate environments, and to check the ability of GSN to characterize the life cycle parameters' relationship. The relation used for the experiments was linear as follow:

$$f([D_1, O_1, E_1]) = \begin{cases} 1 - |\frac{O_1}{max_{D_1}} - 0.7|; E_1 \text{ is A, } D_1 < 0 \\ 1 - |\frac{O_1}{max_{D_1}} - 0.3|; E_1 \text{ is A, } D_1 \geq 0 \\ 1 - |\frac{O_1}{max_{D_1}} - 0.9|; E_1 \text{ is B, } D_1 < 0 \\ 1 - |\frac{O_1}{max_{D_1}} - 0.1|; E_1 \text{ is B, } D_1 \geq 0 \end{cases} ; -500 \leq D_1, O_1 \leq 500$$

7.4.4 Environment and design both driving performance. The objective of the experiments was to not only determine the ability of GSN to effectively use grouping to converge on separate

environments, but also to check the ability of GSN to operate on “noisy” life cycle data. Similar to the previous experiments, a typical representation of a solution was picked to have three parameters, one of each kind. Two different sets of experiments were carried. They differed in the function that was used to generate the LCE relationship. The Griewank function was used for the first experiment, and the Schwefel function was used for the second experiment. The Griewank function (Griewank, 1981) is a standard test functions for unconstrained global optimization, and it is used to test the convergence of optimization functions. It has many widespread local minima regularly distributed, to act as attractor to deceive the search process. The function that was used was a modified Griewank function to make the problem a maximization problem. The function is defined by:

$$f([D_1, O_1, E_1]) = -\left[\frac{D_1^2 + E_1^2}{4000} - \cos(D_1) \times \cos\left(\frac{E_1}{\sqrt{2}}\right) + 1\right]; -500 \leq D_1, O_1 \leq 500$$

The Schwefel function on the other hand is a deceptive function that has its global minimum geometrically distant, over the parameter space, from the next best local minima. Therefore, with the Schwefel function, the search algorithms are potentially prone to convergence in the wrong direction. The function that was used was a modified Schwefel function to make the problem a maximization problem. The function is defined by:

$$f([D_1, O_1, E_1]) = -\left[418.9829n - \sum_{i=1}^n |x_i| \times \sin(\sqrt{|x_i|})\right]$$

$$-500 \leq D_i, O_i \leq 500; 1 \leq i \leq 2; x_1 = D_1, x_2 = O_1, x_3 = E_1$$

For $x_i \in [-100, 100]$, Figure 7.2 shows the typical Griewank function plotted for $n=2$.

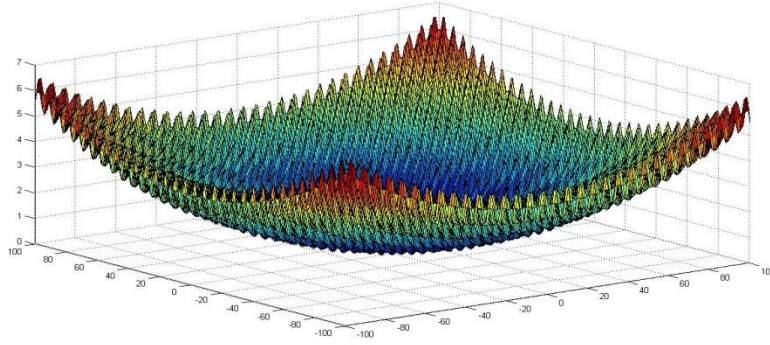


Figure 7.2. Griewank function in $[-100, 100]$ plotted using Matlab

7.4.5 Environment, design and operations all driving performance. The objective of the experiments was to not only determine the ability of GSN to effectively use grouping to converge on separate environments, but also to check the ability of SGA to operate on “noisy” life cycle data. Unlike the previous experiments and setups, two sets of experiments were carried here, each with a different representation of a solution. The Ackley function was first used to model the LCE relationship, with each solution candidate having five parameters, one for the environment, and two for each of the other kind. Then, a modified Schwefel function was used to model the LCE relationship, with each solution candidate encoded with six parameters, two for each kind.

The Schwefel function is a deceptive function that has its global minimum geometrically distant, over the parameter space, from the next best local minima. Therefore, with the Schwefel function, the search algorithms are potentially prone to convergence in the wrong direction. For $x_i \in [-100, 100]$, Figure 7.3 shows a plot of the Schwefel function. The Schwefel function is defined by:

$$f([D_1, D_2, O_1, O_2, E_1, E_2]) = -[418.9829n - \sum_{i=1}^n |x_i| \times \sin(\sqrt{|x_i|})]$$

$$-500 \leq D_i, O_i \leq 500; 1 \leq i \leq 2; x_1 = D_1, x_2 = D_2; x_3 = O_1, x_4 = O_2, x_5 = E_1, x_6 = E_2$$

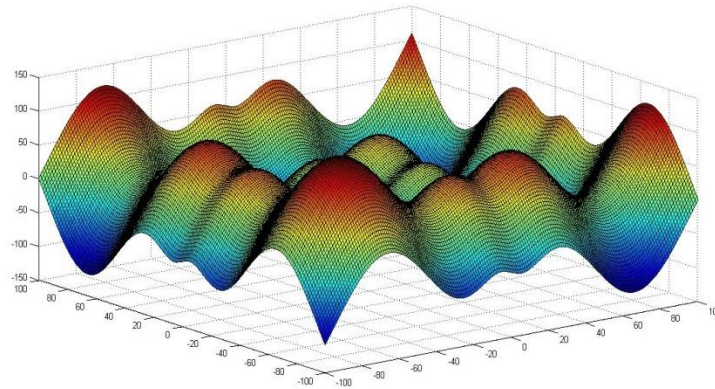


Figure 7.3. Schwefel function in [-100, 100] plotted using Matlab

On the other hand, the Ackley's function function is a widely used multimodal test function. It has the following definition:

$$f([D_1, D_2, O_1, O_2, E_1]) = -[-20 \times e^{-2\sqrt{\frac{1}{n} \times \sum_{i=1}^n x_i}} - e^{\sqrt{\frac{1}{n} \times \sum_{i=1}^n \cos(2\pi x_i)}} + 20 + e^1]$$

$$-500 \leq D_i, O_i \leq 500; 1 \leq i \leq 2; x_1 = D_1, x_2 = D_2, x_3 = O_1, x_4 = O_2$$

For $x_i \in [-25, 25]$, Figure 7.4 shows a plot of the Ackley's function.

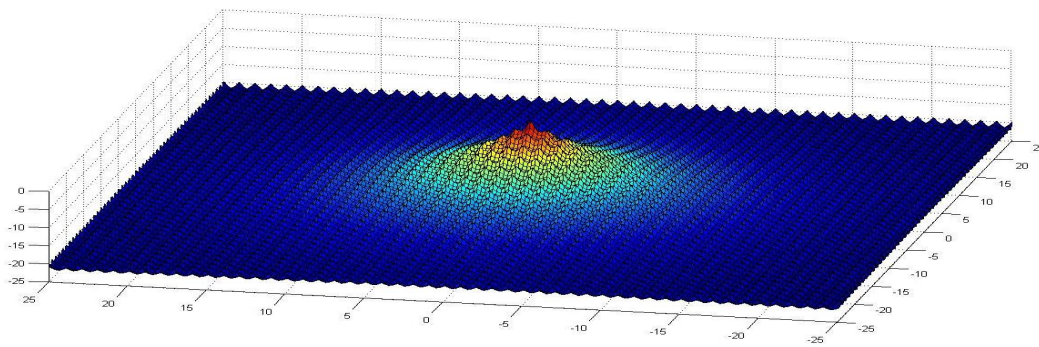


Figure 7.4. Ackley's function in [-25, 25] plotted using Matlab

7.5 Results and Interpretation

For each set of experiments that were carried out, results were collected and an interpretation provided. The following subsections show the typical results for each set of experiment.

7.5.1 Environment driving design for performance. Figure 7.5 shows a plot of the final performance values of the population for one experiment. The relative fitness values were used to plot the fitness of the population. Relative fitness values were used to give each individual as many relative fitness values as the number of existing themes. The relative fitness value of an individual to a group was calculated as a product of that individual (absolute) fitness with their membership level to that group. The group average fitness per generation was then calculated as the mean of the averaged fitness values of the whole population calculated theme wise.

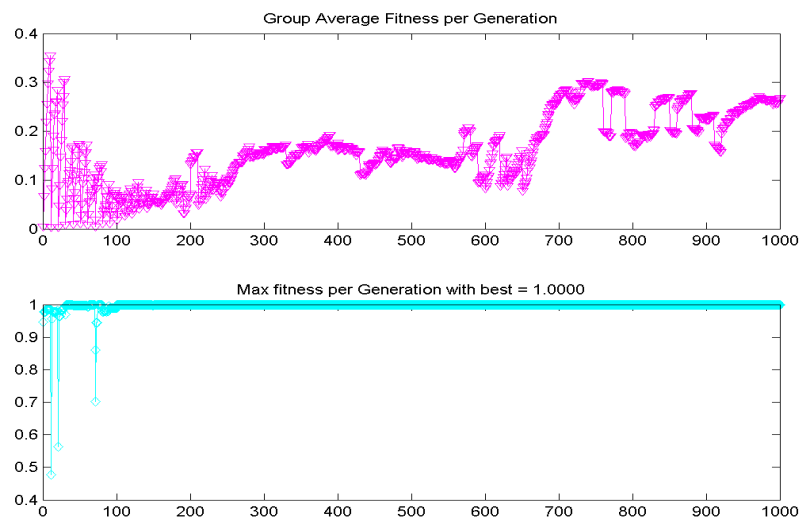


Figure 7.5. GSN results per generation for experiment set 1

Figure 7.5 shows that the population's average performance improved over generations. The improvements displayed in Figure 7.5 are not steady, proof of the impact of the membership level values. Table 7.2 shows the population converged. A lack of diversity can be observed, since that the convergence was to a single point for each environment.

Table 7.2

Sample of final population of GSN for experiment set 1

Population			
Design	Operational	Environmental	Fitness
350.0362	-246.5235797	A	0.999927602
150.0976	-171.6999546	B	0.999804854

From Table 7.3, it can be observed that the themes did not improve from start to finish. This was not expected since that the themes contributed to the search process.

Plotting the final population gave Figure 7.6. Nothing further can be inferred from the graph. The final population lack of diversity makes it impossible to capture the nature of the LCE parameters' relationship. The lack of diversity in the final population also makes the scaling of the final solutions hard to achieve.

Table 7.3

Starting and ending themes in experiment set 1

Themes Start			
Design	Operational	Environmental	Fitness
65.1214	0.4947	A	0.4302
265.8169	-92.3475	B	0.7683
120.0846	-113.8714	A	0.5401
80.1779	-286.7696	B	0.8603
-271.4478	40.3410	A	0.2428
Themes End			
Design	Operational	Environmental	Fitness
40.0775	380.4257	A	0.3801
-483.2775	-467.6816	B	0.2665
145.0424	336.2579	A	0.5900
49.9128	375.1393	B	0.79982
194.6495	369.2472	A	0.6892

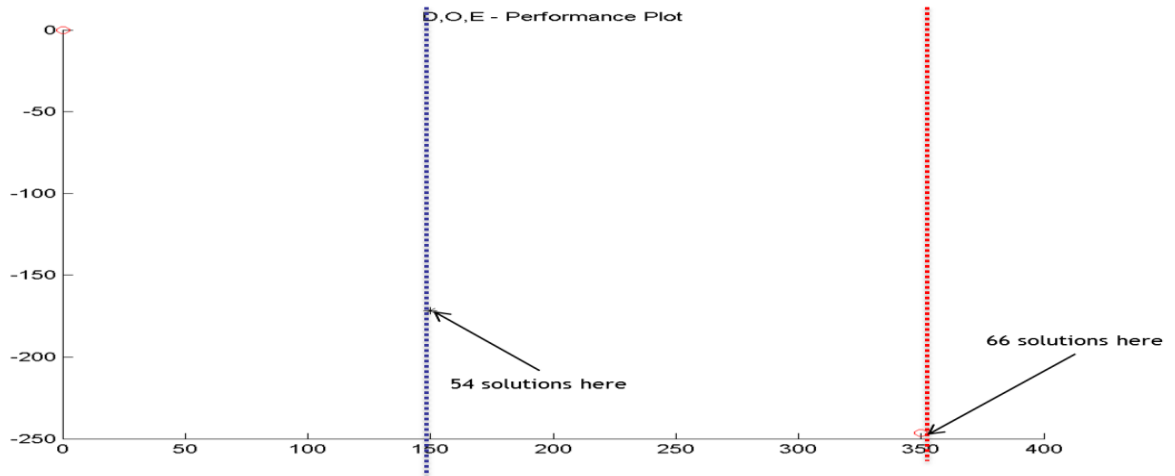


Figure 7.6. Final population plot for experiment set 1

7.5.2 Environment driving both design and operations for performance. Figure 7.7 shows a plot of the final results for one of the experiments. From Figure 7.7, the population's average performance can be seen to improve.

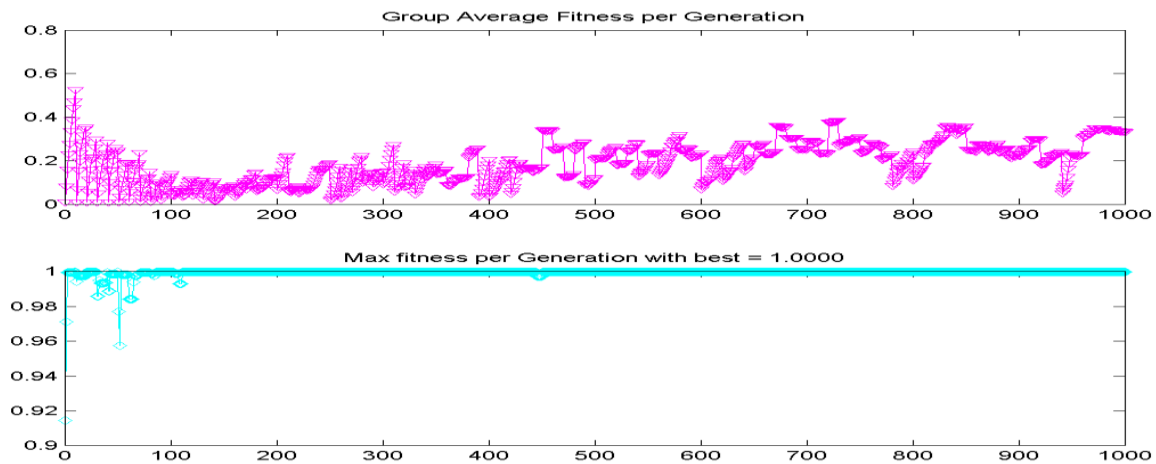


Figure 7.7. GSN results per generation for experiment set 2

Table 7.4 shows a sample of the final population. The population converged. Like with the previous experiment, a lack of diversity can be observed, since that the convergence was to a single point for each environment.

Table 7.4

Sample of final population for GSN in experiment set 2

Final Population Sample			
Design	Operational	Environmental	Fitness
-234.27425	-350	A	1
201.075351	50	B	1

Figure 7.8 shows a plot of the final population. The algorithm had an environment wise convergence. Although converging in this scenario is good, it does remove the diversity from GSN. The red dotted line represents the relationship of LCE parameters to performance in environment A, and the blue dotted line the same in environment B. From Figure 7.8, there is no way to tell or approximate the LCE parameters' relationship.

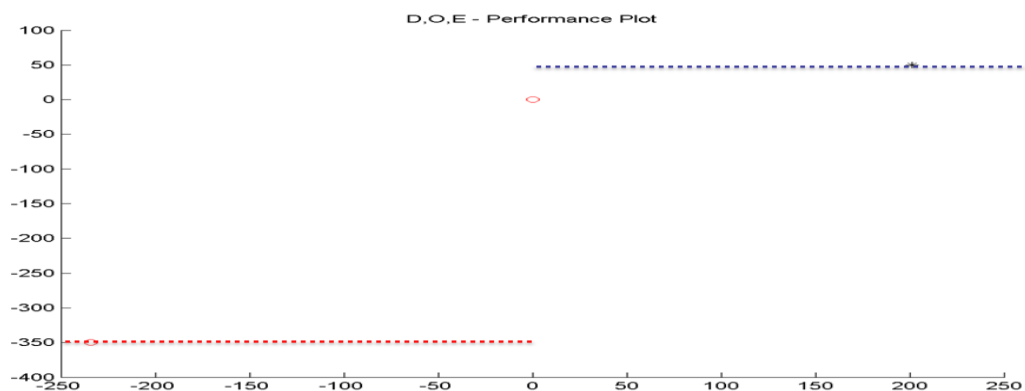


Figure 7.8. Final population plot for experiment set 2

7.5.3 Environment and design both driving operations for performance. Figure 7.9 shows a plot of the final results for one of the experiments.

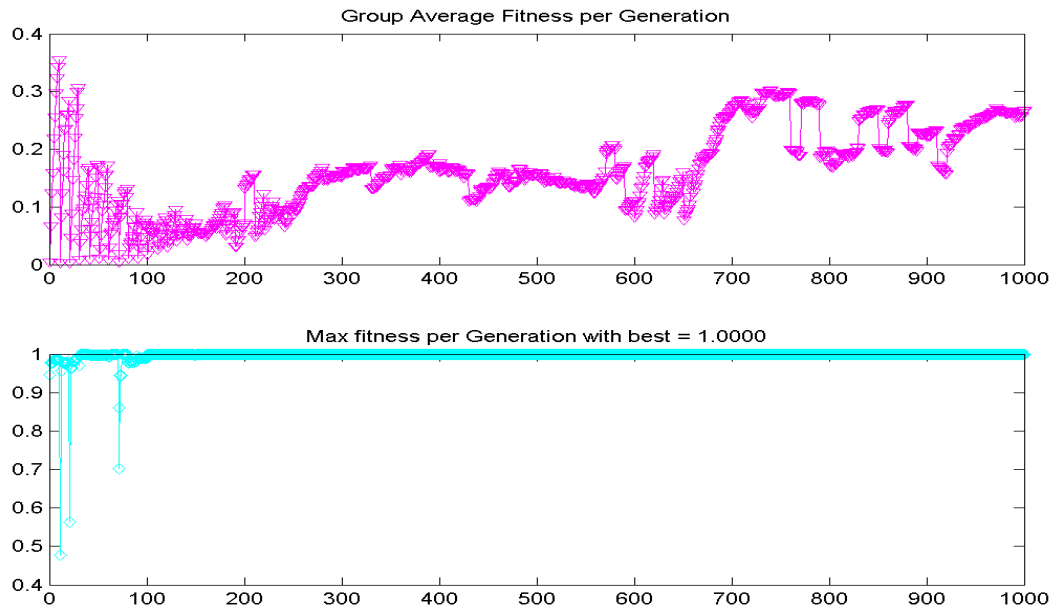


Figure 7.9. GSN results per generation for experiment set 3

From Figure 7.9, a trend similar to experiment set 1 and experiment set 2 is observable. A quick look at the final population show that the algorithm converged. Table 7.5 shows a sample of the final population.

Table 7.5

Sample of final population for SGN in experiment set 3

Population			
Design	Operational	Environmental	Performance
-350.049	-149.99653	A	1
-350	-147.21621	A	1
-102.348	-350	B	1
-102.348	-350	B	1

Figure 7.10 shows a plot of the final population. The algorithm had an environment wise convergence. Convergence removes the diversity from GSN. Without diversity in the final population, the scaling of the final solutions would be hard to achieve. There is no way to tell or

even approximate the LCE parameters' relationship. The trend so far seems to be that GSN would converge, and cannot be used to approximate an LCE parameters' relationship to performance.

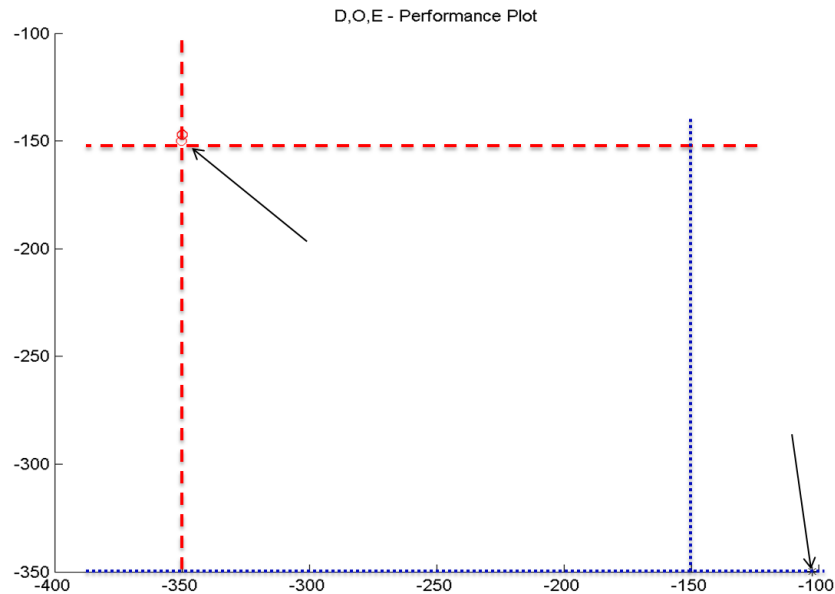


Figure 7.10. Final population plot for experiment set 3

7.5.4 Environment and the design both driving performance. As mentioned during the setup phase, two sets of experiments were carried out for this case.

7.5.4.1 Using Griewank to characterize LCE's relationship. The Griewank function was modified to turn the problem into a maximization problem with global best performance value being 0. Figure 7.11 shows a plot of the final results for one of the experiments.

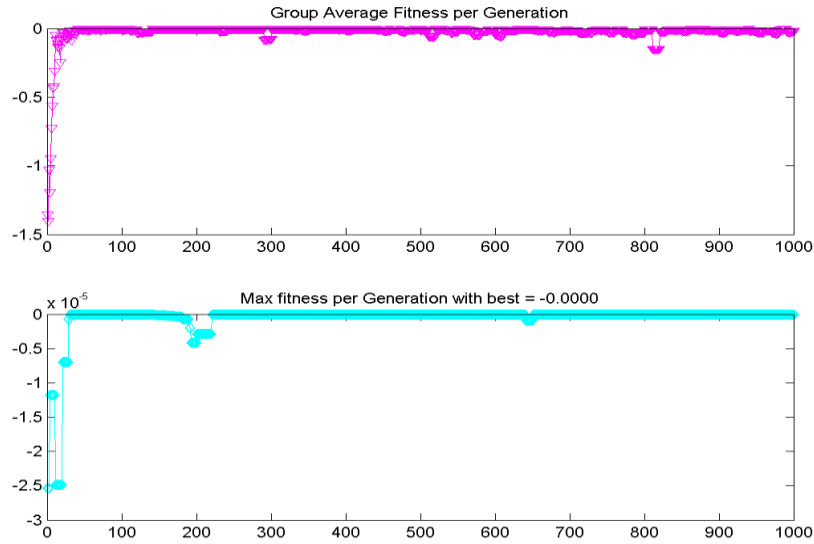


Figure 7.11. GSN results per generation for experiment set 4 – Griewank

There is a convergence after a steady but rapid average fitness growth. Table 7.6 shows a sample of the final population.

Table 7.6

Sample of final population for SGN in experiment set 4– Griewank

Final Population Sample			
Design	Operational	Environmental	Fitness
-0.05237	491.7339	A	-0.00137
-0.05237	491.7339	A	-0.00137
-3.13813	-314.11	B	-0.5588
-3.13813	-314.11	B	-0.5588

Plotting the final population yields Figure 7.12 where the narrow range of values for the design parameters shows the ability of GSN to resolve best solutions across and within environment’s boundaries. However, there is not enough diversity within the final solution to try and apportion the observed performance values to the solutions’ LCE parameters classes.

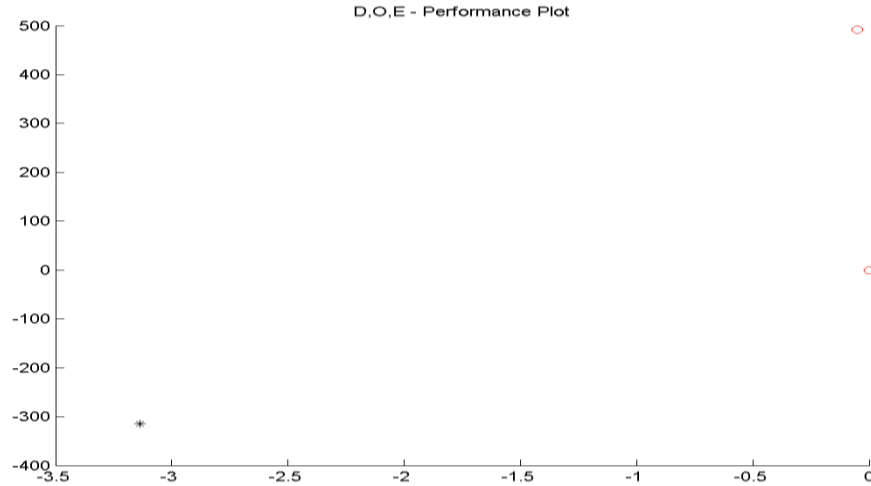


Figure 7.12. Final population plot for experiment set 4 – Griewank

7.5.4.2 Using Schwefel to characterize LCE's relationship. The Schwefel function was modified to turn the problem into a maximization problem with global best fitness value being 0. Figure 7.13 shows a plot of the final results for one of the experiments.

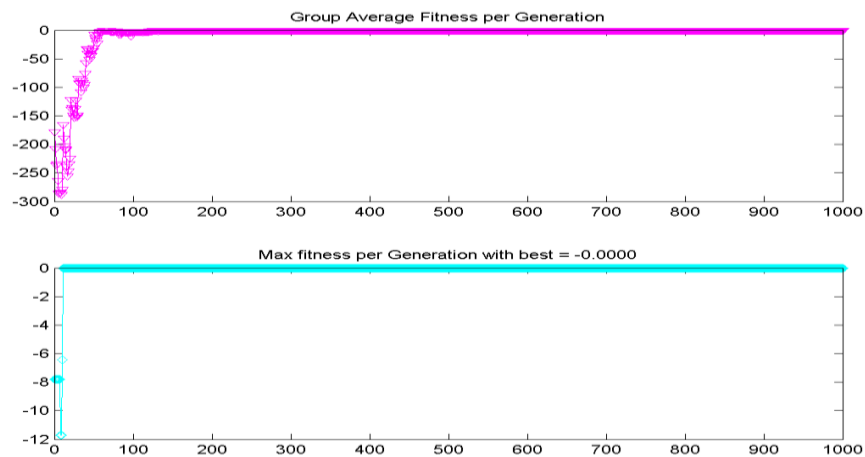


Figure 7.13. SGA results per generation for experiment set 4 – Schwefel

Figure 7.14 is the plot of the final population. The convergence of the algorithm is denoted by the concentration of the final solutions at two locations of the design-operational domain. However, population diversity was lost.

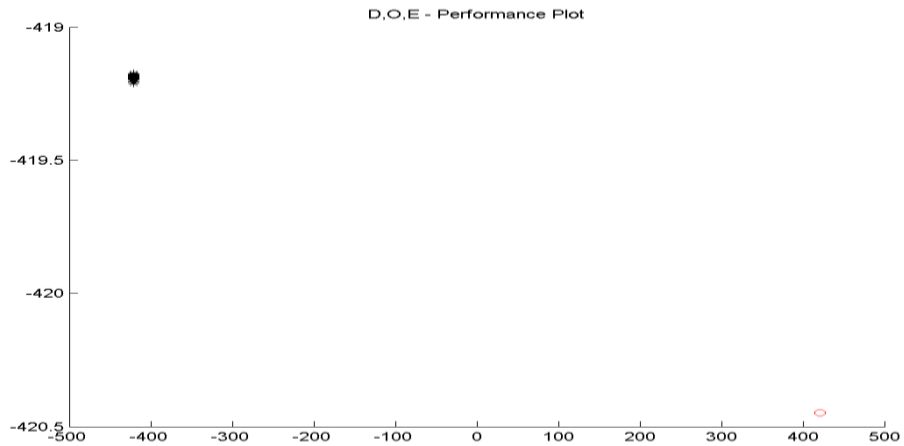


Figure 7.14. Final population plot for experiment set 4 – Schwefel

7.5.5 Environment, design and operations all driving performance. Performing experiments with more than three parameters and comparing the results obtained to other known GA-based approaches make possible seeing the potential benefit of grouping on GA, as well as the possible use of GSN as a general optimization method. GA is a trial and error method to problem solving. GA is solution-oriented and makes no attempt to discover why a solution works; merely that it is a solution. GSN on the other hand was built to not only discover solutions that work, but also solutions that are diverse so they can be grouped and scaled. The purpose of the experiments carried out in this section was to determine whether the mimicking of social interactions built into GSN make GSN a lesser metaheuristic performer when compared to other well-known GA derivatives. Two sets of experiments were carried out for this case.

7.5.5.1 Using Ackley to characterize LCE's relationship. The Ackley's function was modified to turn the problem into a maximization problem with global best performance value being 0. Figure 7.15 shows a plot of the final results for one of the experiments.

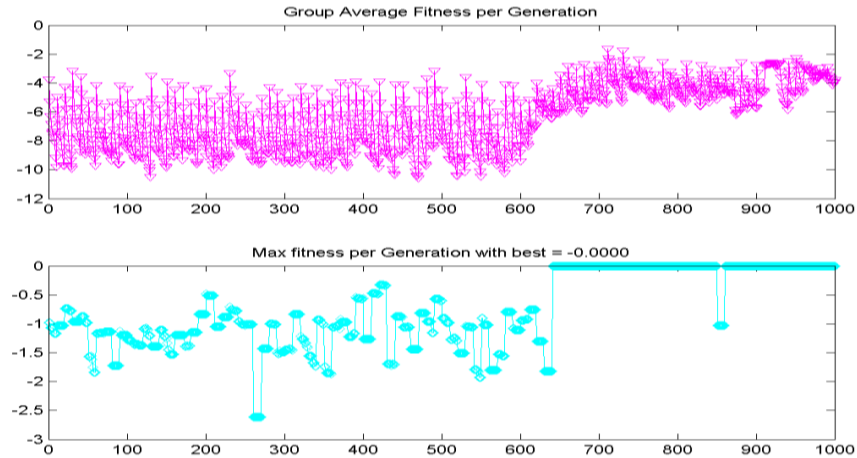


Figure 7.15. GSN results per generation for experiment set 5 – Ackley

From Figure 7.15, it can be seen that as the number of generations increases, population's average fitness first remains constant before starting to improve at around generation 600.

Although the absolute best solution (0) was not reached, the group average fitness generation shows that the relative fitness was close to the absolute best. Such a value for the relative contrasts with absolute fitness values from Table 7.7. The contrast shows the importance of the membership level values that can be leveraged via a modified aggregated performance measurement to achieve better results. A sample of the final population is displayed in Table 7.7.

Table 7.7

Sample of final population for GSN experiment set 5 – Ackley

Population					
Design	Design	Operation	Operation	Environment	Fitness
2.4066	-9.9765	-1.9909	-1.8854	A	-14.1942
-432.75	-257.25	-187.5	-492	B	-21.7183
-290	424	-314	-278	B	-20
-104	36	-180	-22	B	-20
364	36	340	-336	B	-20
2.4066	-9.9765	-1.9909	-1.8854	A	-14.1942

Unlike SGA when tested in the same conditions, GSN had a diversity built into the final population. The final population could be used for some further analysis.

The implementation of parallel GA (PGA) that was made had a total of five subpopulations evolving separately from one another. Table 7.8 represents a sample of the final population obtained after running PGA. Like GSN, the final population in PGA has diversity.

Table 7.8

Sample of final population for PGA experiment set 5 – Ackley

Population					
Design	Design	Operational	Operational	Environment	Fitness
-164.799	-481.572	143.7725	-50.9452	B	-21.5891
-150.984	-129.014	45.69213	-48.8014	B	-21.0889
-152.78	42.47536	-0.91078	-143.042	B	-21.4305
134.262	-24.4091	-17.4822	3.941685	A	-21.9349
-10.1991	-158.589	19.10951	-50.6574	B	-21.7926
157.0182	180.3656	-10.7803	246.5109	A	-21.8312
-110.008	9.519213	-55.0603	157.9034	A	-21.167
32.89862	-79.0304	-54.6876	297.3046	A	-21.4126
-166.391	-3.13498	328.9591	-93.6539	A	-21.6443
48.71317	-351.099	128.035	-93.0879	B	-20.8908
18.68839	88.73163	-156.407	-70.0866	B	-21.8296
-247.845	216.9395	3.951581	229.2697	A	-20.9307

Figure 7.16 shows the evolution per generation of the PGA implementation. Besides subpopulation 1, all other subpopulations appear to perform exactly the same.

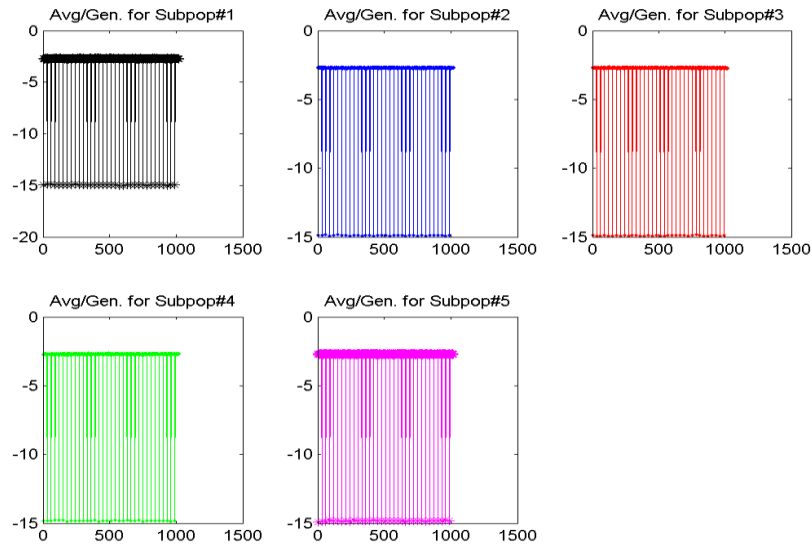


Figure 7.16. Average population fitness over time with PGA for experiment set 5 – Ackley

The implementation of island GA (IGA) that was made had a total of five subpopulations evolving on separate islands from one another. A migration operator was available to IGA that every 30 generations, would allow the islands to exchange in a round-robin way, 5 of their best solutions as a way to re-introduce diversity within all the islands. Table 7.9 represents a sample of the final population obtained after running IGA. Like GSN, the final population in IGA also has some diversity.

Figure 7.17 shows the evolution per generation of the IGA implementation. The graph is similar to that of PGA with a lot of jumps within the average fitness, characteristic of a noisy environment.

Table 7.9

Sample of final population for IGA experiment set 5 – Ackley

Population					
Design	Design	Operational	Operational	Environment	Fitness
59.17157	149.4894	78.91444	-44.9946	-A	-21.3225
-51.3367	67.59259	-58.8961	-185.948	421	-21.6164
-6.18678	-127.84	-0.44204	14.86595	-A	-21.5407
-16.9806	-66.4336	377.3741	-265.759	A	-21.85
36.25633	149.859	-100.201	-98.6796	A	-21.5935
-124.87	-63.334	6.866562	-211.447	A	-21.7417
79.23545	-131.754	-107.063	-82.734	A	-21.4536
-37.0189	-45.1076	-44.081	-148.812	B	-20.588
-153.066	-290.224	62.93955	-140.625	A	-21.3334
38.14309	60.99853	407.2399	-113.165	A	-20.9871
-135.449	-44.9966	239.352	46.58049	B	-22.0175
149.4617	104.3652	152.6735	55.39732	A	-22.2333
-29.057	-43.0327	261.2078	341.5205	A	-21.3733
11.56537	-149.069	-154.211	-148.015	A	-21.3592
106.5239	128.9473	-398.941	-297.036	A	-21.1254
-48.8613	233.3871	28.19601	-116.926	A	-21.3975

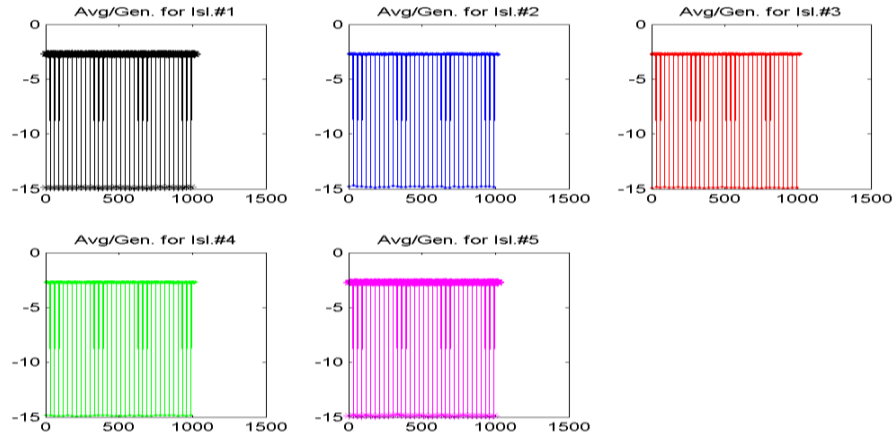


Figure 7.17. Average population fitness over time with IGA for experiment set 5 – Ackley

7.5.5.2 Using Schwefel to characterize LCE's relationship. The Schwefel function was modified to turn the problem into a maximization problem with global best performance value being 0 with six parameters. Figure 7.18 shows a plot of the final results for one of the experiments.

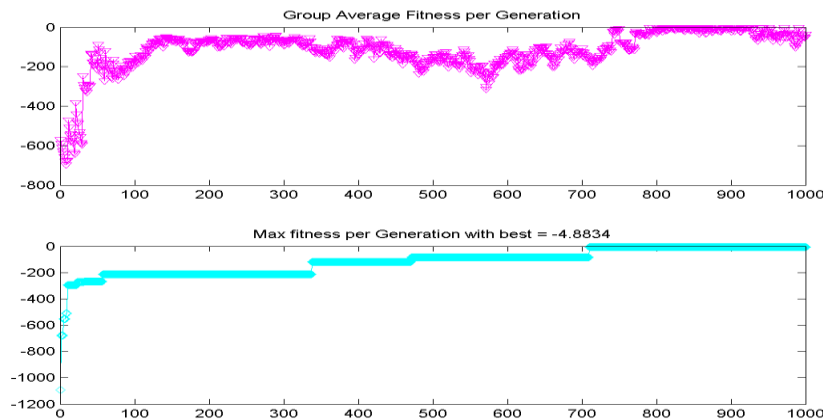


Figure 7.18. GSN results per generation for experiment set 5 – Schwefel

From Figure 7.18, the population is observed to be constantly improving. Within the [500 500] range, the Schwefel function, as used, is known to converge ($f(x^*) = 0$) for $x^* = (s_1, s_2, \dots, s_n)$ where $s_i = \pm 421$ for $1 \leq i \leq n$. Table 7.10 is a sample of the final population. Two observations are important here: first, the final population does not have much diversity built in it, and second,

the final population appears not be fall away from one of the known optimum solution of the problem.

Table 7.10

Sample of final population for GSN experiment set 5 – Schwefel

Population						
Design	Design	Operation	Operation	Environment	Environment	Performance
435.8014	-407.731	-421.017	447.0183	A1	A2	-140.2422491
435.803	-407.726	-420.994	447.0146	A1	A2	-138.5087513
434.8028	-410.375	-434.876	449.2655	B1	B2	-170.8471386
435.5377	-408.429	-424.676	447.6116	B1	B2	-142.7093332

The same implementation of parallel GA (PGA) was modified to use a phenotype of length six and was run with subpopulations evolving separately from one another. Table 7.11 represents a sample of the final population obtained after running PGA. Unlike SGA, the final population in PGA has a lot more diversity. Figure 7.19 shows the evolution per generation of the PGA implementation.

The same implementation of island GA (IGA) was modified to use a phenotype of length six and was run with subpopulations evolving separately from one another. Table 7.12 represents a sample of the final population obtained after running IGA. Unlike SGA, the final population in IGA has a lot more diversity.

Table 7.11

Sample of final population for PGA experiment set 5 – Schwefel

Population						
Design	Design	Operation	Operation	Environment	Environment	Fitness
-110.764	110.6206	114.1634	-122.78	B1	B2	-2103.88
-25.7738	-152.613	168.5419	-135.198	B1	B2	-1773.22
-149.203	-41.7851	150.3702	-21.3514	A1	A2	-1786.03
19.25329	-467.823	-85.7605	110.0545	-A1	A2	-1610.85

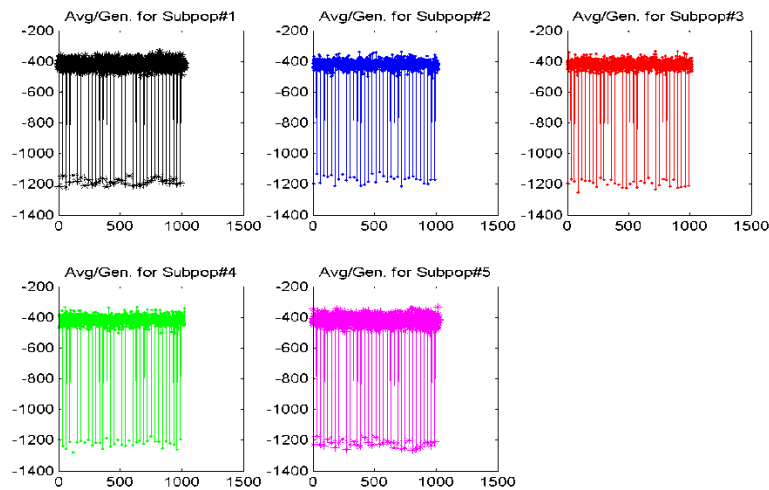


Figure 7.19. Average population fitness over time with PGA for experiment set 5 – Schwefel

Table 7.12

Sample of final population for IGA experiment set 5 – Schwefel

Population						
Design	Design	Operation	Operation	Environment	Environment	Fitness
75.329	-167.453	-262.074	300.293	A1	A2	-1984.768
-33.249	154.567	341.833	-63.650	B1	B2	-1770.517
83.422	-122.429	294.779	-45.117	B1	B2	-2048.165
-186.50	175.472	-61.005	206.677	A1	A2	-1138.464
-18.631	-8.129	-133.941	98.169	B1	B2	-1848.615
-364.10	107.688	443.436	101.915	B1	B2	-1386.967
-138.43	119.805	-320.32	224.222	A1	A2	-2005.699
130.081	-346.562	-287.362	133.558	A1	A2	-2260.600
13.845	-251.76	-202.456	-190.149	A1	A2	-1342.991
258.964	55.81715	-328.877	-61.453	A1	A2	-1875.343
52.933	228.1491	-420.826	121.588	A1	A2	-1204.704
40.352	-220.977	-66.664	98.310	A1	A2	-1491.028
29.213	58.4483	-39.935	-114.615	A1	A2	-1749.626
493.048	385.762	-107.189	5.968	B1	B2	-1588.027

Figure 7.20 shows the evolution per generation of the IGA implementation. The jumpy fitness evaluations were not present in GSN.

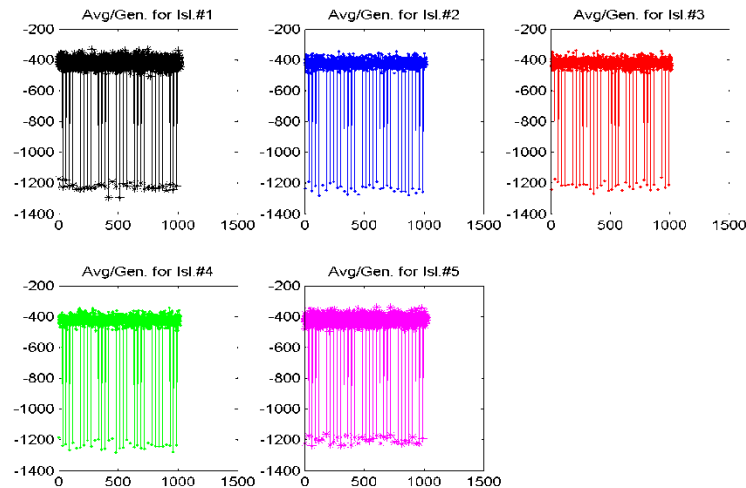


Figure 7.20. Average population fitness over time with IGA for experiment set 5 – Schwefel

7.6 Summary

Within this chapter, genetic social networks (GSN) were applied to the generalized life cycle product design (GLPD) approach of continuous product design. Simulations of GSN were carried out according to a set of factors including the relationship between LCE parameters, the number of parameters, and the shape of the search domain. The quality of the solutions returned by GSN was assessed using different assessment criteria including their ability to capture the LCE parameters' relationship to performance, as well as their ability to scale.

A first set of experiments (1-4) was successfully carried to better understand how GSN works on different types of environments, and how the GSN built-in social interactions mimicking shapes both the search process and the interpretation of the returned results. Whether the nature of the LCE parameters' relationship was simple or not, GSN appears to converge to two solutions, one for each environment. Using both absolute and relative fitness values in GSN enabled a better understanding of the meaning of membership levels. The final themes, and the final membership level values, both outputs of GSN, could be used to inform about ways the observed LCE parameters can be tuned to increase the performance of a live system. Assuming

for instance that the life cycle aggregated performance measurement is defined as a linear map of selected LCE parameters, and then the membership level values could be used to scale their corresponding solutions in order to achieve the best possible performance.

A last (5th) set of experiments that was carried to assess the impact of social networking on the GA search process by comparing GSN to both parallel GA (PGA) and island GA (IGA) proved conclusive. The performance of GSN was better when compared to that of PGA, IGA, and SGA. However, as the population would improve in GSN, the same trend was absent from the themes. Therefore, although they contribute to the optimization process by helping the GSN pick the leaders, the themes appear to only drive the optimization process.

Overall, the GSN would leverage the connections of each solution to each group to generate system efficiency. By having leaders and themes both influencing individuals, yet “moving” at a less frequent pace than individuals, GSN converged by mimicking the concept of positive deviance. Table 7.13 summarizes the results of all the experiments.

Table 7.13

Summary of experiments on GSN

Linear & Polynomial Performance	Griewank & Schwefel Performance	Ackley & Schwefel Performance
Convergence observed within population	Convergence observed within population	Performs better than SGA, IGA and PGA on optimization
Delimited characterization of LCE because of convergence	Delimited characterization of LCE because of convergence	Delimited characterization of LCE because of convergence
Themes and leaders updates reflected by average fitness change pattern	Themes and leaders updates reflected by average fitness change pattern	
	Performed better than SGA in terms of the quality of found solutions	

Table 7.14 summarizes the results of the GSN algorithm compared to those obtained by SGA, IGA and PGA for the same experiment.

Table 7.14

Comparative results of GSN, SGA, IGA, and PGA

Algorithm	Ranking the Achieved Best by Algorithm and Environment			
	Ackley Performance		Schwefel Performance	
	Environment A	Environment B	Environment A	Environment B
SGA	2	2	2	2
GSN	1	1	1	1
PGA	3	4	4	3
IGA	4	3	3	4

Chapter 8

Conclusion

The work presented in this dissertation addressed the modeling and the application of biomimetic metaheuristics to product life cycle engineering. A broader classification of lifecycle data was suggested. The generalized life cycle product design (GLPD) model, generic model for sustainable continuous product design was presented. Two new metaheuristics that are GA-based, using the concepts of fish schooling (SGA) or the concepts of social network dynamics (GSN) were presented, implemented and applied to GLPD. The basic functionality of both SGA and GSN for GLPD was assessed using a design of experiments.

From a LCE relationship standpoint, SGA performed better than GSN by being able to capture patterns of good performance from life cycle data. A solution quality metric named trait performance indexes (TPI) was defined and used with SGA. The use of TPI values demonstrated their ability, in some cases, to help the product designer decide whether to maintain the permissible values for a life cycle parameter, or to change them without getting a life cycle performance hit. The use of TPI could also help a designer decide whether to turn an operational parameter into a design parameter or vice versa. A non-parametric clustering method named the geometrically expanded membership for automated clustering (GEMAC) was created and used with SGA.

GSN converges most of the time, even when used within an environment with a high density of deceptive attractors (Griewank or Ackley). The final themes, and the final membership level values, both outputs of GSN, could be used to inform about ways the observed LCE parameters can be tuned to increase the performance of a live system.

Both presented metaheuristic methodologies have their strengths and limitations as listed throughout Chapter 5 and Chapter 7. The limitations can either come from the way life cycle fitness is calculated, or come from the way either method uses grouping to drive optimization.

8.1 Contributions

The intellectual contributions of the dissertation are listed as follow:

- A sustainable continuous product design approach called GLPD was devised and presented. The approach is generic enough to be applied to existing methodologies.
 - The approach claims a top-down-up approach, and takes from both bottom-up, and top-down methodologies
 - The approach turns design problems into optimization problems
- A GA-based approach to optimization named SGA, that mimics fish schooling, was presented. SGA was built for the GLPD approach. The ability of SGA to deal with grouping, and unconstrained optimization was tested and the results presented.
- A GA-based approach to optimization named GSN, that mimics social networks' interactions, was presented. GSN was built for the GLPD approach. The ability of GSN to deal with grouping, and unconstrained optimization was tested and the results presented.

8.2 Future Directions

This research opens many opportunities for future research, as there is still a lot that can be learned on SGA or GSN either by tuning their respective parameters to make them more problem-specific or by applying to new types of problem to further study the impact of grouping.

Following are a couple of suggestions for derivative and exploratory work that can be undertaken to improve the work performed:

- Improve the performance of SGA by tuning some of its parameters (tabu distance, behavior threshold values, metrics, clustering...). A fine-tuning could yield to better performance with heterogeneous (from different environments) groups.
- Apply SGA or GSN to stochastic optimization problems, and determine when their use is appropriate. The work presented here just showed that either methodology (SGA and GSN) deals with unconstrained optimization problem in the worst case scenario as better as other forms of GA-based metaheuristics.
- Apply SGA and GSN to the study and life cycle-based design of an actual product or service. Although the work presented was about using either methodology for continuous product design, it would be interesting to see them being used on a real product or service design.

References

- Akanle, O. M., & Zhang, D. Z. (2008). Agent-based model for optimising supply-chain configurations. *International Journal of Production Economics*, 115(2), 444-460. doi: <http://dx.doi.org/10.1016/j.ijpe.2008.02.019>
- Albuquerque, P., & Nevskaya, Y. (2012). *The Impact of Innovation on Product Usage: A Dynamic Model with Progression in Content Consumption*. Working Paper. Simon Graduate School of Business. University of Rochester, New York.
- Allenby, B. R. (1992). *Design for Environment: Implementing Industrial Ecology*. (Doctorate of Philosophy), Rutgers University, New Brunswick, NJ.
- Anastas, P., & Warner, J. (2000). *Green chemistry: theory and practice*: Oxford University Press.
- Asiedu, Y., & Gu, P. (1998). Product life cycle cost analysis: state of the art review. *International Journal of Production Research*, 36(4), 883-908.
- AT&L, D. (2012). *DoD Maintenance FACT BOOK*. http://www.acq.osd.mil/log/mpp/factbooks/DoD_Maintenance_Fact_Book_2012.pdf: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics.
- Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., . . . Zdravkovic, V. (2008). Empirical investigation of starling flocks: a benchmark study in collective animal behaviour. *Animal Behaviour*, 76(1), 201-215. doi: 10.1016/j.anbehav.2008.02.004
- Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., . . . Zdravkovic, V. (2008). Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *PNAS*, 105(4), 1232–1237.

- Bapuji, H., & Beamish, P. W. (2008). Avoid Hazardous Design Flaws Retrieved 24 September, 2012, from <http://hbr.org/2008/03/avoid-hazardous-design-flaws/ar/1>
- Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, 286(5439), 509-512. doi: 10.1126/science.286.5439.509
- Barabási, A. L., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., & Vicsek, T. (2002). Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, 311(3-4), 590-614. doi: [http://dx.doi.org/10.1016/S0378-4371\(02\)00736-7](http://dx.doi.org/10.1016/S0378-4371(02)00736-7)
- Benyus, J. (1997). *Biomimicry: innovation inspired by nature*: William Morrow.
- Berrah, L., Mauris, G., & Vernadat, F. (2004). Information aggregation in industrial performance measurement: rationales, issues and definitions. *International Journal of Production Research*, 42(20), 211-225.
- Blanchard, B. S., & Fabrycky, W. J. (2011). *Systems Engineering And Analysis* (Fifth ed.). Upper Saddle River, NJ: Pearson Prentice-Hall.
- Blizzard, J. L., & Klotz, L. E. (2012). A framework for sustainable whole systems design. *Design Studies*, 33(5), 456-479. doi: 10.1016/j.destud.2012.03.001
- Bloom, N. (2005). *Reliability Centered Maintenance (RCM) : Implementation Made Simple: Implementation Made Simple*: Mcgraw-hill.
- Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 11(4), 14-24.
- Boehm, B. (2000). *Spiral Development: Experience, Principles, and Refinements Spiral Development Workshop*. Pittsburgh: CMU/SEI-2000-SR-008.
- Boguna, M., Romualdo, P.-S., Diaz-Guilera, A., & Arenas, A. (2003). Emergence of clustering, correlations, and communities in a social network model. *Physical Review E* 70, 056122.

- Bonabeau, E., & Dagorn, L. (1995). Possible universality in the size distribution of fish schools. *PHYSICAL REVIEW. E. STATISTICAL PHYSICS, PLASMAS, FLUIDS, AND RELATED INTERDISCIPLINARY TOPICS*, 51(6). doi: citeulike-article-id:1128133
- Bosner, R. (2006). Patented biologically-inspired technological innovations: a twenty year view. *Journal of Bionic Engineering*, Vol 3, 39-41.
- Bosner, R., & Vincent, J. (2006). Technology trajectories, innovation, and the growth of biomimetics. *Journal of Mechanical Engineering Science*, Vol 221, 1177-1180.
- Braungart, M., McDonough, W., & Bollinger, A. (2007). Cradle-to-cradle design: creating healthy emissions – a strategy for eco-effective product and system design. *Journal of Cleaner Production*, 15(13–14), 1337-1348. doi: 10.1016/j.jclepro.2006.08.003
- Cantú-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms*. Dordrecht: Kluwer Academic.
- Cerný, V. (1985). A thermodynamical approach to the traveling salesman problem. *Journal of Optimization Theory and Applications*, vol. 45(no. 1), pp. 41–51.
- Cha, S.-H. (2007). Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES*, 1(4), 300-307.
- Chen, C.-C., & Chuang, M.-C. (2008). Integrating the Kano model into a robust design approach to enhance customer satisfaction with product design. *International Journal of Production Economics*, 114(2), 667-681. doi: <http://dx.doi.org/10.1016/j.ijpe.2008.02.015>

- Clark, T., & Charter, M. (1999). Eco-design Checklists for Electronic Manufacturers, Systems Integrators, and Suppliers of Components and Sub-assemblies. The Centre for Sustainable Design.
- Collins, M., & Brebbia, C. (2004). *Design and nature II: comparing design in nature with science and engineering*: Wessex Institute of Technology Press.
- Couzin, I. D., Krause, J., James, R., Ruxton, G. D., & Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218, 1-11. doi: 10.1006/jtbi.2002.3065
- Davis, L. (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.
- DEFENSE, O. O. T. U. S. O. (1996). *DoD Guide to Integrated Product and Process Development*. ACQUISITION AND TECHNOLOGY Retrieved from <https://www.acquisition.gov/sevensteps/library/dod-guide-to-integrated.pdf>.
- Dhillon, B. S. (2006). *Maintainability, Maintenance, and Reliability for Engineers*: Taylor & Francis.
- Dorigo, M., Maniezzo, V., & Coloni, A. (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26(no. 1), 29–41.
- Dorigo, M., & Caro, G. D. (1999). *The Ant Colony Optimization meta-heuristic*. London, UK: D. Corne et al.
- Dorigo, M., Caro, G. D., & Gambardella, a. L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, vol. 5(no. 2), pp. 137–172.
- Dorigo, M., & Stützle, T. (2003). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. *In Handbook of metaheuristics*, 250-285.

- Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. Cambridge, MA: MIT Press.
- Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique. *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*.
- Duda, M., & Shaw, J. (1997). Life cycle assessment. *Society*, 35(1), 38-43. doi: 10.1007/s12115-997-1054-x
- Elimam, A. A., & Dodin, B. (2013). Project scheduling in optimizing integrated supply chain operations. *European Journal of Operational Research*, 224(3), 530-541. doi: <http://dx.doi.org/10.1016/j.ejor.2012.09.007>
- ElMaraghy, H. (2007). Reconfigurable Process Plans For Responsive Manufacturing Systems. In P. Cunha & P. Maropoulos (Eds.), *Digital Enterprise Technology* (pp. 35-44): Springer US.
- EPA. (17 October 2010). Defining Life Cycle Assessment (LCA).
- Eubank, S., Guclu, H., Kumar, V. A., Marathe, M. V., Srinivasan, A., Toroczkai, Z., & Wang, N. (2004). Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988), 180-184.
- Fakhim, B., Behnia, M., Armfield, S. W., & Srinarayana, N. (2011). Cooling solutions in an operational data centre: A case study. *Applied Thermal Engineering*, 31(14–15), 2279-2291. doi: 10.1016/j.applthermaleng.2011.03.025
- Fiksel, J. (1996). *Design for Environment: Creating Eco-efficient Products and Processes*. New York: McGraw-Hill.
- Fogel, L. J., Owens, A. J., & Walsh, a. M. J. (1966). *Artificial Intelligence Through Simulated Evolution*: John Wiley & Sons.

- Forbes, P. (2005). *The Gecko's foot: bio-inspiration, engineering new materials and devices from nature*: Harper Collins.
- Forsberg, K., & Mooz, H. (October 1991). *The Relationship of System Engineering to the Project Cycle*. Paper presented at the Proceedings of the National Council for Systems Engineering (NCOSE), Chattanooga, Tennessee.
- Forsberg, K., & Mooz, H. (1997). *Visualizing System Engineering and Project Management as an Integrated Process*. Paper presented at the Proceedings of the International Council for Systems Engineering (INCOSE) Conference, Los Angeles, CA.
- Ghasimi, S. A., Ramli, R., & Saibani, N. A genetic algorithm for optimizing defective goods supply chain costs using JIT logistics and each-cycle lengths. *Applied Mathematical Modelling*(0). doi: <http://dx.doi.org/10.1016/j.apm.2013.08.023>
- Giudice, F., La Rosa, G., & Risitano, A. (2005). Materials selection in the Life-Cycle Design process: a method to integrate mechanical and environmental performances in optimal choice. *Materials & Design*, 26(1), 9-20. doi: 10.1016/j.matdes.2004.04.006
- Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing*, vol. 1(no. 3).
- Glover, F. (1990). Tabu search—part II. *ORSA Journal on Computing*, vol. 2(no. 1), 4–32.
- Glover, F., & Laguna, a. M. (1997). *Tabu Search*: Kluwer Academic Publishers.
- Goel, P. S., & Singh, N. (1998). Creativity and innovation in durable product development. *Computers & Industrial Engineering*, 35 (1-2), 5-8.
- Goldberg, D. E. (1994). *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass [u.a.]: Addison-Wesley.
- Goss, S., Aron, S., Deneubourg, J.-L., & Pasteels, J.-M. (1989). Self-organized shortcuts in the Argentine ant,. *Naturwissenschaften*, 76, 579-581.

- Graedel, T., Allenby, B., & Combrie, P. (1995). Matrix approaches to abridged life-cycle assessment. *Environmental Science and Technology in Society*, 29 (3), 134-139.
- Granovetter, M. (1973). The strength of weak ties. *American Journal of Sociology*, 78, 1360-1380.
- Griewank, A. O. (1981). Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34(1), 11-39. doi: 10.1007/bf00933356
- Griffin, A. (1997a). PDMA research on new product development practices: updating trends, and benchmarking best practices. *The Journal of Product Innovation Management*, 14, 429-458.
- Griffin, A. (1997b). Modeling and measuring product development cycle time across industries. *Journal of Engineering and Technology Management*, 14(1), 1-24. doi: 10.1016/s0923-4748(97)00004-0
- Gupta, Y. (1983). Life Cycle Cost Models and Associated Uncertainties. In J. K. Skwirzynski (Ed.), *Electronic Systems Effectiveness and Life Cycle Costing* (Vol. 3, pp. 535-549): Springer Berlin Heidelberg.
- Hamerly, G. (2010). *Making k-means even faster*. Paper presented at the In SIAM International Conference on Data Mining (SDM).
- Hassan, M. K., Azubir, N. A. M., Nizam, H. M. I., Toha, S. F., & Ibrahim, B. S. K. K. (2012). Optimal Design of Electric Power Assisted Steering System (EPAS) Using GA-PID Method. *Procedia Engineering*, 41(0), 614-621. doi: <http://dx.doi.org/10.1016/j.proeng.2012.07.220>
- Helms, M., Vattam, S. S., & Goel, A. K. (2009). Biologically inspired design: process and products. *Design Studies*, 30(5), 606-622. doi: 10.1016/j.destud.2009.04.003

- Hemelrijk, C. K., & Hildenbrandt, H. (2008). Self-Organized Shape and Frontal Density of Fish Schools. *Ethology*, *114*, 245–254. doi: 10.1111/j.1439-0310.2007.01459.x
- Hemelrijk, C. K., & Hildenbrandt, H. (2012). Schools of fish and flocks of birds: their shape and internal structure by self-organization. *Interface Focus*. doi: 10.1098/rsfs.2012.0025
- Hemelrijk, C. K., Hildenbrandt, H., Reinders, J., & Stamhuis, E. J. (2010). Emergence of Oblong School Shape: Models and Empirical Data of Fish. *Ethology*, *116*(11), 1099-1112. doi: 10.1111/j.1439-0310.2010.01818.x
- Henley, E. J., & Kumamoto, H. (1985). *Design for Reliability and Safety Control*. Upper Saddle River, NJ: Pearson Printice-Hall.
- Hock, T. (1997). Integrated product development. *Sadhana*, *22*(2), 189-198. doi: 10.1007/bf02744488
- Holland, J. (1975). *Adaptation in Natural and Artificial Systems*.
- Hong-Bae Jun, Dimitris Kiritsis, & Xirouchakis, P. (2007). Research issues on closed-loop PLM. *Computers in Industry*, *58*, 855–868. doi: 10.1016/j.compind.2007.04.001
- Horner, K. (1993). Methodology as a Productivity Tool. In J. Keyes (Ed.), *Software Productivity Handbook* (pp. 97-117). New York, NY: Windcrest: McGraw-Hill.
- Im, I., Hong, S., & Kang, M. S. (2011). An international comparison of technology adoption: Testing the UTAUT model. *Information & Management*, *48*(1), 1-8. doi: 10.1016/j.im.2010.09.001
- Inada, Y. (2000). Steering mechanism of fish schools. *Complexity International*, *8*.
- INCOSE. (2006). *SYSTEMS ENGINEERING HANDBOOK A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES*.

- Janz, D., Sihm, W., & Warnecke, H. J. (2005). Product Redesign Using Value-Oriented Life Cycle Costing. *CIRP Annals - Manufacturing Technology*, 54(1), 9-12. doi: [http://dx.doi.org/10.1016/S0007-8506\(07\)60038-9](http://dx.doi.org/10.1016/S0007-8506(07)60038-9)
- Jardim-Goncalves, R., Grilo, A., & Steiger-Garcia, A. (2006). Challenging the interoperability between computers in industry with MDA and SOA. *Computers in Industry*, 57(8-9), 679-689. doi: 10.1016/j.compind.2006.04.013
- Jenkins, M. (2013). Innovate or Imitate? The Role of Collective Beliefs in Competences in Competing Firms. *Long Range Planning*(0). doi: <http://dx.doi.org/10.1016/j.lrp.2013.04.001>
- Jeong, K.-Y. (2000). Conceptual frame for development of optimized simulation-based scheduling systems. *Expert Systems with Applications*, 18(4), 299-306. doi: [http://dx.doi.org/10.1016/S0957-4174\(00\)00011-7](http://dx.doi.org/10.1016/S0957-4174(00)00011-7)
- Kabak, Ö., & Ülengin, F. (2011). Possibilistic linear-programming approach for supply chain networking decisions. *European Journal of Operational Research*, 209(3), 253-264. doi: <http://dx.doi.org/10.1016/j.ejor.2010.09.025>
- Keeney, R. L., & Lilién, G. L. (1987). New industrial product design and evaluation using multiattribute value analysis. *Journal of Product Innovation Management*, 4(3), 185-198. doi: 10.1016/0737-6782(87)90003-8
- Kennedy, J. (1997, 13-16 Apr 1997). *The particle swarm: social adaptation of knowledge*. Paper presented at the Evolutionary Computation, 1997., IEEE International Conference on.
- Kennedy, J., & Eberhart, R. (1995, Nov/Dec 1995). *Particle swarm optimization*. Paper presented at the Neural Networks, 1995. Proceedings., IEEE International Conference on.

- Kirkpatrick, S., Jr., C. D. G., & Vecchi, a. M. P. (1983). Optimization by simulated annealing. *Science*, vol. 220, pp. 671–680.
- Knight, P., & Jenkins, J. O. (2009). Adopting and applying eco-design techniques: a practitioners perspective. *Journal of Cleaner Production*, 17(5), 549-558. doi: 10.1016/j.jclepro.2008.10.002
- Kopácsi, S., Kovács, G., Anufriev, A., & Michelini, R. (2007). Ambient intelligence as enabling technology for modern business paradigms. *Robotics and Computer-Integrated Manufacturing*, 23(2), 242-256. doi: 10.1016/j.rcim.2006.01.002
- Krause, E. F. (1987). *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*: Dover Publications.
- Kühme, T. (1993). User-centered approach to adaptive interfaces. *Knowledge-Based Systems*, 6(4), 239-248. doi: 10.1016/0950-7051(93)90015-1
- Kumar, A. V. (2005). Introduction. In F. Kreith & D. Y. Goswami (Eds.), *The CRC Handbook of Mechanical Engineering* (Second ed., pp. 1576-1577): CRC Press LLC.
- Larrosa-Guerrero, A., Scott, K., Head, I. M., Mateo, F., Ginesta, A., & Godinez, C. (2010). Effect of temperature on the performance of microbial fuel cells. *Fuel*, 89(12), 3985-3994. doi: 10.1016/j.fuel.2010.06.025
- Lavie, T., & Meyer, J. (2010). Benefits and costs of adaptive user interfaces. *International Journal of Human-Computer Studies*, 68(8), 508-524. doi: 10.1016/j.ijhcs.2010.01.004
- Lehto, M. R., Landry, S. J., & Buck, J. (2007). *Introduction to Human Factors and Ergonomics for Engineers*: Taylor & Francis.

- Lin, H. K., Harding, J. A., & Shahbaz, M. (2004). Manufacturing system engineering ontology for semantic interoperability across extended project teams. *International Journal of Production Research*, 42 (24), 5099-5118.
- Lourenço, H. R., Martin, O., & Stutzle, a. T. (2002). Iterated local search. In F. G. a. G. Kochenberger (Ed.), *Handbook of Metaheuristics*, (Vol. 57, pp. 321–353): Kluwer Academic Publishers.
- Low, K. H. (2009). Preface: Why biomimetics? *Mechanism and Machine Theory*, 44(3), 511-512. doi: 10.1016/j.mechmachtheory.2008.11.008
- Luke, S., & Spector, L. A comparison of crossover and mutation in genetic programming.
- Mahfoud, S. W. (1995). *A comparison of parallel and sequential niching methods*. Paper presented at the Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA).
- Massarutto, A., Carli, A. d., & Graffi, M. (2011). Material and energy recovery in integrated waste management systems: A life-cycle costing approach. *Waste Management*, 31(9–10), 2102-2111. doi: <http://dx.doi.org/10.1016/j.wasman.2011.05.017>
- Mayyas, A. T., Qattawi, A., Mayyas, A. R., & Omar, M. A. (2012). Life cycle assessment-based selection for a sustainable lightweight body-in-white design. *Energy*, 39(1), 412-425. doi: 10.1016/j.energy.2011.12.033
- Mengshoel, O. J., & Goldberg, D. E. (2008). The Crowding Approach to Niching in Genetic Algorithms. *Evolutionary Computation*, 16(3), 315-354. doi: 10.1162/evco.2008.16.3.315
- Michalewicz, Z., & Fogel, D. B. (2004). *How to Solve It: Modern Heuristics*. Germany: Springer.

- Mill, F., & Sherlock, A. (2000). Biological analogies in manufacturing. *Computers in Industry*, 43(2), 153-160. doi: 10.1016/s0166-3615(00)00064-6
- Milligan, G., & Cooper, M. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2), 159-179.
- Morris, R. (2009). *The Fundamentals of Product Design*.
- Moyle, P. B., & Joseph J. Cech, J. (2004). *Fishes: An Introduction to Ichthyology* (5 ed.): Benjamin Cummings.
- Mueller, K. G., & Besant, C. B. (1999). *Streamlining life cycle analysis: a method*. Paper presented at the Proceedings of the First International Symposium on Environmentally Conscious Design and Inverse Manufacturing, Tokyo.
- MWG. (2010). INCOSE Systems Engineering Measurement Primer v2.0. In INCOSE (Ed.), *A Basic Introduction to Measurement Concepts and Use for Systems Engineering*. San Diego, CA: International Council on Systems Engineering.
- Neely, A. (1998). *Measuring Business Performance*. Economist Books.
- Nelson, B., Wilson, J., & Yen, J. (2009, 18-21 Oct. 2009). *A study of biologically-inspired design as a context for enhancing student innovation*. Paper presented at the Frontiers in Education Conference, 2009. FIE '09. 39th IEEE.
- Nock, R., & Nielsen, F. (2006). On Weighting Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8), 1-13.
- Ohno, K. (2011). The optimal control of just-in-time-based production and distribution systems and performance comparisons with optimized pull systems. *European Journal of Operational Research*, 213(1), 124-133. doi: <http://dx.doi.org/10.1016/j.ejor.2011.03.005>

- Oyewole, S. A., Haight, J. M., & Freivalds, A. (2010). The ergonomic design of classroom furniture/computer work station for first graders in the elementary school. *International Journal of Industrial Ergonomics*, 40(4), 437-447. doi: 10.1016/j.ergon.2010.02.002
- Ozcan, E., & Mohan, C. K. (1999, 1999). *Particle swarm optimization: surfing the waves*. Paper presented at the Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on.
- Pal, N. R., & Biswas, J. (1997). Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6), 847-857. doi: 10.1016/s0031-3203(96)00127-6
- Parrish, J. K., Viscido, S. V., & Grunbaum, D. (2002). Self-organized fish schools: an examination of emergent properties. *BIOLOGICAL BULLETIN- MARINE BIOLOGICAL LABORATORY, VOL 202; PART 3*, 296-305.
- Partridge, B. L. (1980). The effect of school size on the structure and dynamics of minnow schools. *Animal Behaviour*, 28(1), 68-IN63. doi: 10.1016/s0003-3472(80)80009-1
- Partridge, B. L. J. (1982). The structure and function of fish schools. *Scientific American*, 246.
- Partridge, B. L. J., Johansson, J., & Kalisk, J. (1983). Structure of schools of giant bluefin tuna in Cape Cod Bay. *Environ. Biol. Fish.*, 9, 253–262.
- PILCHER, J. J., NADLER, E., & BUSCH, C. (2002). Effects of hot and cold temperature exposure on performance: a meta-analytic review. *ERGONOMICS*, 45(10), 682-698.
- Pitcher, T. J. (2001). FISH SCHOOLING *Encyclopedia of Ocean Sciences* (Vol. 2, pp. 975–987): Elsevier Ltd.
- Ramze Rezaee, M., Lelieveldt, B. P. F., & Reiber, J. H. C. (1998). A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, 19(3–4), 237-246. doi: 10.1016/s0167-8655(97)00168-2

- Rechenberg, I. (1973). *Evolutionsstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Information*. Freiburg, Germany.
- Roozenburg, N., & Eekels, J. (1995). *Product Design: Fundamentals and Methods*. Chichester: Wiley.
- Ross, P., & Corne, D. (1994). Applications of Genetic Algorithms. *AISB Quarterly* 89, 23-30.
- Royce, W. W. (1970). Amplify Learning *Managing the Development of Large Software Systems*.
- Sääksvuori, A., & Immonen, A. (2008). Product Life Cycle Management Retrieved from <http://www.springerlink.com/content/V01R625382465324> doi:10.1007/978-3-540-78172-1_1
- Saaty, R. W. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, 9(3–5), 161-176. doi: 10.1016/0270-0255(87)90473-8
- Saaty, T. L. (1982). *Decision making for leaders: The analytical hierarchy process for decisions in a complex world*. Belmont, California: Lifetime Learning Publications.
- Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, 48(1), 9-26. doi: 10.1016/0377-2217(90)90057-i
- Saaty, T. L. (2005). Analytic Hierarchy Process *Encyclopedia of Biostatistics*: John Wiley & Sons, Ltd.
- Sarkar, B. K., Mandal, P., Saha, R., Mookherjee, S., & Sanyal, D. (2013). GA-optimized feedforward-PID tracking control for a rugged electrohydraulic system design. *ISA Transactions*, 52(6), 853-861. doi: <http://dx.doi.org/10.1016/j.isatra.2013.07.008>
- Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.*: John Wiley & Sons.

- Sedighzadeh, D., & Masehian, E. (2009). An particle swarm optimization method, taxonomy and applications. *Proceedings of the international journal of computer theory and engineering*, 5, 486-502.
- Seltzer, L. (2012). Is Apple Or NFC The Bigger Loser With iPhone 5? Retrieved 24 September, 2012, from <http://www.informationweek.com/byte/personal-tech/wireless/is-apple-or-nfc-the-bigger-loser-with-ip/240007606?queryText=iphone%20%20nfc>
- Spears, W. M. (1992). Crossover or mutation. *Foundations of genetic algorithms 2*, 221-237.
- Stanfield, P. M., King, R. E., & Joines, J. A. (1996). Scheduling arrivals to a production system in a fuzzy environment. *European Journal of Operational Research*, 93(1), 75-87. doi: [http://dx.doi.org/10.1016/0377-2217\(95\)00117-4](http://dx.doi.org/10.1016/0377-2217(95)00117-4)
- Stark, J. (2011). Product Lifecycle Management (pp. 1-16): Springer London.
- Suh, N. P. (1999). Engineering Design. In F. Kreith (Ed.), *The CRC Handbook of Mechanical Engineering* (pp. 2-17). Boca Raton: CRC Press LLC.
- Sumpter, D. J. T. (2006). The principles of collective animal behaviour. *Philosophical Transaction of Royal Society B*, 361, pp. 5-22. doi: 10.1098/rstb.2005.1733
- Technologies, I. (2012). Functions and Features Retrieved 22 September 2012, from http://www.inclusive.com/mmr/findings/functions_and_features.htm
- Toivonen, R., Kovanen, L., Kivelä, M., Onnela, J.-P., Saramäki, J., & Kaski, K. (2009). A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks*, 31(4), 240-254. doi: 10.1016/j.socnet.2009.06.004
- Toulabi, M. R., Shiroei, M., & Ranjbar, A. M. (2014). Robust analysis and design of power system load frequency control using the Kharitonov's theorem. *International Journal of*

Electrical Power & Energy Systems, 55(0), 51-58. doi:

<http://dx.doi.org/10.1016/j.ijepes.2013.08.014>

The University of Reading: What is Biomimetics? (Retrieved June 5, 2012). from

<http://www.reading.ac.uk/biomimetics/about.htm>

Verma, D., Farr, J., & Johannesen, L. H. (2003). System training metrics and measures: A key operational effectiveness imperative. *Systems Engineering*, 6(4), 238-248. doi: 10.1002/sys.10047

Verma, D., & Gallois, B. (2001). *Graduate Program in System Design and Operational Effectiveness (SDOE): Interface between developers/providers, and users/consumers*. Paper presented at the International Conference on Engineering Design (ICED), Glasgow, United Kingdom.

Vincent, J., Bogatyreva, O., & Bogatyrev, N. (2007). Towards a theory of biomimetics. *Comparative Biochemistry and Physiology - Part A: Molecular & Integrative Physiology*, 146(4, Supplement), S129. doi: 10.1016/j.cbpa.2007.01.241

Vincent, J. F. V., Bogatyreva, O. A., R., B. N., Adrian, B., & Pahl, A.-K. (2006). Biomimetics: its practice and theory. *Journal of The Royal Society Interface*, 3(9), 471-482. doi: 10.1098/rsif.2006.0127

Vogel, S. (2000). *Cat's paws and catapults: mechanical worlds of nature and people*.

Wallace, D. R., Jakiela, M. J., & Flowers, W. C. (1996). Design search under probabilistic specifications using genetic algorithms. *Computer-Aided Design*, 28(5), 405-421. doi: [http://dx.doi.org/10.1016/0010-4485\(95\)00059-3](http://dx.doi.org/10.1016/0010-4485(95)00059-3)

- Wanko, P. T., & Stanfield, M. P. (2012). *Adaptive Metaheuristics with Schooling Genetic Algorithms*. Paper presented at the Industrial and Systems Engineering Research Conference, Orlando, Florida.
- Whitley, D., Rana, S., & Heckendorn, R. B. (1998). The Island Model Genetic Algorithm: On Separability, Population Size and Convergence. *Journal of Computing and Information Technology*, 7, 33-47.
- Wiendahl, H. P., ElMaraghy, H. A., Nyhuis, P., Zäh, M. F., Wiendahl, H. H., Duffie, N., & Brieke, M. (2007). Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals - Manufacturing Technology*, 56(2), 783-809. doi: 10.1016/j.cirp.2007.10.003
- Wong, L. H., Pattison, P., & Robins, G. (2006). A spatial model for social networks. *Physica A: Statistical Mechanics and its Applications*, 360(1), 99-120. doi: 10.1016/j.physa.2005.04.029
- Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., . . . Steinberg, D. (2008). Top 10 Algorithms in Data Mining. *Knowledge and Information Systems*, 14(1), 1-37.
- Wu, Z.-j., Li, L.-z., Chen, Y., & Cai, Y. (2010, 17-19 Nov. 2010). *User's behavior -based creative product design process*. Paper presented at the Computer-Aided Industrial Design & Conceptual Design (CAIDCD), 2010 IEEE 11th International Conference on.
- Zahedi, F. (1986). The Analytic Hierarchy Process—A Survey of the Method and its Applications. *Interfaces*, 16(4), 96-108. doi: 10.1287/inte.16.4.96

Appendix A

This appendix contains the main algorithm for SGA. All the files that are referenced follow starting with Appendix E. to run the code for all 5 sets of experiments that were carried for the dissertation, some minor changes need to be made to this file and all the supporting files. Just follow the comments left within each file. The name of the following file is “sga.m.”

```
%reset workspace and memory

clear global variables;

clear all;

clc;

global pop envStart envEnd offset nbrAlleles NbrOidx domainRange tolerance tabu
tabuDistance tabuLen tabuIdx limits ext stop nc;

stop = 0;

stop_count = 0;

tabuIdx=0;

tolerance=1e-11;

nbrAlleles=3; %each chromosome will be encoded using nbrAlleles alleles

tabuDistance=5*nbrAlleles; %minimum distance for tabu

ext=100;%500;

%Sets number of each parameter

nDp=1;

nOp=1;

nEp=1;

%generate field data, this is where different types of relation are tested

[X,Y] = meshgrid(-ext:1:ext);

[Z minZ maxZ] = performance(X,Y);

nbr_repetitions=1;
```

```

max_iterations=1020;% 100
maxSameMaxPeriod=3;
exactIterCount = 0;
past_avg = 0.0;
tabuLen=round(max_iterations*.20);% 20% of # of iterations
tabu=zeros(tabuLen,nbrAlleles);
tabuF=zeros(tabuLen,1);
fromAnBnTots = zeros(tabuLen,3);
GemacPeriod=10; % determines how often GEMAC runs.
%for behavior logging, use the following: 1==FF;2==SM;3==PA
nbrIter_AvgFitness_tConvg_nSchools_reptNbr=zeros(max_iterations,3,nbr_repetitions);
sizStaFitMax=cell(1,max_iterations);
pop_size=120;
envAtot=round(0.55*pop_size);% 55% of population comes from environment A
envBtot=pop_size-envAtot;% 45% of population comes from environment B
nbr_children=round(0.50*pop_size);% 50% of pop size
center=[0 0];% cartesian is [101,101] Matlab
envStart=0;
envEnd=6;
domainStart=-ext;
domainEnd=+ext;
offset=ext+1;
domainRange=domainEnd-domainStart+1;
limits=[domainStart domainEnd; domainStart domainEnd; envStart envEnd];% we are in dim=3
sigma=domainRange/6;
%behavior cutoff values
lw_cut=0.80;% 20% under averaged mean

```

```

hg_cut=1.05;%5% over averaged mean

pop=zeros(pop_size+2*nr_children,nbrAlleles+2);%2 - |schID|fit)

for i=1:nbr_repetitions

    sgafig = figure('Name','SGA TEST | In Progress Solutions', 'NumberTitle', 'off', 'Units',
'normalized', 'Position', [10 10 90 90]);

    array1=normrnd(center(1,1),sigma,[1,pop_size]);
    array2=normrnd(center(1,2),sigma,[1,pop_size]);
    envID=randperm(pop_size);

    %makes sure we start with feasible solutions, and correct the offset
    for h=1:pop_size

        pop(h,1)=array1(h)-domainRange*round(array1(h)/domainRange);
        pop(h,2)=array2(h)-domainRange*round(array2(h)/domainRange);

        %randomly assign population to environments
        if envID(h)<= envAtot

            pop(h,nbrAlleles)=envStart;%environment A
        else

            pop(h,nbrAlleles)=envEnd;%environment B
        end
    end
end

nmaxes=zeros(1,max_iterations);

%%%%%%oIndexes(1,1:pop_size+2*nr_children)=1:1:pop_size+2*nr_children;

j=0;

while and(j < max_iterations, stop == 0)

    NbrOidx=pop_size;

    %creates schools of fish

    [IDX nc]=gemac();

```

```

period=0;
while period < GemacPeriod;
    t = cputime;
    period=period+1;
    j=j+1;
    nbrIter_AvgFitness_tConvg_nSchools_reptNbr(j,3,i)=nc;
    sizStaFitMax_temp = struct('data','data'); %creates a 1-by-1 structure with no fields.
    data_tmp=zeros(4,nc);

    schools_statuses=zeros(1,nc);
    sum_perf=zeros(1,nc);
    mean_perf=zeros(1,nc);
    school_cm=zeros(nc,nbrAlleles);%But, do not include the environment since it is discrete
    max_perf=zeros(1,nc);
    fitnesses=zeros(NbrOidx,nc);
    fromAnBnTot=zeros(nc,3);
    %neatly repack pop and fitnesses from returned GEMAC output
    indexes=zeros(1,nc);
    whereat=zeros(pop_size+2*nbr_children,nc);
    for k1=1:NbrOidx;
        indexes(IDX(k1))=indexes(IDX(k1))+1;
        %copy performance and school ID the fish belongs to
        pop(k1,nbrAlleles+1:nbrAlleles+2)=[IDX(k1),performance(pop(k1,1:nbrAlleles))];
        sum_perf(1,IDX(k1))=sum_perf(1,IDX(k1))+pop(k1,nbrAlleles+2);
        fitnesses(indexes(IDX(k1)),IDX(k1))=pop(k1,nbrAlleles+2);
        school_cm(IDX(k1),1:nbrAlleles-1)=school_cm(IDX(k1),1:nbrAlleles-1)+pop(k1,1:nbrAlleles-1);
        whereat(indexes(IDX(k1)),IDX(k1))=k1;
    end
end

```



```

    if pop(k1,nbrAlleles+2)>max_perf(IDX(k1));
        max_perf(IDX(k1))=pop(k1,nbrAlleles+2);
    end
end

% When computing breeding proportion per school -- always round
% whether minimizing or maximizing, we only want to deal with
% numbers >0
for k2=1:nc
    school_cm(k2,:)=school_cm(k2,+)/indexes(k2);
    fromAnBnTot(k2,3)=indexes(k2);
    for k21=1:indexes(k2)
        if pop(whereat(k21,k2), nbrAlleles)==envStart
            fromAnBnTot(k2,1)=fromAnBnTot(k2,1)+1;
        else
            fromAnBnTot(k2,2)=fromAnBnTot(k2,2)+1;
        end
    end
end
school_cm(k2,nbrAlleles)=envStart;
opt1=performance(school_cm(k2,));
school_cm(k2,nbrAlleles)=envEnd;
opt2=performance(school_cm(k2,));
%the average of the cluster is a weighted average

mean_perf(1,k2)=(opt1*fromAnBnTot(k2,1)+opt2*fromAnBnTot(k2,2))/fromAnBnTot(k2,3);
if fromAnBnTot(k2,1)>=fromAnBnTot(k2,2)
    school_cm(k2,nbrAlleles)=envStart;%envEnd;
end

```

```

    end
end
%Determines best and puts it inside (recency) tabu if not yet in already
is_there=0;
[v i33]=max(mean_perf);
for k22=1:min(tabuLen,tabuIdx+1)
    %Close to there is as good as being there
    if (similarity('cityblock',school_cm(i33,:),tabu(k22,:)) <= tabuDistance)
        is_there=1;
        break;
    end
end
end
if is_there == 0 %insert in tabu list and record best for tpi
    tabu(mod(tabuIdx,tabuLen)+1,:)=school_cm(i33,:);
    tabuF(mod(tabuIdx,tabuLen)+1,1)=performance(school_cm(i33,:));
    fromAnBnTots(mod(tabuIdx,tabuLen)+1,:)=fromAnBnTot(i33,:);
    tabuIdx=tabuIdx+1;
end
nmaxes(1,j)=max(max_perf);
nbr_breed=pop_size+nr_children-NbrOidx;
mean_schools=mean(mean_perf);
ratios=sum_perf.*indexes;
totalRatio=sum(ratios);
breed_per_school=round(nbr_breed*ratios./totalRatio);
%in case some sums were negative while others were positive
for ps=1:length(breed_per_school)
    if breed_per_school(ps)<0

```

```

        breed_per_school(ps)=0;
    end
end
%Check whether a stopping criteria has been reached
if (abs(past_avg-mean_schools)<=tolerance)
    stop_count = stop_count +1;
    if stop_count == maxSameMaxPeriod
        exactIterCount = j;
        stop = 1;
    end
else
    past_avg = mean_schools;
    stop_count = 0;
end
% we are only interested in the behavior of schools that can breed
for k3=1:nc
    if breed_per_school(k3)<1; continue; end
    %%if indexes(k3)==1; schools_statuses(k3)=3; continue; end
    if mean_perf(k3)<=lw_cut*mean_schools
        schools_statuses(k3)=3; continue;%Predator Avoidance
    end
    if mean_perf(k3)>=hg_cut*mean_schools
        %schools_statuses(k3)=1;%Food Foraging
        %school is not close to tabu location and food still
        % available for consumption by the fish
        [bool pos]=isClear(school_cm(k3,:));
        % If food has depleted, then it is time to move
    end
end

```

```

if or(abs(pos-mod(tabuIdx,tabuLen)) < 05*GamacPeriod, pos==0)
    schools_statuses(k3)=1;%Food Foraging
else %get away from tabu location
    schools_statuses(k3)=3;%Predator Avoidance
end
else %default status
    schools_statuses(k3)=2;%School maintenance
end
end
end
%record data for all schools and store them
data_tmp(1,:)=indexes;%stores number of fish per school
data_tmp(2,:)=schools_statuses;%stores statuses of all schools
data_tmp(3,:)=mean_perf;%stores avg perf of each school
data_tmp(4,:)=max_perf;%stores the best performer per school
sizStaFitMax_temp.data=data_tmp;
sizStaFitMax{1,j}=sizStaFitMax_temp;
%Execute behaviors
for k4=1:nc
    if breed_per_school(1,k4)==0; continue; end %No need
    switch schools_statuses(k4)
        case 1 %Food Foraging – crossover rate > mutation rate
            wtpc=selection(0,breed_per_school(1,k4),fitnesses(1:indexes(k4),k4),0);
            for k41=1:2:breed_per_school(1,k4)
child=crossOver(pop(whereat(wtpc(k41),k4),:),pop(whereat(wtpc(k41+1),k4),:));
                NbrOidx=NbrOidx+1;
                pop(NbrOidx,:)=child;
                indexes(k4)=indexes(k4)+1;
            end
        end
    end
end

```

```

%           whereat(indexes(k4),k4)=NbrOidx;
end
case 2 %School Maintenance – crossover rate = mutation rate
start1=indexes(k4);
wtpc=selection(0,round(breed_per_school(1,k4)/2),fitnesses(1:indexes(k4),k4),0);
for k42=1:2:round(breed_per_school(1,k4)/2)

child=crossOver(pop(whereat(wtpc(k42),k4),:),pop(whereat(wtpc(k42+1),k4),:));
NbrOidx=NbrOidx+1;
pop(NbrOidx,:)=child;
%           indexes(k4)=indexes(k4)+1;
%           whereat(indexes(k4),k4)=NbrOidx;
end
wtpm=selection(0,round(breed_per_school(1,k4)/2),fitnesses(1:start1,k4),1);
for k42=1:round(breed_per_school(1,k4)/2)
%school_maintenance=max(1.0-mean_perf(k4)/nmaxes(1,j),howFar)

child=mutation(pop(whereat(wtpm(k42),k4),:),max(tolerance,mean_perf(k4)/nmaxes(1,j)));
%child=mutation(pop(whereat(wtpm(k42),k4),:),randint(1,1,[round((1.0-
mean_perf(k4)/mean_schools)*10000)
round((mean_perf(k4)/mean_schools)*10000)]/(10000)*domainRange);

%child=mutation(pop(whereat(wtpm(k42),k4),:),max(howFar2,howFar)*domainRange);
NbrOidx=NbrOidx+1;
pop(NbrOidx,:)=child;
%           indexes(k4)=indexes(k4)+1;
%           whereat(indexes(k4),k4)=NbrOidx;
end
case 3 %Predator Avoidance – Mutation precedes Crossover

```

```

start=NbrOidx+1;
start2=indexes(k4);
%Prob Select = f (fitness, distance from CM)
%Farther from CM & higher fitness = higher prob
%First, compute all distances to school CM
dist2CM=zeros(indexes(k4),1);
for k423=1:indexes(k4)

dist2CM(k423,1)=similarity('cityblock',school_cm(k4,:),pop(whereat(k423,k4),1:nbrAlleles));

end

%Compute the vector for probabilities
vect4prob=dist2CM.*fitnesses(1:indexes(k4),k4);
wtpm2=selection(0,round(breed_per_school(1,k4)/2),vect4prob,1);
for k43=1:round(breed_per_school(1,k4)/2)
    child=mutation(pop(whereat(wtpm2(k43),k4),:),max(tolerance,
mean_perf(k4)/nmaxes(1,j)));
    NbrOidx=NbrOidx+1;
    pop(NbrOidx,:)=child;
%
    indexes(k4)=indexes(k4)+1;
%
    whereat(indexes(k4),k4)=NbrOidx;
end

%Crossover: 1st parent comes from pool of mutant fish

wtpc1=selection(0,round(breed_per_school(1,k4)/2),pop(start:NbrOidx,nbrAlleles+2),1);

%Crossover: 2d parent comes from remaining school
wtpc2=selection(0,round(breed_per_school(1,k4)/2),fitnesses(1:start2,k4),1);
for k44=1:round(breed_per_school(1,k4)/2)
    child=crossOver(pop(start+wtpc1(k44)-1,:),pop(whereat(wtpc2(k44),k4),:));

```

```

        NbrOidx=NbrOidx+1;
        pop(NbrOidx,:)=child;
%         indexes(k4)=indexes(k4)+1;
%         whereat(indexes(k4),k4)=NbrOidx;
    end
    otherwise
        error('This should never occur')%nothing to be done
    end
end
%split population by environment -- to maintain ratio
%keeps population size constant
%First, remove duplicates if any
pop=unique(pop(1:NbrOidx,:), 'rows');
NbrOidx=size(pop,1);
popA = pop(pop(1:NbrOidx,nbrAlleles)==envStart,:);
popB = pop(pop(1:NbrOidx,nbrAlleles)==envEnd,:);
nbr_popA=length(popA);
nbr_popB=length(popB);
nbr_popA_to_remove = nbr_popA-envAtot;
nbr_popB_to_remove = nbr_popB-envBtot;
if nbr_popA_to_remove>0
    %create the perf vector
    perf_vector=popA(:,nbrAlleles+2);
    %select indexes to remove
    idx_to_remove=selection(1,nbr_popA_to_remove,perf_vector,0);
    i2r=sort(idx_to_remove);
    for k6=1:nbr_popA_to_remove

```

```

        popA(i2r(k6)-k6+1,:)=[];
%       perf_vector(i2r(k6)-k6+1,:)=[];
    end
end
if nbr_popB_to_remove>0
    %create the perf vector
    perf_vector=popB(:,nbrAlleles+2);
    %select indexes to remove
    idx_to_remove=selection(1,nbr_popB_to_remove,perf_vector,0);
    i2r=sort(idx_to_remove);
    for k6=1:nbr_popB_to_remove
        popB(i2r(k6)-k6+1,:)=[];
%       perf_vector(i2r(k6)-k6+1,:)=[];
    end
end
pop=zeros(pop_size+2*nbr_children,nbrAlleles+2
NbrOidx=size(popA,1)+size(popB,1);
pop(1:NbrOidx,:)=vertcat(popA,popB);
clf(sgafig,'reset');%deletes from the current figure all graphics objects
%record cputime it took
nbrIter_AvgFitness_tConvg_nSchools_reptNbr(j,2,i)= cputime-t;
nbrIter_AvgFitness_tConvg_nSchools_reptNbr(j,1,i)=mean_schools;
%creates population of fishes and group them
%[IDX nc]=gemac();
fprintf('Done with iteration %d.\n',j);
%pause;
uniqueIDs=unique(pop(1:NbrOidx,nbrAlleles+1));

```



```

hmany=size(uniqueIDs,1);
newIDs=1:1:hmany;
for y=1:pop_size
    for z=1:hmany
        if(pop(y,nbrAlleles+1)==uniqueIDs(z,1))
            pop(y,nbrAlleles+1)=newIDs(1,z);
            break;
        end
    end
end
IDX=pop(1:NbrOidx,nbrAlleles+1);%% %IDX=pop(:,nbrAlleles+1);
if max(IDX) ~= hmany %check the packing happened well
    error('Error: length(uniqueIDs) value MUST MATCH hmany');
end
nc=hmany;
end
end
%update value of max_iterations if convergence caused by 'stop == 1'
if stop == 1
    max_iterations = j;%exactIterCount;
    tabu=tabu(1:tabuIdx,1:nbrAlleles);
end
%Format data for SGA results plotting
nf=zeros(1,max_iterations);
statuses=zeros(3,max_iterations);
nmeans=zeros(1,max_iterations);
iter_vect=linspace(1,max_iterations,max_iterations);

```

```

for m1=1:max_iterations
    nf(1,m1)=nbrIter_AvgFitness_tConvg_nSchools_reptNbr(m1,3,i);%sizStaFitMax(m1).data;
    nmeans(1,m1)=nbrIter_AvgFitness_tConvg_nSchools_reptNbr(m1,1,i);
    nmaxes(1,m1)=max(sizStaFitMax{1,m1}.data(4,:));
    for m2=1:length(sizStaFitMax{1,m1}.data(2,:))
        switch sizStaFitMax{1,m1}.data(2,m2)
            case 1
                statuses(1,m1)=statuses(1,m1)+1;%FF
            case 2
                statuses(2,m1)=statuses(2,m1)+1;%SM
            case 3
                statuses(3,m1)=statuses(3,m1)+1;%PA
            %otherwise %school without progeny
            %error('Case should never occur');
        end
    end
end
fprintf('Done with iteration %d.\n',m1);
end
% Plots the SGA Results
sgares=figure('Name','TEST_SGA | Results','Numbertitle','off');
subplot(2,2,1);
plot(iter_vect,nf,'k*-');
title('# Schools per Generation');
subplot(2,2,2);
plot(iter_vect,nmeans,'mv-');
title('School Average Fitness per Generation');
subplot(2,2,3);

```

```

plot(iter_vect,nmaxes(1,1:max_iterations),'cd-');
title(sprintf('Max fitness per Generation with best = % 1.4f',max(nmaxes)));
subplot(2,2,4);
plot(iter_vect,statuses(1,1:max_iterations),'b.-');%FF
hold on
plot(iter_vect,statuses(2,1:max_iterations),'r.-');%SM
plot(iter_vect,statuses(3,1:max_iterations),'g.-');%PA
title('Behavior per Generation. SM-R, PA-G, FF-B');
hold off;
filename=strcat('results',num2str(i));
saveas(sgares,filename,'png');% exports figure to JPEG
%Compute TPI index of all parameters
tabu = tabu(1:min(tabuIdx,tabuLen),:);
[tpi pd] = impactOnPerformance(tabu,nDp,nOp,nEp,limits);%lce_extremes);
disp('TPI values are:');
disp(tpi);
disp('PD values are:');
disp(pd);
end

```

Appendix B

This appendix contains the main algorithm for GSN. All the files that are referenced follow starting with Appendix E. to run the code for all 5 sets of experiments that were carried for the dissertation, some minor changes need to be made to this file and all the supporting files. Just follow the comments left within each file. The name of the following file is “gsn.m.”

```
%reset workspace and memory

clear global variables;

clear all;

clc;

global offset MinKnowledge tolerance pop domainRange themes nbrAlleles NbrOidx
memberships stop ext limits envEnd envStart;% tabu tabuLen tabuDistance;

stop = 0;

stop_count = 0;

MinKnowledge=0.7;

tolerance=1e-11;

nbrAlleles=3; %each chromosome will be encoded using nbrAlleles alleles

%tabuDistance=15*nbrAlleles; %minimum distance for tabu

ext=500;

%Sets number of each parameter

nDp=1;

nOp=1;

nEp=1;

%generate field

[X,Y] = meshgrid(-ext:1:ext);

[Z minZ maxZ] = performance(X,Y);

nbr_repetitions=1;
```

```

max_iterations=1000;
maxSameMaxPeriod=3;
exactIterCount = 0;
past_avg = 0.0;
%tabuLen=round(max_iterations*.20);%20% of # of iterations
%tabu=zeros(tabuLen,nbrAlleles);
lTerm=10;%determines how many generations leaders are elected for
NbrIter_AvgFitness_tConvg_nGroups_reptNbr=zeros(max_iterations,3,nbr_repetitions);
pop_size=120;
percentA=0.55;
envAtot=round(percentA*pop_size);% 55% of population comes from environment A
envBtot=pop_size-envAtot;% 45% of population comes from environment B
nbr_children=round(0.5*pop_size);% 50% of pop_size
nbr_themes=5;
%howFar=0.01;%proportion of search domain to be used for mutation range
%howFar2=0.50;%proportion of search domain to be used for mutation range
center=[0 0];% {0,0} cartesian is [101,101] Matlab
%lce_extremes=zeros(nbrAlleles,2);
envStart=-421;%0;
envEnd=421;%6;
domainStart=-ext;
domainEnd=+ext;
offset=ext+1;
domainRange=domainEnd-domainStart+1;
limits=[domainStart domainEnd; domainStart domainEnd; envStart envEnd];% we are in dim=3
sigma=domainRange/6;

```

```

pop=zeros(pop_size+2*nr_children,nbrAlleles+1);% 1 - |fit|
themes=zeros(nr_themes,nbrAlleles+1);
%Zp=zeros(pop_size+2*nr_children,1);
nmaxes=zeros(1,max_iterations);

for i=1:nr_repetitions

    gsnfig = figure('Name','GSN TEST | In Progress Solutions', 'NumberTitle', 'off', 'Units',
'normalized', 'Position', [.05 .05 .90 .90]);

    array1=normrnd(center(1,1),sigma,[1,pop_size]);
    array2=normrnd(center(1,2),sigma,[1,pop_size]);
    array4=normrnd(center(1,1),sigma,[1,nr_themes]);
    array5=normrnd(center(1,2),sigma,[1,nr_themes]);
    array6=randperm(nr_themes);
    envID=randperm(pop_size);

    %makes sure we start with feasible solutions, and correct the offset
    for h=1:pop_size

        pop(h,1)=array1(h)-domainRange*round(array1(h)/domainRange);
        pop(h,2)=array2(h)-domainRange*round(array2(h)/domainRange);

        %randomly assign population to environments
        if envID(h)<= envAtot

            pop(h,nbrAlleles)=envStart;%environment A
        else

            pop(h,nbrAlleles)=envEnd;%environment B
        end

        pop(h,nbrAlleles+1)=performance(pop(h,1:nbrAlleles));
    end

%    for h2=1:nbrAlleles
%        lce_extremes(h2,1)=min(limits(h2,:));

```

```

%     Ice_extremes(h2,2)=max(limits(h2,:));
% end

% makes sure we start with feasible group themes, and correct the offset
for h2=1:nbr_themes

    themes(h2,1)=array4(h2)-domainRange*round(array4(h2)/domainRange);
    themes(h2,2)=array5(h2)-domainRange*round(array5(h2)/domainRange);
    if array6(h2)<=round(percentA*nbr_themes)
        themes(h2,nbrAlleles)=envStart;%environment A
    else
        themes(h2,nbrAlleles)=envEnd;%environment B
    end

    themes(h2,nbrAlleles+1)=performance(themes(h2,1:nbrAlleles));
end

themes0 = themes;

j=0;
NbrOidx=pop_size;
%Zp(1:NbrOidx,1)=performance(pop(1:NbrOidx,1),pop(1:NbrOidx,2),pop(1:NbrOidx,3));
while and(j < max_iterations, stop == 0)

    %creates population of individuals and group them <- put out of loop so it
    %can be called right at the end of each iteration
    leadIndx=createSocialGroups();
    nc = length(leadIndx);

        membership=zeros(pop_size+2*nbr_children,nc);
        membership(1:NbrOidx,:)=memberships;

    term_counter=0;
    NbrIter_AvgFitness_tConvg_nGroups_reptNbr(j+1,3,i)=nc;

```

```

while term_counter < lTerm
    t = cputime;
    j=j+1;
    sum_perf=zeros(1,nc);
    mean_perf=zeros(1,nc);
    max_perf=zeros(1,nc);
    perf=zeros(NbrOidx,nc);
    for k2=1:nc
        perf(:,k2)=membership(1:NbrOidx,k2).*pop(1:NbrOidx,nbrAlleles+1);
        sum_perf(1,k2)=sum(perf(:,k2));
        max_perf(1,k2)=max(perf(:,k2));
        mean_perf(1,k2)=mean(perf(:,k2));
    end
    term_counter=term_counter+1;
    nmaxes(1,j)=max(max(perf));%max(pop(1:NbrOidx,nbrAlleles+1));
    new_members=pop_size+nbr_children-NbrOidx;
    ratios=sum_perf./sum(sum_perf);
    members_per_group=round(new_members*ratios);
%     %in case some sums were negative while others were positive
%     for ps=1:length(members_per_group)
%         if members_per_group(ps)<0
%             members_per_group(ps)=0;
%         end
%     end
    mean_groups=mean(mean_perf);
    %effective number added
    %to_add=0;

```



```

%for k3=1:nc
%  to_add=to_add+2*round(members_per_group(1,k3)/2);
%end
%for_children=zeros(to_add,3);
%Check whether a stopping criteria is reached
if (abs(past_avg-mean_groups)<tolerance)
    stop_count = stop_count +1;
    if stop_count == maxSameMaxPeriod
        exactIterCount = j;
        stop = 1;
    end
else
    past_avg = mean_groups;
    stop_count = 0;
end
%plotting part -- Not available from 4+ dimensions
%  contour3(X,Y,Z,70)
%  %%%%%%%%%%%%%%%init_pos=campos;
%  xlabel('X-plan','FontSize',8)
%  ylabel('Y-plan','FontSize',8)
%  zlabel('Z-performance','FontSize',8)
%  %colormap default
%  colormap('white')% 'bone' 'white' 'winter'
%  colorbar
%
%  hold on
%  plot3(pop(1:NbrOidx,1),pop(1:NbrOidx,2),pop(1:NbrOidx,nbrAlleles+1),'ob');

```

```

% campos([0 0 maxZ]);
% title(sprintf('Total Fitness Average = %1.4f, Iteration = %d',mean_groups,j));
% drawnow
% hold off;
% % Uncomment the next 2 lines if you don't mind waiting
% filename=strcat('frame',num2str(j));
% saveas(gsnfig,filename,'jpg');% exports figure to JPEG
%execute GA operators
for k4=1:nc
    wtpc=selection(0,round(members_per_group(1,k4)/2),perf(:,k4),0);
    for k42=1:2:round(members_per_group(1,k4)/2)
        child=crossOver2(pop(wtpc(k42,:),),pop(wtpc(k42+1,:),));
        NbrOidx=NbrOidx+1;
        pop(NbrOidx,:)=child;
        %Crossover requires 2 parents (think fuzzy AND to pass on membership)
        %child_membership=min([membership(wtpc(k42,:),);
membership(wtpc(k42+1,:),)]);
        %membership(NbrOidx,:)= child_membership;
        membership(NbrOidx,:)= min([membership(wtpc(k42,:),);
membership(wtpc(k42+1,:),)]);
    end
    wtpm=selection(0,round(members_per_group(1,k4)/2),perf(:,k4),1);
    for k42=1:round(members_per_group(1,k4)/2)

child=mutation2(pop(wtpm(k42,:),),mean_perf(k4)/max(tolerance,abs(nmaxes(1,j))));
        NbrOidx=NbrOidx+1;
        pop(NbrOidx,:)=child;
        %Mutation requires 1 parent (think OR, fuzzy OR)

```

```

%child_membership=max([membership(wtpm(k42),:);rand(1,nc)]);
%membership(NbrOidx,:)= child_membership;
membership(NbrOidx,:)= max([membership(wtpm(k42),:);rand(1,nc)]);
end
end
%keeps population size constant
%First, remove duplicates if any
[pop idxs]=unique(pop(1:NbrOidx,:), 'rows');
%remove corresponding rows
NbrOidx=size(pop,1);
membrs=zeros(NbrOidx,size(membership,2));
for cr=1:NbrOidx
    membrs(cr,:)=membership(idxs(cr),:);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%split population by environment -- to maintain ratio
popA = pop(pop(1:NbrOidx,nbrAlleles)==envStart,:);
popB = pop(pop(1:NbrOidx,nbrAlleles)==envEnd,:);
lenA=length(popA);
lenB=length(popB);
nbr_popA_to_remove =lenA-envAtot;
nbr_popB_to_remove = lenB-envBtot;
    mbr_vectorA=zeros(lenA,nc);
mbr_vectorB=zeros(lenB,nc);
    idx=ones(1,2);
for h = 1:NbrOidx
    if pop(h,nbrAlleles)==envStart;%environment A

```

```

    mbr_vectorA(idx(1,:)=membrs(h,:);
                idx(1)=idx(1)+1;
else %environment B
    mbr_vectorB(idx(2,:)=membrs(h,:);
                idx(2)=idx(2)+1;
end
end

if nbr_popA_to_remove>0
    %create the perf vector
    perf_vectorA=popA(:,nbrAlleles+1);
    %select indexes to remove
    idx_to_remove=selection(1,nbr_popA_to_remove,perf_vectorA,0);
    i2r=sort(idx_to_remove);
    for k6=1:nbr_popA_to_remove
        popA(i2r(k6)-k6+1,:)=[];
        perf_vectorA(i2r(k6)-k6+1,:)=[];
        mbr_vectorA(i2r(k6)-k6+1,:)=[];
    end
end

if nbr_popB_to_remove>0
    %create the perf vector
    perf_vectorB=popB(:,nbrAlleles+1);
    %select indexes to remove
    idx_to_remove=selection(1,nbr_popB_to_remove,perf_vectorB,0);
    i2r=sort(idx_to_remove);
    for k6=1:nbr_popB_to_remove

```

```

        popB(i2r(k6)-k6+1,:)=[];
        perf_vectorB(i2r(k6)-k6+1,:)=[];
        mbr_vectorB(i2r(k6)-k6+1,:)=[];
    end
end

pop_size=size(popA,1)+size(popB,1);
pop=zeros(pop_size+2*nr_children,nbrAlleles+1);% 1 - |fit|
membership=zeros(pop_size+2*nr_children,nc);
NbrOidx=pop_size;
pop(1:NbrOidx,:)=vertcat(popA,popB);
membership(1:NbrOidx,:)=vertcat(mbr_vectorA,mbr_vectorB);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf(gsnfig,'reset');% deletes from the current figure all graphics objects
%record cputime it took
NbrIter_AvgFitness_tConvg_nGroups_reptNbr(j,2,i)= cputime-t;
NbrIter_AvgFitness_tConvg_nGroups_reptNbr(j,1,i)=mean_groups;
fprintf('Done with iteration %d.\n',j);
%prep data for next iteration
%[Zp minZp maxZp] = performance(pop(:,1),pop(:,2),pop(:,3));
end
disp('time to upgrade community themes and re-elect leaders ;-)\n');
%%Upgrade community themes for new leaders --> dynamic themes
for m=1:nbr_themes
    for t_col=1:nbrAlleles-1
new_val=themes(m,t_col)+sum(perf(1:NbrOidx,m).*pop(1:NbrOidx,t_col))/sum(perf(1:NbrOidx,m));

```

```

        themes(m,t_col)=new_val-domainRange*round(new_val/domainRange);
    end
    themes(m,nbrAlleles+1)=performance(themes(m,1:nbrAlleles));
end
end

%update value of max_iterations if convergence caused by 'stop == 1'
if stop == 1
    max_iterations = exactIterCount;
%    if tabuLen > exactIterCount
%        tabu=tabu(1:exactIterCount,nbrAlleles);
%    end
end

% nmeans=zeros(1,max_iterations);
nmeans=NbrIter_AvgFitness_tConv_nGroups_reptNbr(1:j,1,i);
% Plots the GSN Results
gsnres=figure('Name','TEST_GSN | Results','Numbertitle','off');
subplot(2,1,1);
iter_vect=linspace(1,j,j);
plot(iter_vect,nmeans,'mv-');
title('Group Average Fitness per Generation');
subplot(2,1,2);
plot(iter_vect,nmaxes(1,1:j),'cd-');
title(sprintf('Max fitness per Generation with best = %1.4f',max(nmaxes)));
filename=strcat('results',num2str(i));
saveas(gsnres,filename,'png');% exports figure to JPEG

```

```
%Compute TPI index of all parameters
% [tpi pd] = impactOnPerformance(tabu,nDp,nOp,nEp,limits);%lce_extremes);
% disp('TPI values are:');
% disp(tpi);
% disp('PD values are:');
% disp(pd);
end
```

Appendix C

This appendix contains the main algorithm for IGA. All the files that are referenced follow starting with Appendix E. to run the code for all 5 sets of experiments that were carried for the dissertation, some minor changes need to be made to this file and all the supporting files. Just follow the comments left within each file. The name of the following file is “iga.m.”

```
%reset workspace and memory

clear global variables;

clear all;

clc;

global iSize envEnd envStart nbrAlleles migrationInterval tolerance islands popSize
domainRange limits nbrIslands migrationSize pop;

%initialize variables

nbrAlleles=3;%gen1|gen2|evmt|perf

nbrIslands=5;

popSize=120;

migrationInterval=30;

migrationSize=5;

tolerance=1e-11;

iSize=round(popSize/nbrIslands);%popSize should be a multiple of nbrIslands

ext=500;

%generate field

%[X,Y] = meshgrid(-ext:1:ext);

envAtot=round(0.65*popSize);%65% of tot pop

envBtot=popSize-envAtot;

center=[0 0];

envStart=0;
```



```

envEnd=6;
domainStart=-ext;
domainEnd=+ext;
offset=ext+1;
domainRange=domainEnd-domainStart+1;
limits=[domainStart domainEnd; domainStart domainEnd; envStart envEnd];% we are in dim=3
sigma=domainRange/6;
nbr_replications=1;
max_iterations=1020;
xVal=1:1:max_iterations;
for i=1:nbr_replications
    %For summary data plotting
    %to record all bests from each island
    bests=zeros(max_iterations,nbrAlleles+1,nbrIslands);
    averages=zeros(max_iterations,1,nbrIslands);
    %initializes population
    pop=zeros(popSize,nbrAlleles+1);% % gen1|gen2|evmt|perf
    array1=normrnd(center(1,1),sigma,[1,popSize]);
    array2=normrnd(center(1,2),sigma,[1,popSize]);
    array3=zeros(1,popSize);
    envID=randperm(popSize);
    %makes sure we start with feasible solutions, and correct the offset
    for h=1:popSize
        pop(h,1)=array1(h)-domainRange*round(array1(h)/domainRange);
        pop(h,2)=array2(h)-domainRange*round(array2(h)/domainRange);
        %randomly assign population to environments
        if envID(h)<= envAtot

```

```

        pop(h,nbrAlleles)=envStart;%environment A
    else
        pop(h,nbrAlleles)=envEnd;%environment B
    end
    %Add performance values
    pop(h,nbrAlleles+1)=performance(pop(h,1:nbrAlleles));
end

%initializes islands structure and populates it
islands=zeros(iSize,nbrAlleles+1,nbrIslands);
create_islands();
j=0;
nbrOffspring=round(0.50*popSize)/nbrIslands;% 50% of pop size
while j<round(max_iterations/migrationInterval);
    allBests=zeros(migrationInterval,nbrAlleles+1,nbrIslands);
    allAverages=zeros(migrationInterval,1,nbrIslands);

    j=j+1;
    %Go through all islands, run them all in parallel
    parfor k = 1:nbrIslands
        [allAverages(:,1,k) allBests(:,:,k)]=ga(k,nbrOffspring);
    end
    %Store bests
    for k=1:nbrIslands
        bests((j-1)*migrationInterval+1:j*migrationInterval,:,k)=allBests(:,:,k);
        averages((j-1)*migrationInterval+1:j*migrationInterval,1,k)=allAverages(:,1,k);
    end
end

```

```

    %Proceeds with migration to add diversity within each subpopulation
    migrate();
end

%plot all bests for each subpopulation
% Plots the IGA Results
igares=figure('Name','TEST_IGA | Results','Numbertitle','off');
subplot(2,3,1);
plot(xVal,bests(:,nbrAlleles+1,1),'k*-');
title('Bests/Gen. for Isl.#1');
subplot(2,3,2);
plot(xVal,bests(:,nbrAlleles+1,2),'b.-');
title('Bests/Gen. for Isl.#2');
subplot(2,3,3);
plot(xVal,bests(:,nbrAlleles+1,3),'r.-');
title('Bests/Gen. for Isl.#3');
subplot(2,3,4);
plot(xVal,bests(:,nbrAlleles+1,4),'g.-');
title('Bests/Gen. for Isl.#4');
subplot(2,3,5);
plot(xVal,bests(:,nbrAlleles+1,5),'m*-');
title('Bests/Gen. for Isl.#5');

filename=strcat('allBests',num2str(i));
saveas(igares,filename,'png');% exports figure to PNG

```

```
igares2=figure('Name','TEST_IGA | Results','Numbertitle','off');
subplot(2,3,1);
plot(xVal,averages(:,1,1),'k*-');
title('Avg/Gen. for Isl.#1');
subplot(2,3,2);
plot(xVal,averages(:,1,2),'b.-');
title('Avg/Gen. for Isl.#2');
subplot(2,3,3);
plot(xVal,averages(:,1,3),'r.-');
title('Avg/Gen. for Isl.#3');
subplot(2,3,4);
plot(xVal,averages(:,1,4),'g.-');
title('Avg/Gen. for Isl.#4');
subplot(2,3,5);
plot(xVal,averages(:,1,5),'m*-');
title('Avg/Gen. for Isl.#5');

filename=strcat('allAverages',num2str(i));
saveas(igares2,filename,'png');% exports figure to PNG
% waits for a key press (any key) before continuing
%pause
endj
```

Appendix D

This appendix contains the main algorithm for PGA. All the files that are referenced follow starting with Appendix E. to run the code for all 5 sets of experiments that were carried for the dissertation, some minor changes need to be made to this file and all the supporting files. Just follow the comments left within each file. The name of the following file is “pga.m.”

```
%reset workspace and memory
clear global variables;
clear all;
clc;

global iSize envEnd envStart nbrAlleles migrationInterval tolerance islands popSize
domainRange limits nbrIslands migrationSize pop;

%initialize variables
nbrAlleles=5;%gen1|gen2|evmt|perf
nbrIslands=4;
popSize=120;
migrationInterval=30;
migrationSize=2;
tolerance=1e-11;
iSize=round(popSize/nbrIslands);%popSize should be a multiple of nbrIslands
ext=500;
%generate field
%[X,Y] = meshgrid(-ext:1:ext);
envAtot=round(0.55*popSize);%65% of tot pop
envBtot=popSize-envAtot;
center=[0 0];
envStart=0;
```

```

envEnd=6;
domainStart=-ext;
domainEnd=+ext;
offset=ext+1;
domainRange=domainEnd-domainStart+1;
limits=[domainStart domainEnd; domainStart domainEnd; envStart envEnd];% we are in dim=3
sigma=domainRange/6;

nbr_replications=1;
max_iterations=1020;
xVal=1:1:max_iterations;

for i=1:nbr_replications
    %For summary data plotting
    %to record all bests from each island
    bests=zeros(max_iterations,nbrAlleles+1,nbrIslands);

    averages=zeros(max_iterations,1,nbrIslands);

    %initializes population
    pop=zeros(popSize,nbrAlleles+1);% gen1|gen2|gen3|gen4|evmt|perf
    array1=normrnd(center(1,1),sigma,[1,popSize]);
    array2=normrnd(center(1,2),sigma,[1,popSize]);
    array3=zeros(1,popSize);
    envID=randperm(popSize);
    %makes sure we start with feasible solutions, and correct the offset
    for h=1:popSize

```

```

pop(h,1)=array1(h)-domainRange*round(array1(h)/domainRange);
pop(h,2)=array2(h)-domainRange*round(array2(h)/domainRange);
%randomly assign population to environments
if envID(h)<= envAtot
    pop(h,nbrAlleles)=envStart;%environment A
else
    pop(h,nbrAlleles)=envEnd;%environment B
end
%Add performance values
pop(h,nbrAlleles+1)=performance(pop(h,1:nbrAlleles));
end

%initializes islands structure and populates it
islands=zeros(iSize,nbrAlleles+1,nbrIslands);
create_islands();
j=0;
nbrOffspring=round(0.50*popSize)/nbrIslands;% 50% of pop size
while j<round(max_iterations/migrationInterval);
    allBests=zeros(migrationInterval,nbrAlleles+1,nbrIslands);
    allAverages=zeros(migrationInterval,1,nbrIslands);

    j=j+1;
    %Go through all islands, run them all in parallel
    parfor k = 1:nbrIslands
        [allAverages(:,1,k) allBests(:,:,k)]=ga(k,nbrOffspring);
    end
    %Store bests

```

```

for k=1:nbrIslands
    bests((j-1)*migrationInterval+1:j*migrationInterval,:,k)=allBests(:,k);
    averages((j-1)*migrationInterval+1:j*migrationInterval,1,k)=allAverages(:,1,k);
end

%No migration this time
end

%plot all bests for each subpopulation
% Plots the PGA Results
pgares=figure('Name','TEST_PGA | Results','Numbertitle','off');
subplot(2,3,1);
plot(xVal,bests(:,nbrAlleles+1,1),'k*-');
title('Bests/Gen. for Subpop#1');
subplot(2,3,2);
plot(xVal,bests(:,nbrAlleles+1,2),'b.-');
title('Bests/Gen. for Subpop#2');
subplot(2,3,3);
plot(xVal,bests(:,nbrAlleles+1,3),'r.-');
title('Bests/Gen. for Subpop#3');
subplot(2,3,4);
plot(xVal,bests(:,nbrAlleles+1,4),'g.-');
title('Bests/Gen. for Subpop#4');
subplot(2,3,5);
plot(xVal,bests(:,nbrAlleles+1,5),'m*-');
title('Bests/Gen. for Subpop#5');

filename=strcat('results',num2str(i));

```



```
saveas(pgares,filename,'png');% exports figure to PNG

pgares2=figure('Name','TEST_PGA | Results','Numbertitle','off');
subplot(2,3,1);
plot(xVal,averages(:,1,1),'k*-');
title('Avg/Gen. for Subpop#1');
subplot(2,3,2);
plot(xVal,averages(:,1,2),'b.-');
title('Avg/Gen. for Subpop#2');
subplot(2,3,3);
plot(xVal,averages(:,1,3),'r.-');
title('Avg/Gen. for Subpop#3');
subplot(2,3,4);
plot(xVal,averages(:,1,4),'g.-');
title('Avg/Gen. for Subpop#4');
subplot(2,3,5);
plot(xVal,averages(:,1,5),'m*-');
title('Avg/Gen. for Subpop#5');

filename=strcat('allAverages',num2str(i));
saveas(pgares2,filename,'png');% exports figure to PNG

% waits for a key press (any key) before continuing

%pause

end
```

Appendix E

This appendix contains the support files for SGA, GSN, IGA, and PGA implementations.

In all cases, the name of the file is given prior the listing of its contents.

```
*****gemac.m*****
function varargout = gemac()
% Geometrically Expanded Membership for Automated Clustering--> GEMAC
global pop NbrOidx nbrAlleles;
f=2.7;% 1.97;% 2.7;% 2.7183;% power of proximity
distances=zeros(NbrOidx,NbrOidx);
for i1=1:NbrOidx-1
    for j1=i1+1:NbrOidx
        % working with integers is always faster than with reals
        distances(i1,j1)=round(similarity('cityblock',pop(i1,1:nbrAlleles),pop(j1,1:nbrAlleles)));
        distances(j1,i1)=distances(i1,j1);
    end
end
distances2=distances;
mode_node_proxy=zeros(1,NbrOidx);
ppop=0.12;% 12% of the population will be sampled
s2=round(ppop*NbrOidx);
rsamples=randint(NbrOidx,s2,[1 NbrOidx]);
for ik=1:NbrOidx
    for ij=1:s2
        mode_node_proxy(ik)=mode_node_proxy(ik)+distances(ik,rsamples(ik,ij));
    end
    mode_node_proxy(ik)=mode_node_proxy(ik)/s2;
end
```

```

end
plp=NbrOidx;
k=10000;% must be a value that does not occur within data
e2nc=[(1/f^1) (1/f^2) (1/f^3) (1/f^4) (1/f^5)];
%max_level=5;
k_used=0;
distances(1,1)=k;
distances(2:plp,1)=Inf;% no fuzzy membership allowed.
%level0
%everything was working fine till I substituted i2 with level0(1,i2) in
%following for-loop. I also changed level0=linspace(1,NbrOidx,NbrOidx) to
%level0=randperm(NbrOidx)
level0=linspace(1,NbrOidx,NbrOidx);%randperm(NbrOidx);%
%level1
for i2=1:plp%-1
    if level0(i2)==0; continue; end%no double usage at level0
    %level1conn=[];
    level1conn=zeros(1,NbrOidx);%pre-allocate for speed
    idx1=1;
    for j2=1:plp
        if distances(i2,j2)<=mode_node_proxy(i2)*e2nc(1)
            distances(i2,j2)=k;
            %level1conn=[level1conn j2];
            level1conn(1,idx1)=j2;
            idx1=idx1+1;
            level0(1,j2)=0;
            k_used=1;

```

```

    distances(i2+1:plp,j2)=Inf;% no fuzzy membership allowed.
    distances(1:i2-1,j2)=Inf;% no fuzzy membership allowed.
end
end
%level2
for j3=1:length(level1conn)
    %level2conn=[];
    level2conn=zeros(1,NbrOidx);% pre-allocate for speed
    idx2=1;
    if level1conn(1,j3)==0; break; end% no double usage for level1
    for j4=1:plp
        if distances(level1conn(1,j3),j4)<=mode_node_proxy(i2)*e2nc(2)
            %level2conn=[level2conn j4];
            level2conn(1,idx2)=j4;
            idx2=idx2+1;
            level0(1,j4)=0;
            distances(level1conn(1,j3),j4)=k;
            distances(level1conn(1,j3)+1:plp,j4)=Inf;% no fuzzy membership allowed.
            distances(1:level1conn(1,j3)-1,j4)=Inf;% no fuzzy membership allowed.
        end
    end
end
end
%level3
for j5=1:length(level2conn)
    %level3 --we stop here for now
    %level3conn=[];
    level3conn=zeros(1,NbrOidx);% pre-allocate for speed
    idx3=1;

```

```

if level2conn(1,j5)==0; break; end%no double usage for level2
for j6=1:plp
    if distances(level2conn(1,j5),j6)<=mode_node_proxy(i2)*e2nc(3)
        %level3conn=[level3conn j6];
        level3conn(1,idx2)=j6;
        idx3=idx3+1;
        level0(1,j6)=0;
        distances(level2conn(1,j5),j6)=k;
        distances(level2conn(1,j5)+1:NbrOidx,j6)=Inf;%no fuzzy membership allowed.
        distances(1:level2conn(1,j5)-1,j6)=Inf;%no fuzzy membership allowed.
    end
end
%level4 --will start here
end
end
if k_used
    k=k+1;
    k_used=0;
end
end
for i2=1:plp
    if not(isinf(max(distances(:,i2))))
        distances(1,i2)=k;
        k=k+1;
    end
end
end

```

```

%finish the partitioning
CIX=min(distances)-9999;
NC=max(CIX);%returns the number of clusters
% Return Outputs
if nargout
    varargout{1} = CIX;
    varargout{2} = NC;
    varargout{3} = distances2;
    varargout{4} = mode_node_proxy;
end
end
*****"ga.m"*****

%GA Simple GA Algorithm with performance that is environment dependent
function varargout = ga(islIdx, nbrOffspring)
global nbrAlleles migrationInterval islands iSize domainRange;
%sets the population--(gen1|gen2|evmt|perf)
%iSize=round(popSize/nbrIslands);% popSize should be a multiple of iSize
lpop=zeros(iSize+2*nbrOffspring,nbrAlleles+1);
lpop(1:iSize,:)=islands(:, :, islIdx);
best=zeros(migrationInterval,nbrAlleles+1);
average=zeros(migrationInterval,1);
NbrOidx=iSize;
%determine ratio for each environment
envAtot=sum(lpop(:,nbrAlleles)==1);
envBtot=iSize-envAtot;
% Run the GA
for iter = 1:1:migrationInterval

```

```

%execute GA operators - 50% XER, 50% uTION
wtpc=selection(0,round(nbrOffspring/2),lpop(1:iSize,nbrAlleles+1),0);
for k42=1:2:round(nbrOffspring/2)
    child=crossOver2(lpop(wtpc(k42),:),lpop(wtpc(k42+1),:));
    NbrOidx=NbrOidx+1;
    lpop(NbrOidx,:)=child;
end

%Use the same population as the XER operator
wtpm=selection(0,round(nbrOffspring/2),lpop(1:iSize,nbrAlleles+1),1);
for k42=1:round(nbrOffspring/2)
    child=mutation2(lpop(wtpm(k42),:),max(rand(1,3))*domainRange);
    NbrOidx=NbrOidx+1;
    lpop(NbrOidx,:)=child;
end

%Finds the best and the average and record them both
[v iBest]=max(lpop(1:NbrOidx,nbrAlleles+1));
best(iter,:)=lpop(iBest,:);
average(iter,1)=mean(lpop(:,nbrAlleles+1));

%Reduction/culling process - preserve environment ratios
%First, remove duplicates if any
lpop=unique(lpop,'rows');
pSize=length(lpop);
lpopA = lpop(lpop(1:pSize,nbrAlleles)==1,:);
lpopB = lpop(lpop(1:pSize,nbrAlleles)==2,:);
nbr_lpopA=size(lpopA,1);
nbr_lpopB=size(lpopB,1);

```

```

nbr_lpopA_to_remove = nbr_lpopA-envAtot;
nbr_lpopB_to_remove = nbr_lpopB-envBtot;
idxA_to_remove=selection(1,nbr_lpopA_to_remove,lpopA(1:nbr_lpopA,nbrAlleles+1),0);
i2r=sort(idxA_to_remove);
for k6=1:nbr_lpopA_to_remove
    lpopA(i2r(k6)-k6+1,:)=[];
end
idxB_to_remove=selection(1,nbr_lpopB_to_remove,lpopB(1:nbr_lpopB,nbrAlleles+1),0);
i2r=sort(idxB_to_remove);
for k6=1:nbr_lpopB_to_remove
    lpopB(i2r(k6)-k6+1,:)=[];
end

%Resets population
lpop=zeros(iSize+2*nbrOffspring,nbrAlleles+1);
lpop(1:size(lpopA,1)+size(lpopB,1),:)=vertcat(lpopA,lpopB);
NbrOidx=size(lpopA,1)+size(lpopB,1);%ideally, should be: iSize
end

% Return Outputs
if nargout
    varargout{1} = average;
    varargout{2} = best;
end

*****similarity.m*****

% Similarity measure of numerical data%
function measure = similarity(type,X,Y)

```



```
lx=length(X);
ly=length(Y);
if lx ~= ly, error('genes"length must match'); end
Z=[X;Y];
switch lower(type)
    case {'euclidean','default','dist'}
        %disp('Computing Euclidean distance')
        measure=pdist(Z,'euclidean');
    case 'seuclidean'
        %disp('Computing the Standardized Euclidean distance')
        measure=pdist(Z,'seuclidean');
    case 'mahalanobis'
        %disp('Computing the Standardized Euclidean distance')
        measure=pdist(Z,'mahalanobis');
    case {'cityblock','manhattan','taxicab'}
        %disp('computing the manhattan distance')
        measure=pdist(Z,'cityblock');
        case 'minkowski'
            %disp('computing the minkowski distance')
            measure=pdist(Z,'minkowski');
    case 'cosine'
        %disp('Computing cosine distance')
        measure=pdist(Z,'cosine');
    case 'correlation'
        %disp('Computing the correlation distance')
        measure=pdist(Z,'correlation');
    case 'spearman'
```

```

    %disp('Computing the spearman distance')
    measure=pdist(Z,'spearman');
case 'hamming'
    %disp('computing the hamming distance')
    measure=pdist(Z,'hamming');
        case 'jaccard'
            %disp('Computing the jaccard distance')
            measure=pdist(Z,'jaccard');
case {'chebychev', 'chessboard', 'sup norm'}
    %disp('computing the chebychev distance')
    measure=pdist(Z,'chebychev');
case 'canberra'
    %disp('computing the canberra distance')
    measure=sum(abs(X-Y)./abs(X+Y));
        case {'bray-curtis', 'sørensen', 'braycurtisdistance'}
            %disp('Computing the Bray-Curtis distance')
            measure=(sum(abs(X-Y))/sum(abs(X+Y)));
case {'matching'}
    %disp('Computing the matching distance')
    measure=sum(X==Y);
otherwise
    error('Similarity measure requested is Unknown. Nothing is done')
end
end

*****similarity.m*****

```

```

function [TPI PD] = impactOnPerformance( bests, nDp, nOp, nEp, lce_min_max)
%Outputs:
%- Trait Performance Indicator (TPI) -- Class-wise (Dsg, Opr, Evm)
%   vector whose values indicates how significant (0 1.0) the given values
%   of a parameter are to the performance
%- Parameter Delta (PD) -- range of parameters for the given performance
%Inputs:
%- nDp/nOp/nEp -- number of Design/Operational/Environmental parameters
%- bests -- best solutions returned by the metaheuristic
%- lce_min_max -- min and max of all considered LCE parameters
%Assumptions:
%- lce_min_max has each allele [min max] values defined row-wise
%- the order (row-wise) of parameters within lce_min_max is Dp|Op|Ep
%- the order (column-wise) of parameters within bests is Dp|Op|Ep
[nBests nAlleles]=size(bests);

if (nDp+nOp+nEp ~= nAlleles)
    error('# of alleles does not equal the sum (nOp+nDp+nEp)');
end

% lce_sizes=[0,nDp,nOp+nDp,nOp+nDp+nEp];

%Use Tanimoto-like distance to compute metric
%We only have 3 classes of LCE parameters
%But each class can be encoded on multiple dimensions

num=zeros(1,nAlleles);%

```

```

dem=zeros(1,nAlleles);%

% Works with values passed within bests and lce_mins_maxs
% for j=1:nAlleles
%   for i=lce_sizes(j)+1:lce_sizes(j+1)
%       num(j)=num(j)+max(bests(:,i))-min(bests(:,i));
%       dem(j)=dem(j)+lce_min_max(i,2)-lce_min_max(i,1);
%   end
% end

for j=1:nAlleles
    num(j)=max(bests(:,j))-min(bests(:,j));
    dem(j)=lce_min_max(j,2)-lce_min_max(j,1);
end

% Outputs MUST always be a 1xnAlleles points
PD=num;
TPI=num./dem;
end

*****'isClear.m'*****

function varargout=isClear(schoolCM)
% Check whether a value belongs to a tabu list
% the tabu list implements a recency list
global tabu tabuDistance tabuIdx tabuLen;
for i=1:min(tabuLen,tabuIdx+1)

```

```

if similarity('cityblock',schoolCM,tabu(i,:))<=tabuDistance
    % Return Outputs
    if nargout
        varargout{1} = -5;
        varargout{2} = i;
    end
    return;
end
end

%new CM does not belong to the list
if nargout
    varargout{1} = 5;
    varargout{2} = 0;
end
end

*****"mutation.m"*****

function child=mutation(parent, range)
%Mutates a parent to create a new solution
%to be used to select parents for mutation process
global nbrAlleles limits nc domainRange envStart envEnd;
%limits contains the limits accross each dimension
%along dim i, limits(i,1)-->min, limits(i,2)-->max
child=zeros(1,size(parent,2));
hm2m=randint(1,1,[1 (nbrAlleles-1)*2]);%Sets the number of alleles or allele to mutate
if hm2m <= nbrAlleles-1

```

```

for i=1:nbrAlleles-1
    if i == hm2m
        r=rand;

        child(1,i)=parent(1,i)+(range*r)*limits(i,2)-
domainRange*round((parent(1,i)+(range*r)*limits(i,2))/domainRange);

        %child(1,i)=parent(1,i)+r*limits(i,2)-
domainRange*round((parent(1,i)+r*limits(i,2))/domainRange);

    else

        child(1,i)=parent(1,i);

    end

end

r=rand;

child(1,hm2m)=parent(1,hm2m)+(range*r)*limits(hm2m,2)-
domainRange*round((parent(1,hm2m)+(range*r)*limits(hm2m,2))/domainRange);

%child(1,i)=parent(1,i)+r*limits(i,2)-domainRange*round((parent(1,i)+r*
%limits(i,2))/domainRange);

else

for i=1:nbrAlleles-1

    r=rand;

    child(1,i)=parent(1,i)+(range*r)*limits(i,2)-
domainRange*round((parent(1,i)+(range*r)*limits(i,2))/domainRange);

    %child(1,i)=parent(1,i)+r*limits(i,2)-
domainRange*round((parent(1,i)+r*limits(i,2))/domainRange);

    end

end

%environment and school ID might change

if rand>0.5

    child(1,nbrAlleles-1:nbrAlleles+1)=parent(1,nbrAlleles-1:nbrAlleles+1);

else

```

```

env=randint(1,1,[limits(nbrAlleles,1) limits(nbrAlleles,2)]);
if env<0.5*(envEnd+envStart)
    child(1,nbrAlleles-1)=envStart;
    child(1,nbrAlleles)=envStart;
else
    child(1,nbrAlleles-1)=envEnd;
    child(1,nbrAlleles)=envEnd;
end
child(1,nbrAlleles+1)=randint(1,1,[1 nc]);
end

%perf=performance(child(1:nbrAlleles));
% if perf > 1.0
%   child(1:nbrAlleles)
%   error('Performance value should never exceed 1.0');
% else
    child(1,nbrAlleles+2)=performance(child(1,1:nbrAlleles));%perf;
% end
end

*****"mutation2.m"*****

function child=mutation2(parent, range)
%Mutates a parent to create a new solution
%to be used to select parents for mutation process
global nbrAlleles limits domainRange envStart envEnd;
%limits contains the limits accross each dimension
%along dim i, limits(i,1)-->min, limits(i,2)-->max

```

```

child=zeros(1,size(parent,2));
for i=1:nbrAlleles-1
    r=rand;
    child(1,i)=parent(1,i)+(range+r)*limits(i,2)-
domainRange*round((parent(1,i)+(range+r)*limits(i,2))/domainRange);
end

%environment might change
if rand>0.5
    child(1,nbrAlleles:nbrAlleles+1)=parent(1,nbrAlleles:nbrAlleles+1);
else
    child(1,nbrAlleles)=randint(1,1,[limits(nbrAlleles,1) limits(nbrAlleles,2)]);
    if child(1,nbrAlleles)<0.5*(envEnd+envStart)
        child(1,nbrAlleles)=envStart;
    else
        child(1,nbrAlleles)=envEnd;
    end
end
end
child(1,nbrAlleles+1)=performance(child(1,1:nbrAlleles));
end

*****performance.m*****

function varargout = performance( varargin )
%PERFORMANCE Summary of this function goes here
% Detailed explanation goes here
global envStart tolerance ext;%envEnd

```



```

maxV=ext;
% persistent perf;
% global offset;
if nargin == 1; % assuming solution was passed in
    soln=varargin{1};%+offset;
%   X=[soln(1), soln(2)];
%   Y=[soln(3), soln(4)];
%   E=[soln(5), soln(6)];
%   X=[soln(1), soln(2)];
%   Y=[soln(3), soln(4)];
%   E=soln(5);
%   X=[soln(1)+rand, soln(2)+rand];
%   Y=[soln(3)+rand, soln(4)+rand];
%   E=[soln(5)+rand, soln(6)+rand];
    X=soln(1);
    Y=soln(2);
    E=soln(3);
elseif nargin==3% assuming X,Y,E were passed in
    X=varargin{1};
    Y=varargin{2};
    E=varargin{3};
else % assuming (nargin==2) X and Y were passed in (Very Special Case)
    X=varargin{1};
    Y=varargin{2};
    Z=-X.*sin(sqrt(abs(X)))-Y.*sin(sqrt(abs(Y)));
    %maxZ=max(max(Z));minZ=min(min(Z));
    if nargout; % assumes nargout value of 3

```

```

    varargout{1} = Z;
    varargout{2} = min(min(Z));
    varargout{3} = max(max(Z));
    return;
end
end
sizeZ=size(X,1);
Z=zeros(sizeZ,1);
    %performance is Dp and Op dependent
for i=1:sizeZ
    %Linear
%    if X(i) == 50
%        Z(i)=1;
%    else
%        Z(i)=tolerance;
%    end
%    if E(i)==envStart; %environment A
%        Z(i)=abs(1-abs((X(i)/maxV)-0.7));
%    else %environment B ==> E(i)==2
%        Z(i)=abs(1-abs((X(i)/maxV)-0.3));
%    end
%    if E(i)==envStart; %environment A
%        Z(i)=1-abs(abs(X(i)/maxV)-0.7)*abs(abs(Y(i)/maxV)-0.3);
%    else %environment B ==> E(i)==2
%        Z(i)=1-abs(abs(X(i)/maxV)-0.3)*abs(abs(Y(i)/maxV)-0.7);
%    end
%    if E(i)==envStart; %environment A

```

```

%   if X(i)< 0
%       Z(i)=1-abs(abs(Y(i)/maxV)-0.7);
%   else
%       Z(i)=1-abs(abs(Y(i)/maxV)-0.3);
%   end
%   else %environment B
%       if X(i)< 0
%           Z(i)=1-abs(abs(Y(i)/maxV)-0.9);
%       else
%           Z(i)=1-abs(abs(Y(i)/maxV)-0.1);
%       end
%   end
%   end

%Griewank OpX1, EvX1
Z(i)=-1*griewank([X(i) E(i)]);

%Ackley OPX2, DsX2
%   Z(i)=-1*ackley([X(1) X(2) Y(1) Y(2)]);

%Schwefel OPX2, DsX2, EnvX1
%Z(i)=-1*schw([X(1) X(2) Y(1) Y(2) E]);
%   Z(i)=-1*schw([X Y E]);

%Schwefel OPX2, DsX2, EnvX2
%   Z(i)=-1*schw([X(1) X(2) Y(1) Y(2) E(1) E(2)]);
end

%performance is Dp and Ep dependent
%   for i=1:size(X,1)
%       if E(i)==1; %environment A
%           Z(i)=(pi-abs(X(i)/maxV+0.75));
%       else %environment B ==> E==2

```

```

%   Z(i)=(1-abs(X(i)/maxV-0.25));
%   end
%   end

%performance is Dp, Op, and Ep dependent
%   if E(i)==1; %environment A
%       Z(i)=(pi-abs(X(i)/maxV+0.33))+(pi-abs(Y(i)/maxV-0.65));
%   else %environment B ==> E==2
%       Z(i)=(1-abs(X(i)/maxV-0.85))+(1-abs(Y(i)/maxV+0.55));
%   end

%performance is Op and Ep dependent
%   if E(i)==1; %environment A
%       Z(i)=(pi-abs(Y(i)/maxV-0.5));
%   else %environment B ==> E==2
%       Z(i)=(1-abs(Y(i)/maxV+0.5));
%   end

%performance is neither Dp, Op, or Ep dependent
%   Z(i)=perf(X(i)+offset, Y(i)+offset);

% Schwefel function
%Z(i)=-X(i)*sin(sqrt(abs(X(i))))-Y(i)*sin(sqrt(abs(Y(i))));

% Schwefel function
%Z=-X.*sin(sqrt(abs(X)))-Y.*sin(sqrt(abs(Y)));

% Griewank function
%S(i)=X(i)^2+Y(i)^2;
%P(i)=(cos(X(i)))*(cos(Y(i))/sqrt(2));
%Z(i)=S(i)/4000-P(i)+1;

% Griewank function

```

```

%S=X.^2+Y.^2;
%P=(cos(X)./1).*(cos(Y)./sqrt(2));
%Z=S./4000-P+1;
% Ackley function
%S(i)=X(i)^2+Y(i)^2;
%P(i)=(cos(X(i)))*(cos(Y(i))/sqrt(2));
%Z(i)=S(i)/4000-P(i)+1;
% Ackley function
%S=X.^2+Y.^2;
%P=(cos(X)./1).*(cos(Y)./sqrt(2));
%Z=S./4000-P+1;
%maxZ=max(max(Z));minZ=min(min(Z));
if nargout; %assumes nargout value of 3
    varargout{ 1 } = Z;%Z';
    varargout{ 2 } = min(min(Z));%minZ;
    varargout{ 3 } = max(max(Z));%maxZ;
end
end

*****"selection.m"*****

function varargout = selection(isCulling, number, perf_vect, isMutation)
%SELECTION Summary of this function goes here
% Detailed explanation goes here
% 1. isCulling determines the direction (strong vs. weak) of the bias
% 2. number determines either:
%    - the number of solutions to flag for removal

```

```

% - or the number of parents to be selected for GA operations
%3. perf_vect is the performance vector of a (sub)population
global tolerance;
ssize=size(perf_vect,1);
%perf_vect=abs(perf_vect);%perf_vect must always contains positive values
[max_perf_vect loc]=max(perf_vect);
min_perf_vect=min(perf_vect);
if or(max_perf_vect==min_perf_vect,abs(max_perf_vect)-min_perf_vect<=tolerance) %All numbers are equal
    probs=cumsum((1/ssize)*ones(1,ssize));
else
    %Assumes perf_vect is a column vector
    if isCulling %bias toward strongest
        probs=cumsum(max_perf_vect-perf_vect);
    else
        probs=cumsum(perf_vect-min_perf_vect);
    end
end

probs=probs/max(probs);%makes the actual cumulative probabilities
% cprobs=zeros(1,ssize);
% cprobs(1)=probs(1);
% for r=ssize:-1:2
%   cprobs(r)=probs(r)-probs(r-1);
% end

%if everything was right then this MUST be true: probs=cumsum(cprobs)
if isCulling
    %if isCulling or is set, we want to return the indexes of the solutions to remove

```

```

    %fprintf('isCulling=%d, number=%d, isMutation=%d,
length(perf_vect)=%d.\n',isCulling,number,isMutation,length(perf_vect));

    indexes=zeros(1,number);

    for j=1:number
        picked=rand;
        for i=1:ssize
            if picked<=probs(i)%u2b picked<=probs(i)
                indexes(j)=i;
                %Adjust probs(i) value b4 passing ctrl back
                perf_vect(i,1) = max_perf_vect;
                if (max_perf_vect-min(perf_vect)<=tolerance) %All numbers are equal
                    perf_vect = .5*(ssize-j-1)*ones(ssize,1);
                    max_perf_vect=ssize-j-1;
                    %just in case this happens during a generation, we need
                    %to make sure all selected idxs can no longer be
                    %selected since they must all be unique
                    for k=1:j
                        perf_vect(indexes(1,k),1) = max_perf_vect;
                    end
                    perf_vect(loc,1) = max_perf_vect;

                %
                disp('perf_vect values are all equal now');
                %
                perf_vect=perf_vect
                %
                j=j
                %
                loc=loc

                probs=cumsum(max_perf_vect-perf_vect);
            else

```

```

        probs=cumsum(max_perf_vect-perf_vect);%probs=cumsum(max(perf_vect)-
perf_vect);
        end
        probs=probs/max(probs);
        %perf_vect=perf_vect
%        probs=cumsum(cprobs);
%        probs=probs/max(probs);
        break;
    end
end
end
elseif isMutation
    %if isMutation is set, we want to return he indexes of the solutions to mutate
    if ssize==1
        indexes =ones(1,number); %special case
    else
        indexes=zeros(1,number);
        for j=1:number
            picked=rand;
            for i=1:length(perf_vect)
                if picked<=probs(i)
                    indexes(j)=i;
                    break;
                end
            end
        end
    end
end
end
elseif not(isMutation) %crossover case

```



```

indexes=zeros(1,2*number);%doubles since a child requires 2 parents
first=zeros(1,number);
count=0;
not_tired=1;
not_tired_max=10;
j=1;
while j<2*number
    found=0;
    while not(found)
        picked=rand;
        for i=1:ssize
            if picked<=probs(i)
                if not(any(abs(first-i)==0))
                    not_tired=1;%resets value
                    indexes(j)=i;
                    count = count + 1;
                    first(count)=i;
                    indexes(j+1)=mod(i,ssize)+1;%speeds up process
                    j=j+2;
                    found=1;
                    break;
                else
                    not_tired=not_tired+1;
                    if not_tired == not_tired_max
                        not_tired=1;%resets value
                        indexes(j)=i;
                        count = count + 1;
                    end
                end
            end
        end
    end
end

```

```

        first(count)=i;
        indexes(j+1)=mod(i+1,ssize)+1;%speeds up process
        j=j+2;
        found=1;
    end
    break;
end
end
end
end
end
end
end
end
end

%Return Results
if nargout
    varargout{1}=indexes;
end
end

*****"schw.m"*****

function y = schw(x)
% Schwefel function
% Matlab Code by A. Hedar (Nov. 23, 2005).
% The number of variables n should be adjusted below.
% The default value of n = 2.
% Global minimum achieved at x*=(s,s,...,s) where s=420.9687
n = 6;

```

```
s = sum(-abs(x).*sin(sqrt(abs(x))));
```

```
y = 418.9829*n+s;
```

```
*****" ackley.m"*****
```

```
function y = ackley(x)
```

```
%
```

```
% Ackley function.
```

```
% Matlab Code by A. Hedar (Sep. 29, 2005).
```

```
% The number of variables n should be adjusted below.
```

```
% The default value of n =2.
```

```
%
```

```
n = 4;
```

```
a = 20; b = 0.2; c = 2*pi;
```

```
s1 = 0; s2 = 0;
```

```
for i=1:n;
```

```
    s1 = s1+x(i)^2;
```

```
    s2 = s2+cos(c*x(i));
```

```
end
```

```
y = -a*exp(-b*sqrt(1/n*s1))-exp(1/n*s2)+a+exp(1);
```

```
*****" createSocialGroups.m"*****
```

```
function varargout = createSocialGroups()
```

```
%CLUSTER Summary of this function goes here
```

```
% Detailed explanation goes here
```

```
% themes --> Matrix (nthemes x ncols) representing the list of themes
```

```

% pop    --> Matrix (NbrOidx x ncols) representing the pop
% fitnesses --> Column/Row vector containing fitnesses values for the
%         pop
global pop NbrOidx nbrAlleles themes MinKnowledge tolerance memberships;

%max_schewfel=3353.8140;
nthemes=size(themes,1);%used to be length(themes);
fitnesses=pop(1:NbrOidx,nbrAlleles+1);
%adjusting fitness values if necessary
if abs(min(pop(1:NbrOidx,nbrAlleles+1)))<tolerance
    fitnesses(1:NbrOidx,1)=fitnesses(1:NbrOidx,1)-min(pop(1:NbrOidx,nbrAlleles+1))+tolerance;
end
%fitnesses=fitnesses+max_schewfel;
%Step#1: Find topic affinities using 'sørensen' similarity
affinities=zeros(NbrOidx, nthemes);
for i2=1:nthemes
    for i1=1:NbrOidx
        affinities(i1,i2)=similarity('sørensen',themes(i2,:),pop(i1,:));
    end
    %normalize affinities or set values to 1.0 if all null
    if abs(max(affinities(:,i2))) > tolerance
        affinities(:,i2)=affinities(:,i2)/max(affinities(:,i2));
    else
        affinities(1:NbrOidx,i2)=ones(NbrOidx,1);
    end
end
end
%Step#2: Find leaders (most knowledgeable person on their topic)

```

```

[m_v l_i]=max(affinities,[],1);
%enforce minimum knowledge
i3=1;
while i3<=length(m_v)
    if (m_v(i3)<MinKnowledge)
        l_i(i3)=[];
        %themes(:,i3)=[];
        m_v(i3)=[];
    else
        i3=i3+1;
    end
end
%Step#3: Compute influence matrix (memberships) values, scales and normalizes them
memberships=zeros(NbrOidx,length(l_i));
for j2=1:length(l_i)
    for j1=1:NbrOidx
        if (affinities(j1,j2)~=affinities(l_i(j2),j2))%no div by 0
            memberships(j1,j2)=fitnesses(j1)*fitnesses(l_i(j2))/(tolerance+
similarity('cityblock',pop(l_i(j2),:),pop(j1,:)));
        else
            memberships(j1,j2)=1.0;%since affinities will be normalized
        end
    end
end
[val idx]=max(memberships(:,j2));
if max([abs(affinities(idx,j2)) tolerance])==tolerance
    scaled_max = val*(1+sqrt(5))/2;%scale randomly set to golden ratio
else
    scaled_max = val/affinities(idx,j2);

```

```

end
for j1=1:NbrOidx
    if (affinities(j1,j2)~=affinities(l_i(j2),j2))%no div by 0
        %normalize all influence values
        memberships(j1,j2)=memberships(j1,j2)/scaled_max;
    end
end
end
%Step#4: Return Outputs
if nargout
    % varargout{1} = GroupInterest;
    varargout{1} = l_i;
    varargout{2} = affinities;
end
end

*****" crossOver.m"*****

function childA=crossOver(parent1, parent2)
    %Assumptions:
    %A1. Parents are of the same size
    %A2. Parents are passed along with fitness as last column
    % Performs cross-over with more fit pulling less fit
    % We can afford that since we know our domain to be convex
    global nbrAlleles tolerance;
    %this coefficient guarantees a convex solution
    if parent1(nbrAlleles+2)+parent2(nbrAlleles+2)<tolerance % division by zero yields NaN

```

```

    coeff=[0.5 0.5];
else
coeff=[parent1(nbrAlleles+2)/(parent1(nbrAlleles+2)+parent2(nbrAlleles+2)),parent2(nbrAlleles
+2)/(parent1(nbrAlleles+2)+parent2(nbrAlleles+2))];
end
%Change this for GSN as GSN would only require: psize-1;
%childA_=round(coeff(1)*parent1(1:psize-2)+coeff(2)*parent2(1:psize-2));
%childA_=round(coeff(1)*parent1+coeff(2)*parent2);
childA=coeff(1)*parent1+coeff(2)*parent2;

%Both parents should come from same envmt and belong to the same school
%If it is not the case then stronger parent pulls child to its environment
if coeff(1)>=coeff(2)
    childA(1,nbrAlleles:nbrAlleles+1)=parent1(nbrAlleles:nbrAlleles+1);
else
    childA(1,nbrAlleles:nbrAlleles+1)=parent2(nbrAlleles:nbrAlleles+1);
end
childA(1,nbrAlleles+2)=performance(childA(1,1:nbrAlleles));
% if perfA > 1.0
%   childA(1:nbrAlleles)
%   error('Performance value should not exceed 1.0');
% else
%   childA(psize)=perfA;
% end
end

```

```
*****" crossOver2.m"*****
```

```
function childA=crossOver2(parent1, parent2)
    %Assumptions:
    %A1. Parents are of the same size
    %A2. Parents are passed along with fitness as last column
    % Performs cross-over with more fit pulling less fit
    % We can afford that since we know our domain to be convex
    global nbrAlleles tolerance;
    %this coefficient guarantees a convex solution
    if parent1(nbrAlleles+1)+parent2(nbrAlleles+1)<tolerance % division by zero yields NaN
        coeff=[0.5 0.5];
    else

    coeff=[parent1(nbrAlleles+1)/(parent1(nbrAlleles+1)+parent2(nbrAlleles+1)),parent2(nbrAlleles+1)/(parent1(nbrAlleles+1)+parent2(nbrAlleles+1))];
    end
    %Change this for GSN as GSN would only require: psize-1;
    %childA_=round(coeff(1)*parent1(1:psize-2)+coeff(2)*parent2(1:psize-2));
    %childA_=round(coeff(1)*parent1+coeff(2)*parent2);
    childA=coeff(1)*parent1+coeff(2)*parent2;

    %Both parents should come from same envmt and belong to the same school
    %If it is not the case then stronger parent pulls child to its environment
    if coeff(1)>=coeff(2)
        childA(1,nbrAlleles)=parent1(nbrAlleles);
    else
        childA(1,nbrAlleles)=parent2(nbrAlleles);
    end
end
```



```

end
childA(1,nbrAlleles+1)=performance(childA(1,1:nbrAlleles));
% if perfA > 1.0
%   childA(1:nbrAlleles)
%   error('Performance value should not exceed 1.0');
% else
%   childA(psize)=perfA;
% end
end

*****"griewank.m"*****

function y = griewank(x)
%
% Griewank function
% Matlab Code by A. Hedar (Sep. 29, 2005).
% The number of variables n should be adjusted below.
% The default value of n =2.
%
n = 2;
fr = 4000;
s = 0;
p = 1;
for j = 1:n; s = s+x(j)^2; end
for j = 1:n; p = p*cos(x(j)/sqrt(j)); end
y = s/fr-p+1;

*****"crossOver2.m"*****

```

```

function migrate()

global nbrIslands nbrAlleles migrationSize islands;%5 islands of 8 people each encoded with 3
genes: gen1|gen2|evmt|perf

val_idx=zeros(migrationSize,nbrIslands);

% who will move?

who_move=zeros(nbrIslands*migrationSize,nbrAlleles+1);

gIdx=0;

for i=1:nbrIslands
    [val_idx]=sort(islands(:,nbrAlleles+1,i),'descend');
    val_idx(:,i) = idx(1:migrationSize,1);
    for j=1:migrationSize
        gIdx=gIdx+1;
        who_move(gIdx,:)=islands(idx(j,1),:,i);
    end
end

gIdx=0;

%effective migrations occurs according to a ring topology

for i=1:nbrIslands
    for j=1:migrationSize
        gIdx=gIdx+1;
        islands(val_idx(j,i),:,i) = who_move(mod(i-2,nbrIslands)+1,:);
    end
end

end

end

```