FEATURE SUBSET SELECTION VIA MULTI-OBJECTIVE GENETIC
ALGORITHM

A Thesis
Presented to
The Faculty of Graduate Studies
of
The University of Guelph

by
HAO CHI LAC

In partial fulfilment of requirements
for the degree of
Master of Science
August, 2004

© Hao Chi Lac, 2004

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-494-01824-0

Our file Notre référence

ISBN: 0-494-01824-0

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

ABSTRACT

FEATURE SUBSET SELECTION VIA MULTI-OBJECTIVE GENETIC ALGORITHM

Hao Chi Lac

University of Guelph, 2004

Advisor:

Professor Alex Ferworn

Feature (attribute) selection is an important preprocessing step in data mining because most real-world databases were collected for purposes other than data mining. Consequently, many attributes in these databases are irrelevant for the data mining task. Moreover, some data sets have a curse-of-dimensionality problem and eliminating irrelevant attributes allows more accurate models to be built. In this thesis, a new system is proposed for the feature subset selection problem that is based on the wrapper approach. Specifically, this system works by wrapping our proposed multi-objective genetic algorithm around a regular backpropagation neural network.

Acknowledgements

The writing of this thesis has been a wonderful learning experience and it would not have been made possible without the support of the many people who are so dear to me.

First of all, I would like to thank my supervisors, Dr. Alex Ferworn and Dr. Deborah Stacey, for their invaluable help and guidance during this effort. I would also like to express my appreciation to my friends Hai Le and Wing Shiu for supporting me during those tough times. A special thank goes out to Ivan Rubiales; our many discussions have provided me with more insights than you can imagine.

I would also like to thank my family and parents for supporting me in this endeavour, and for not talking me out of doing another degree.

Finally, I would like to dedicate this thesis to my fiancée, Sialinh Chuong, for her endless support and unconditional love. You are my inspiration and my life-line. I love you!

Contents

List of Figures	v
List of Tables.....	vii
Chapter 1.....	1
Introduction	1
1.1 Introduction to the Problem.....	1
1.2 Contents of the Thesis	3
1.3 Thesis Organization	5
1.4 Thesis Achievement.....	5
Chapter 2.....	7
Related Works	7
2.1 Introduction.....	7
2.2 Multi-Objective Optimization	8
2.3 Multi-Objective Optimization Via Genetic Algorithm	11
2.3.1 Wrapper Approach	12
2.3.2 The Simple Genetic Algorithm	14
2.3.3 GA Individual Encoding Schemes.....	14
2.3.4 Least-Crowded Tournament Selection Operator	15
2.3.5 Recombination Operators.....	17
2.3.6 The Fitness Functions	18
2.3.7 Elitism.....	21
2.3.8 Least-Crowded Selection Algorithm.....	21
2.3.9 Lateral Diversity Mechanism	22
2.4 Competing Approaches.....	23
2.4.1 Classical Methods	23
2.4.2 The Filter and Wrapper Approach.....	24
2.4.3 Multi-Objective Tabu Search	27

2.4.4	Multi-Objective Simulated Annealing	28
2.4.5	Multi-Objective Genetic Algorithm	29
2.4.6	Competing Single-Objective Evolutionary Approaches	30
2.4.6.1	Feature Selection Via Constrained Genetic Algorithm	30
2.4.6.2	Commonality-Based Crossover Operator for Feature Selection	33
2.4.6.3	Feature Weighting Representation.....	34
2.4.6.4	Combining Feature Selection and Feature Weighting.....	36
2.4.6.5	Feature Selection using a Feature Index-Based Individual Encoding.....	38
2.4.6.6	Incorporating the Baldwin Effect for Feature Selection	39
2.4.6.7	Incorporating the Lamarckian for Feature Selection	40
2.4.6.8	Feature Selection Via Search for Partially Defined Instances	41
2.4.6.9	Joint Feature Selection and Instance Selection	42
2.4.7	Competing Multi-Objective Evolutionary Approaches	43
2.4.7.1	Non-Dominated Sorting Genetic Algorithm for Feature Selection	45
2.4.7.2	MOEA and Constructive Neural Network for Feature Selection	47
2.4.7.3	Elitist Niched Pareto Genetic Algorithm for Feature Selection.....	47
2.4.7.4	Feature Selection Via MOGA for Clustering	50
2.5	Conclusion.....	51
Chapter 3.....		53
Method Analysis and Comparison.....		53
3.1	Introduction	53
3.2	Problem Statement.....	54
3.3	Algorithm Analysis and Comparison	57
3.3.1	The Back-propagation Algorithm.....	57
3.3.2	The Single-Objective Genetic Algorithm.....	59
3.3.3	The Multi-Objective Genetic Algorithm	62
3.4	Conclusion.....	67
Chapter 4.....		69
Empirical Validation and Analysis		69
4.1	Introduction	69
4.2	10-Fold Cross-Validation.....	70
4.3	Data Collection and Description	72
4.3.1	Ionosphere Database	72
4.3.2	Hepatitis Database.....	73
4.4	Data Format.....	76
4.4.1	Ionosphere Dataset	76
4.4.2	Hepatitis Dataset.....	78
4.5	Accuracy Rate and Model Complexity Comparison.....	79
4.5.1	The BP-CV Approach	80

4.5.2	The SGA-BP-Wrapper-Cross-Validation Approach	83
4.5.3	The MOGA-BP-Wrapper-Cross-Validation Approach	85
4.5.4	Ionosphere Dataset: Parameters Settings and Results	86
4.5.5	Hepatitis Dataset: Parameters Settings and Results	93
4.5.6	Summary of Experiment.....	99
4.6	Comparison of Convergence	99
4.6.1	Ionosphere Dataset: Results of Convergence Property	100
4.6.2	Hepatitis Dataset: Results of Convergence Property	105
4.6.3	Summary of Experiment.....	109
4.7	The Effects of Lateral Diversity	109
4.7.1	Ionosphere Dataset: No Lateral Diversity	110
4.7.2	Hepatitis Dataset: No Lateral Diversity	112
4.8	The Effects of Vertical Diversity	116
4.8.1	Ionosphere Dataset: No Vertical Diversity	118
4.8.2	Hepatitis Dataset: No Vertical Diversity	123
4.9	Conclusion.....	127
Chapter 5.....		129
Conclusion.....		129
5.1	A MOGA Wrapper Approach for Feature Selection.....	129
5.2	Summary of Contribution	130
5.3	Future Work and Conclusion.....	131
Appendix A		133
Bibliography.....		135

List of Figures

Figure 1.1: The Proposed MOGA Wrapper Algorithm.....	4
Figure 2.1: Decision and Objective Space (From [Deb, 2001])	9
Figure 2.2: The Wrapper Approach (From [Freitas, 1998]).....	13
Figure 2.3: Backpropagation Network	18
Figure 2.4: The Filter Approach (From [Freitas, 1998])	25
Figure 4.1: 10-Fold Cross-Validation Procedure	72
Figure 4.2: Module in BP-CV Approach	80
Figure 4.3: Method to Avoid Overtraining (From [Hecht-Nielsen, 1991]).....	82
Figure 4.4: SGA-BP-Wrapper Algorithm	83
Figure 4.5: Parent Population (MOGA-BP-Wrapper-CV): Ionosphere	103
Figure 4.6: Parent Population (SGA-BP-Wrapper-CV): Ionosphere	104
Figure 4.7: Pareto-Optimal Front (MOGA-BP-Wrapper-CV): Ionosphere	105
Figure 4.8: Parent Population (MOGA-BP-Wrapper-CV): Hepatitis	107
Figure 4.9: Parent Population (SGA-BP-Wrapper-CV): Hepatitis.....	108
Figure 4.10: Pareto-Optimal Front (MOGA-BP-Wrapper-CV): Hepatitis.....	108
Figure 4.11: Parent Population (MOGA-BP-Wrapper-CV: No Lateral Diversity): Ionosphere .	110
Figure 4.12: Pareto-Optimal Front (MOGA-BP-Wrapper-CV: No Lateral Diversity): Ionosphere	111
Figure 4.13: Parent Population (MOGA-BP-Wrapper-CV: No Lateral Diversity): Hepatitis	113
Figure 4.14: Pareto-Optimal Front (MOGA-BP-Wrapper-CV: No Lateral Diversity): Hepatitis	114

Figure 4.15: Parent Population (MOGA-BP-Wrapper-CV: No Vertical Diversity): Ionosphere	119
Figure 4.16: Optimal Front (MOGA-BP-Wrapper-CV: No Vertical Diversity): Ionosphere	120
Figure 4.17: Parent Population (MOGA-BP-Wrapper-CV: No Vertical Diversity): Hepatitis...	124
Figure 4.18: Pareto-Optimal Front (MOGA-BP-Wrapper-CV: No Vertical Diversity): Hepatitis	
.....	125

List of Tables

Table 2.1: Non-Dominated Set Example.....	10
Table 4.1: Dataset Attributes: Hepatitis	74
Table 4.2: Missing Attribute Values: Hepatitis.....	75
Table 4.3: Network Parameters: Ionosphere	88
Table 4.4: Accuracy Rate and Model Complexity Results: Ionosphere.....	90
Table 4.5: Features Selected for Fold 1: Ionosphere	93
Table 4.6: Network Parameters: Hepatitis.....	94
Table 4.7: Accuracy Rate and Model Complexity Results: Hepatitis	96
Table 4.8: Features Selected for Fold 1: Hepatitis	97
Table 4.9: Execution Time	98
Table 4.10: Test of Significance: Ionosphere.....	101
Table 4.11: Test of Significance: Hepatitis	105
Table 4.12: Number of Pareto-Optimal Solutions: Ionosphere.....	112
Table 4.13: Statistics for the Number of Pareto-Optimal Solutions Discovered: Ionosphere.....	112
Table 4.14: Number of Pareto-Optimal Solutions: Hepatitis	115
Table 4.15: Statistics for the Number of Pareto-Optimal Solutions Discovered: Hepatitis	116
Table 4.16: Number of Pareto-Optimal Solutions: Ionosphere	122
Table 4.17: Statistics for the Number of Pareto-Optimal Solutions Discovered: Ionosphere.....	122
Table 4.18: Number of Pareto-Optimal Solutions: Hepatitis	126

Table 4.19: Statistics for the Number of Pareto-Optimal Solutions Discovered: Hepatitis 127

Chapter 1

Introduction

1.1 Introduction to the Problem

Endeavours such as the human genome project have created an explosion in the amount of data collected and stored in real-world databases. The data contained in these databases tend to be both complex and large in size. Moreover, most real-world datasets are collected for purposes other than data mining and thus may contain many irrelevant features that may impede the data mining algorithm from effectively extracting useful relationships [Pappa et al., 2002]. Consequently, a typical and necessary step in the data mining process is the elimination of irrelevant features, a task also known as feature subset selection. By eliminating the irrelevant features, the performance in terms of speed and accuracy of a data mining algorithm can be significantly improved.

From a data mining perspective and for our purpose, three metrics are used to measure the performance of a given feature selection algorithm: (i) the ability to increase the accuracy rate for a given data mining algorithm, (ii) the ability to increase comprehensibility, increase cost savings, and mitigate the curse of dimensionality assuming the dataset is not under sampled by minimizing model complexity (i.e. smaller feature subsets are better), and (iii) the ability to provide “greater

selection” for the user. In this thesis, a method is considered to provide “greater selection” if it provides the user with an opportunity to choose a solution from a set of *Pareto-optimal solutions* based on his/her background knowledge and criteria. The definition of a *Pareto-optimal solution* is given later in Chapter 2 (see Definition 2.9). This third metric is included because the feature subset selection problem is inherently a multi-objective optimization problem (MOOP) and any proposed method should capitalize on this characteristic to provide the user with greater selection in accomplishing his/her task.

The search for a single optimal feature subset in a large complex dataset is considered to be NP-hard [Bins and Draper, 2001]. This complexity of the feature space automatically rules out the use of exhaustive search methods and so many researchers have focused on meta-heuristic search methods to tackle such problems [Pappa et al., 2002; Bala et al., 1997; Emmanouilidis et al., 1999, 2000]. Unfortunately, most of these meta-heuristic search methods are not suitable solutions to the feature subset selection problem or do not fulfill all of our metric requirements because they either optimize a single objective (i.e. single objective genetic algorithm), or implicitly perform feature detection (i.e. multi-layer perceptrons) and thus do not minimize model complexity. This is the situation found in the classifying problem for the ionosphere data set that we will examine, among others. Consequently, this thesis proposes a competitive wrapper approach that incorporates a back-propagation neural network and a multi-objective genetic algorithm, and shows that it is possible to find a Pareto-optimal set of feature subsets that performs at least as well as when the entire feature set is used for data classification.

1.2 Contents of the Thesis

As mentioned earlier, the task of selecting an *optimal* subset of features for a given induction algorithm is NP-hard. The qualifying term *optimal* will be more formally defined in a later chapter. To achieve the task of finding not only one *optimal* solution but a Pareto-optimal set of solutions, we propose a Multi-Objective Genetic Algorithm (MOGA) that works by wrapping around a given induction algorithm; in this thesis the induction algorithm is the back-propagation (BP) algorithm. The general outline of the algorithm is given in Figure 1.1.

Our approach is built upon several past research efforts, the first being the wrapper approach [Kohavi and John, 1997]. The wrapper approach defining characteristics is the large emphasis it places on the induction algorithm in the feature selection process. In our method, the MOGA acts as the wrapper. Our MOGA is just a simple genetic algorithm [Goldberg, 1989] modified to handle MOOPs. As shown in Figure 1.1, our proposed MOGA starts off by creating an initial parent population of size N , and thereafter the BP (i.e. inducer) is applied to each solution. Assuming that each instance (i.e. exemplar) is described by a vector of feature values and a class label, application of the BP to each solution means that the vector elements of each instance corresponding to the selected features of the solution are used as inputs. The output from the BP and the number of features selected becomes the performance measure (i.e. fitness) of a solution. Based on the fitness, the solutions are subject to a selection procedure, and the selected solutions are then subjected to cross-over and mutation operators to produce an offspring population of equal size to that of the parent population. Thereafter, the two populations are merged and the resulting population of size $2N$ is sorted into different fronts by a non-dominated sorting module; the idea of using the non-dominated sorting concept in GAs was introduced by [Goldberg, 1989]. Next, we place non-dominated solutions into a separate set (i.e. the elite set E_i) [Zitzler and Thiele, 1998]. If the size $|E_i|$ of the elite set is greater than N , we apply our *least-crowded*

selection algorithm that selects N least crowded solutions to be in E_{i+1} . Thereafter, if the solutions in the elite set satisfy the stopping criterion, the algorithm terminates; if not, the lateral diversity module is performed [Deb and Goel, 2001]. Depending on the size of the current elite set E_i , the lateral diversity first either replaces all solutions from the first front with the solutions of E_i , or adds all the solutions of E_i to the first front. Thereafter, N solutions are selected that make up the new population by selecting n solutions from each front starting with the first front according to some formula. The process then repeats itself starting from step 3.

1. Create initial random population $P_{i=0}$ of size N .
2. Evaluate each solution in P_i by using the induction algorithm I , where P_i is the population for the i th generation.
3. Perform selection, crossover, and mutation on P_i to get Q_i .
4. Evaluate each solution in Q_i using I .
5. $R_i = P_i \cup Q_i$. Sort R_i into non-dominated levels F_n , where $n \leq N$.
6. Place non-dominated solutions from the first front F_1 into elite set E_i . Delete any duplicate or dominated solutions from E_i . If the number of solutions $|E_i| > N$, use the *least-crowded selection algorithm* to select N solutions. Go to step 8 if stopping criterion is satisfied, otherwise go to step 7.
7. Perform the lateral diversity mechanism as follows. If $|E_i|$ is equal to N , replace all the solutions in F_1 with solutions from E_i . Otherwise if $|E_i| < N$, add E_i to F_1 and eliminate any duplicate solutions; if $|F_1| > N$, eliminate $|F_1| - N$ solutions from F_1 by using the *least-crowded selection algorithm*. Thereafter, select $|x_n|$ solutions from front F_n using the *least-crowded selection algorithm*, where $|x_1| + |x_2| + \dots + |x_n|$ equals size N . The number of solutions x_n selected from each front n is determined by some distribution used. The resulting N solutions selected forms P_{i+1} . Go back to step 3.
8. Return E_i and terminate execution.

Figure 1.1: The Proposed MOGA Wrapper Algorithm

1.3 Thesis Organization

We have five chapters in total in this thesis. In chapter 2, we examine the research efforts our approach builds upon in more detail. Furthermore, we present and discuss in detail some competing evolutionary approaches for the feature subset selection problem. In chapter 3 we examine the feature subset selection problem and provide a definition of a *relevant* or *optimal* feature subset. In addition, we analyze three algorithms with increasing complexity for the feature subset selection problem: (i) the back-propagation algorithm, (ii) the single-objective genetic algorithm, and (iii) our multi-objective genetic algorithm. The comparison will focus on the algorithms' capabilities in terms of obtaining the three goals of maximizing accuracy, minimizing model complexity (i.e. smaller feature subset), and increasing selection for the user (i.e. allowing the user the ability to choose a Pareto-optimal solution from a set of such solutions), and the computational complexities of the algorithms. Chapter 4 will provide an outline of our experimental work performed. Chapter 5 will give some concluding remarks and some future works.

1.4 Thesis Achievement

This thesis shows that it is possible to build a system that can identify a Pareto-optimal set of feature subsets that performs at least as well as when the entire feature set is used for data classification. The significance of finding an “optimal” feature subset is that it simplifies the data analysis due to the smaller model discovered. Moreover, a smaller model can help reduce cost by eliminating the need to collect data for irrelevant attributes. Moreover, because our system attempts to find a set of Pareto-optimal solution, it becomes more useable to the user who can choose which solution(s) to use based on his/her criteria and expertise. Thus, the research

reported in this thesis provides a first step towards developing a system that can help reduce complexity in data analysis and economic costs.

Chapter 2

Related Works

2.1 Introduction

All intelligent agents are confronted with the common task of where to focus their attention. Software system designers must decide which features to use achieve optimal performance, mathematicians must decide which aspects of a problem are relevant in deriving a correct model of the problem, and so forth. The ability to discriminate between the relevant and irrelevant parts of a problem by intelligent agents inevitably implies the concept of learning from experience. In the feature subset selection problem, a learning algorithm is confronted with the problem of selecting a feature subset on which to focus on, while discarding all others. Except for only the most trivial case, the majority of complex real-world problems are intractable when it comes to finding one *optimal* subset, let alone a set of optimal subsets. Consequently, meta-heuristic methods have become popular in tackling NP-hard problems. In this chapter, we will discuss previous research efforts that our work builds upon and present in detail some competing evolutionary approaches.

This chapter is divided into a few sections. In Section 2.2 we discuss multi-objective optimization and show that it is appropriate to cast the feature subset selection problem as a

MOOP. In Section 2.3 we examine in detail previous research efforts that our approach builds upon. This section will also introduce our *least-crowded selection algorithm* for maintaining diversity in a GA population. In Section 2.4 we are going to present in detail competing methods in the evolutionary algorithm area with respect to the feature subset selection problem. In Section 2.5 we provide a brief summary of this chapter and give an outline of our research focus in the following chapters.

2.2 Multi-Objective Optimization

The feature subset selection problem is inherently multi-objective. That is, there is a tradeoff between training accuracy and model complexity (i.e. the number of features). Moreover, it is difficult to determine, for a given problem, the relative importance of these two competing goals. Consequently, in this thesis we have adopted the strategy of casting the feature subset selection problem as a Multi-Objective Optimization Problem (MOOP) [Winston, 1994]. The following is the general form of the MOOP [Deb, 2001]:

$$\left. \begin{array}{ll} \text{Minimize/Maximize} & f_m(\mathbf{x}), \quad m=1, 2, \dots, M; \\ \text{subject to} & g_j(\mathbf{x}), \quad j=1, 2, \dots, J; \\ & h_k(\mathbf{x}), \quad k=1, 2, \dots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i=1, 2, \dots, n. \end{array} \right\} \quad (2.1)$$

In the above formulation, $\mathbf{x}$ denotes a solution and is a vector of n decision variables: $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$. There are M objective functions $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x}))^T$; each can be either minimized or maximized. The last three sets of equations are constraints. A lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound are variable bounds, restricting each decision variable x_i to take a value within these boundaries. These bounds form a *decision variable space* D , or the decision space. Moreover, there is J inequality and K equality constraints that are denoted by $g_j(\mathbf{x})$ and $h_k(\mathbf{x})$ constraint functions, respectively. Hence, a solution $\mathbf{x}$ that does not satisfy all of the $(J + K)$

constraints and all of the variable bounds is considered an *infeasible solution*. If all constraints and bounds are satisfied, the solution is considered a *feasible solution*. The set of all feasible solutions is called the *feasible region*, or S . An additional space that exists in multi-objective optimization created by the objective functions is a multi-dimensional space called the *objective space*, Z . For each solution x in the decision variable space, there exists a point in the objective space, denoted by $f(x) = z = (z_1, z_2, \dots, z_M)^T$. See Figure 2.1 below.

Figure 2.1: Decision and Objective Space (From [Deb, 2001])

We have thus far examined only partially the characteristics that make up a MOOP. To complete the picture, three additional concepts are required: the convexity or non-convexity of a MOOP, domination, and Pareto-optimality.

In MOOPs, whether the spaces (decision and objective space) are convex is important since many algorithms handle convex MOOPs well, but confront difficulty in solving non-convex MOOPs [Deb, 2001]. For a definition of a convex function, refer to [Winston, 1994]. One interesting point to note from the definition of a convex function is that a feasible search space

formed by non-convex constraint functions will enclose a convex region. The following definition provides a useful summary [Deb, 2001]:

Definition 2.5. A multi-objective optimization problem is convex if all objective functions are convex and the feasible region is convex (or all inequality constraints are non-convex and equality constraints are linear).

Now we give a more precise definition of domination and Pareto-optimality. The concept of domination is defined as follows [Deb, 2001]:

Definition 2.6 (weak dominance). A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both conditions 1 and 2 are true:

1. The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives, for all $j = 1, 2, \dots, M$.
2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective $j \in \{1, 2, \dots, M\}$.

Violation of any of the above conditions implies that solution $x^{(1)}$ does not dominate solution $x^{(2)}$.

This dominance relationship is also referred to as a weak dominance relation. A modification of this definition yields a strong dominance relation and gives rise to the following two definitions:

Definition 2.7 (strong dominance). A solution $x^{(1)}$ strongly dominates a solution $x^{(2)}$ if the former is strictly better than the latter in all M objectives [Deb, 2001].

Definition 2.8 (Non-dominated set). Among a set of solutions P , the non-dominated set of solutions P' are those that are not dominated by any member of the set P [Deb, 2001].

The non-dominated set of Definition 2.8 is also known as non-dominated fronts if the vectors are in objective space.

To illustrate Definitions 2.6 to 2.8, we make use of the solutions in Table 2.1.

Table 2.1: Non-Dominated Set Example

Solution	f_1	f_2
1	0.31	5.90
2	0.22	7.09
3	0.66	3.65
4	0.21	5.90
5	0.79	3.97
6	0.27	6.93
7	0.58	4.52

In Table 2.1, f_1 and f_2 corresponds to the two objectives to be optimized. Assuming that we are trying to minimize f_1 and f_2 , the resulting non-dominated sets (i.e. fronts) based on Definitions 2.6 to 2.8 are $\text{Front}_0 = \{3, 4, 7\}$ and $\text{Front}_1 = \{1, 2, 5, 6\}$. Moreover, from the definitions of strong dominance and weak dominance, the following relationships exist between Front_0 and Front_1 :

- $\{3\}$ strongly dominates $\{5\}$, and $\{4\}$ strongly dominates $\{2, 6\}$;
- $\{4\}$ weakly dominates $\{1\}$;
- $\{7\}$ is non-dominating to $\{1, 2, 5, 6\}$, $\{3\}$ is non-dominating to $\{1, 2, 6\}$, and $\{4\}$ is non-dominating to $\{5\}$.

Notice that solution 7 is non-dominating to all the solutions in Front_1 , but yet it belongs in Front_0 . This still satisfies the definition of non-dominated set (Definition 2.8) since solution 3 and 4 strongly and weakly dominates at least one solution in Front_1 , respectively. At the same time, solution 7 is non-dominating to every other solution in Front_0 .

Utilizing Definition 2.6 and Definition 2.7, Pareto-optimality can now be defined as:

Definition 2.9 (Pareto-optimal). A solution i in P' is Pareto optimal if and only if there does not exist a solution j in P that adheres to the conditions of Definition 2.6, where solution i represents $x^{(2)}$ and solution j represents $x^{(1)}$ in that definition. In addition, when the set P is the entire search space ($P = S$), the resulting non-dominated set P' is called the *Pareto-optimal* set, and the corresponding vector in objective space is called the *Pareto-front* [Deb, 2001].

2.3 Multi-Objective Optimization Via Genetic Algorithm

A number of concepts in the area of multi-objective optimization were introduced in the previous section to help put into perspective our approach to the feature subset selection problem. In this section, we will discuss previous work done in the areas of data mining and genetic algorithms that utilize those concepts and show how our approach has built upon them.

2.3.1 Wrapper Approach

The flow of the algorithm in Figure 1.1 embodies the wrapper approach to feature subset selection. As mentioned in Chapter 1, the wrapper approach proposed by [Kohavi and John, 1997] for the purposes of feature subset selection is closely coupled to the inducer (i.e. induction algorithm) that will be applied to the selected features. Refer to Figure 2.2.

As illustrated in Figure 2.2, the wrapper-based approach starts off requiring the input of the entire feature set into a learning algorithm (i.e. a search algorithm). Thereafter, the learning algorithm generates a candidate feature subset which is given to an induction algorithm to evaluate. The specifics of the evaluation of a candidate feature subset depends on the induction algorithm, but the feature values used for evaluation for the instances (i.e. the exemplars in the dataset) corresponds to the selected features. Since there are many candidate feature subsets generated by the learning algorithm, the inducer is run many times, each time with a different candidate feature subset. The performance of the inducer in each run is used to evaluate the quality of the corresponding feature set. Note, also that the inducer is run as a “black box”; the feature selection method operates without any knowledge of the internal operations of the inducer. Only the output of the inducer is used [Freitas, 1998]. After the candidate feature subsets are evaluated by the induction algorithm, the process either stops because a stopping criterion has been satisfied or continues executing. If the stopping criterion is satisfied, the most *optimal* solutions or solutions are returned depending on the actual learning algorithm.

Figure 2.2: The Wrapper Approach (From [Freitas, 1998])

On the other hand, if the stopping criterion is not satisfied, the candidate feature subsets evaluated by the induction algorithm is given back to the learning algorithm which generates a new set of candidate feature subsets based on the updated information from the inputted candidate feature subsets. The specifics of the updated information from the input candidate feature subsets depend upon the objective function(s) of the feature subset selection problem.

2.3.2 The Simple Genetic Algorithm

The algorithm of Figure 1.1 is essentially a Simple Genetic Algorithm (SGA) that is commonly found in literature [Goldberg, 1989] with modifications to handle MOOPs. Using the notation from Figure 1.1, the SGA starts off by creating an initial population P_0 and then the solutions in P_0 are subject to an objective function f to obtain a fitness value for each solution. To generate the next generation population P_{i+1} , the solutions in P_i are subject to selection based on their fitness (i.e. objective) value, crossover, and mutation. Thereafter, the solutions of P_{i+1} are evaluated by f again and then subject to selection, crossover, and mutation to produce the next generation population. This process continues until a stopping criterion is satisfied, in which the optimal solution is returned and execution is terminated. Notice from Figure 1.1 that our algorithm maintains two populations: P_i and a population Q_i . Unlike the SGA, all the solutions from the combined populations (step 5) compete against each other for the chance to be in subsequent generations. Moreover, our approach maintains a separate population of elite individuals (or solutions) that receive higher preference when selection for mating for subsequent generations is made. In the SGA, usually one or a few highly fit individuals get an automatic pass to the next generation.

2.3.3 GA Individual Encoding Schemes

The use of a genetic algorithm requires the selection of an appropriate encoding scheme for a GA individual (i.e. solution) that reflects the form of the solution to the actual problem. In the GA literature, a number of encoding scheme exists [Goldberg, 1989]. In our approach, we use binary encoding to represent the features of a given dataset. A binary encoding is just a linear string of 0s and 1s wherein each bit represents a feature in the problem. Note that order is important since each bit position is mapped to a feature in the problem. In our approach, a standard bit encoding

for feature selection is used for an individual $a = (\alpha_1, \alpha_2, \dots, \alpha_d)$ in the GA population that consisted of a d -element binary-valued vector, where α_i represents the i -th bit (gene) and d is the number of original features. An element in the binary-valued vector assumes a value of 0 if the i -th feature is excluded from the subset and a 1 if it is present in the subset [Siedlecki and Sklansky, 1989].

2.3.4 Least-Crowded Tournament Selection Operator

Similarly to GA individual encoding, there is a large body of literature on the selection, crossover, and mutation operators. The first genetic operator, the selection (or reproduction) operator is responsible for making duplicates of good solutions and eliminating bad solutions in a population, while keeping the population size constant. Essentially, the selection operator is used to exploit the search space. This is achieved by performing the following tasks:

1. Determine good (usually above average) solutions in a population by using solution fitness value(s).
2. Make multiple copies of good solutions.
3. Ensure multiple copies of good solutions can be placed in the population by eliminating worse solutions from the population.

A number of methods such as *tournament selection*, *proportionate selection* and *ranking selection* exist to accomplish the above tasks to create the mating pool [Goldberg and Deb, 1991]. However, the selection methods for a single-objective GA cannot be applied to a MOOP since the relative importance of each objective is not known; that is, for a given solution, there does not exist a single scalar value indicating its fitness level so that it could be ranked relative to other solutions in the population. To circumvent this problem for our *least-crowded tournament selection operator*, we use the concept of domination and De Jong's crowding model [De Jong, 1975].

In our *least-crowded tournament selection operator*, the concept of domination is used to classify solutions into non-dominating fronts that satisfies Definitions 2.6, 2.7, and 2.8. Our *non-dominated sorting algorithm* works as follows. For each solution in the population, a *domination count* n_i indicating the number of solutions which dominate the solution i [Srinivas and Deb, 1994]. At the end of this procedure, all solutions with domination equal to zero are removed from the population and placed in the first non-dominated front. The above procedure is then repeated to get the second non-dominated front. This process continues until all solutions are placed in their proper non-dominated front.

The classification of the solutions into different non-dominating fronts only partially solves the problem of determining the relative importance of each solution to other solutions in the population. There is still the problem of determining the importance of each solution to other solutions in the same non-dominated front. To solve this problem we turn to the work of [De Jong, 1975]. [De Jong, 1975] developed a strategy of favouring solutions that are less crowded in the search space. In his doctoral dissertation, he used a crowding model to introduce diversity among solutions in a GA population to maintain multiple optimal solutions by discouraging the crowding of solutions anywhere in the search space. Since one of the goals in MOOPs is to find a uniform distribution of solutions in the Pareto-optimal front, De Jong's concept of crowding provides a good metric for selecting solutions in a given front.

Based on our *non-dominated sorting algorithm* and [De Jong, 1975], we are now in a position to give our selection operator. Letting P be the set containing all the solutions from which to choose for the tournament, and letting P' be the set containing winners of each tournament played, two scenarios are possible with our *least-crowded tournament selection operator*: (i) a dominating scenario, and (ii) a non-dominating scenario. In the dominating scenario, two solutions s_1 and s_2 are selected randomly from a set P for a given tournament. If s_1 dominates s_2 , then s_1 is the winner. Otherwise, s_2 dominates s_1 , and s_2 is declared the winner. The winner is then placed in P' . In the non-dominating scenario, for each solution s_i selected randomly from P

for a given tournament, where $i = 1, 2$, the distance d_{ij} from s_i to each member of j in P' is first calculated. Then the smallest distance d'_{ij} is noted for each solution s_i and the solution with the larger d'_{ij} is chosen as the winner and added to P' . Though our tournament selection operator can be played with any number of solutions, all of our experiments are run with just two players to minimize the time complexity. Moreover, our tournament selection operator is applicable in both the decision variable space and in the objective space; in our experiment we apply it only in the objective space because our goal is to search for a Pareto-optimal front.

2.3.5 Recombination Operators

The crossover operator is applied next to the solutions (or strings) of the mating pool. Note that the selection operator cannot create any new solutions in the population. It only makes copies of better solutions to the detriment of comparatively less fit solutions. Thus, it is up to the crossover operator to create new solutions. Like the selection operator, there exists a number of crossover operators in the GA literature [Spears, 1998], but in almost all crossover operators, two strings are randomly selected from the mating pool and some portions of the strings are exchanged to produce two new strings. The hope of this exchange is that better solutions would be created through recombination of two solutions.

Like the crossover operator, the mutation operator helps to create new solutions and thus acts as an exploration (search) mechanism. With respect to bit encoding, the bitwise mutation operator changes a 1 to a 0, and vice versa, with a mutation probability of p_m . The mutation operator essentially alters a solution locally in an attempt to create a better solution and to add diversity to the population.

2.3.6 The Fitness Functions

The objective or fitness function provides a performance measure of a GA individual and is problem dependent. In our approach, two objective functions are used. The first objective function is simply the number of features selected by the GA. The other objective function utilizes a backpropagation neural network to obtain an accuracy rate measure for a given GA individual [Oliveira et al., 2003]. Taking its inspiration from biological neurons in the human brain, the back-propagation neural network consists of an input layer, a hidden layer, and an output layer, together with a feed-forward and a backward error process as shown in Figure 2.3.

Figure 2.3: Backpropagation Network

Each neuron in the input layer represents each independent variable and thus takes on a value taken on by that variable. The lines connecting the neurons of the input layer to the neurons of the hidden layer are weights. These weights are analogous to the synaptic strength between neurons in the human brain, thus representing the activation energy of one neuron on the next layer's neuron. The hidden layer is also made up of neurons, but the values of these continuously

change during training. Weights are also connected from the hidden layer to the output layer. Though not indicated in Figure 2.3, the hidden and output layer each has a bias neuron with connections to each of the neurons in its associated layer. Neuron biases can aid the network to converge more quickly during training. For a more detailed introduction of the backpropagation algorithm see [Fausett, 1994] and [Haykin, 1999].

The back-propagation algorithm is used to train the weights in the neural network so that new, never-before seen information entering the input layer will produce sufficiently accurate results in the output layer. The actual training involves two distinct stages – a feedforward stage and a backpropagation of error stage. See Figure 2.3.

During feedforward, each input neuron ($X_i, i = 1, \dots, n$) receives its corresponding feature value and broadcasts this value to each of the hidden neurons ($Z_j, j = 1, \dots, p$). Each hidden neuron sums its weighted input values,

$$z_in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij} \quad (2.2)$$

and then applies its activation function to compute an output value,

$$z_j = f(z_in_j). \quad (2.3)$$

z_j is then sent to each output neuron. In this experiment, there is only one output neuron. Each output neuron ($Y_k, k = 1, \dots, m$) sums its weighted input values,

$$y_in_k = w_{0k} + \sum_{j=1}^p z_j w_{jk} \quad (2.4)$$

and computes its activation by applying its activation function,

$$y_k = f(y_in_k) \quad (2.5)$$

to form the response y_k of the net for the given input pattern.

During the back-propagation of error, each output neuron ($Y_k, k = 1, \dots, m$) receives its class label and from it computes its error information term,

$$\delta_k = (t_k - y_k) f'(y_in_k). \quad (2.6)$$

δ_k is used to distribute the error at the output neurons Y_k back to all the neurons in the hidden layer. This distribution of error information is used to correct the weight and bias terms connecting the hidden layer to the output layer,

$$\Delta w_{jk} = \alpha \delta_k z_j, \Delta w_{0k} = \alpha \delta_k. \quad (2.7)$$

where the learning rate α for the network is between 0 and 1. In a manner similar to δ_k , the factor δ_j ($j = 1, \dots, p$) is computed for each hidden neuron. However, since the error from the hidden neurons does not need to be propagated back to the input layer, it is only used to update the weights connecting the hidden layer to the input layer. It is computed by multiplying the derivative of its activation function by δ_{in_j} ,

$$\delta_j = \delta_{in_j} f'(z_{in_j}), \quad (2.8)$$

where δ_{in_j} is the summation of the weights w_{jk} connecting the hidden layer to the output layer multiplied by the error information term δ_k , and excluding the bias term w_{0k} . The weight correction terms connecting the input layer to the hidden layer is computed in a manner similar to that of w_{jk} ,

$$\Delta v_{ij} = \alpha \delta_j x_i, \Delta v_{0j} = \alpha \delta_j. \quad (2.9)$$

After all the required corrections are determined, the weight and bias terms are updated as follows:

$$\Delta w_{jk}(t+1) = \alpha \delta_k z_j + \mu \Delta w_{jk}(t), \quad (2.10)$$

and

$$\Delta v_{ij}(t+1) = \alpha \delta_j x_i + \mu \Delta v_{ij}(t), \quad (2.11)$$

where the momentum rate μ is between 0 and 1, not including the end points.

2.3.7 Elitism

A variation of the SGA is the inclusion of elitism that permits the fittest solution or a few of the best solutions in a given generation an automatic pass into the next generation (Goldberg, 1989). Our approach implements elitism by maintaining a separate elite set containing non-dominated solutions. Unlike the SGA, N solutions are allowed to go on to the next generation and these N solutions are selected based on our novel *least-crowded selection algorithm* which will be described in a later section. An important issue to address with respect to maintaining a separate elite set is how to handle the size of the elite set in later stages of the simulation. That is, since our approach maintains $2N$ solutions and the size of the elite set is capped at N , in the later stages of the simulation the number of elite solutions will more than likely surpass N . To ensure an elite set size of N or less, we propose a novel *least-crowded selection algorithm* that selects N least crowded solutions when the elite set size exceeds N .

2.3.8 Least-Crowded Selection Algorithm

Our *least-crowded selection algorithm* is based on the concept of crowding proposed by De Jong as described earlier, and works as follows. Adopting the notation of Figure 1.1, let E_i be the elite set for the i th generation containing N' solutions, where $N' > N$. Also, let E_{i+1} be the elite set for the next generation and set it to be empty. To select the N solutions from E_i to include in E_{i+1} , the first task is to find two solutions s_1 and s_2 that is the furthest from each other; s_1 and s_2 is then removed from E_i and added to E_{i+1} . If more than two such solutions exist, a pair of solutions that are furthest from each other are randomly selected; this strategy is used just to keep computational overhead to a minimum. To find s_3 , a number of calculations are performed. First, a set D_{jk} containing all the distance d_{jk} of each member j in E_i from each member k in E_{i+1} is

calculated. Thereafter, the smallest value d_{jk}^s in the set of D_{jk} is noted for each member j and placed in an empty set L . A solution with a value d_{jk}^s corresponding to the same largest value d_{jk}^s in L is removed from E_i and added to E_{i+1} ; this solution is s_3 . If more than one member j from E_i contains the same value d_{jk}^s , the member to include in E_{i+1} is selected randomly. To find $s_4, \dots, s_N$, the above process to find s_3 is repeated until N solutions are in E_{i+1} .

2.3.9 Lateral Diversity Mechanism

An extremely important issue in the GA literature is the balance between exploitation versus exploration. In MOGA, too much emphasis placed on exploiting the search space in the form of elitism may cause the true Pareto-optimal front to be missed if the GA preserves non-dominated solutions that lie away from the Pareto-optimal front. The result is that the GA may inadvertently converge to a local Pareto-optimal front because the elitism mechanism deletes most solutions belonging to other fronts, and to offset this would require an equally powerful explorative mechanism. In this thesis, a “local” Pareto-optimal front is similar to when a backpropagation algorithm prematurely converges to a local optimum instead of the global optimum. Consequently, to offset the elitism mechanism in our approach described earlier, we include a lateral diversity mechanism that not only favours the selection of elite solutions along the first front, but to force the selection of solutions along other fronts [Deb and Goel, 2001]. The lateral diversity mechanism proposed by [Deb and Goel, 2001] works as follows.

First, the population $R_i = P_i \cup Q_i$ is sorted for non-domination. Assume that the number of non-dominated fronts in the population is K . Since our approach uses the geometric distribution, the maximum number of individual allowed in the i th front ($i = 1, 2, \dots, K$) in the new population of size N is

$$n_i = N [(1 - r) / (1 - r^K)] r^{i-1}. \quad (2.12)$$

Since $r < 1$, the maximum allowable number of individuals in the first front is highest. Thereafter, the number of solutions permitted in each front is exponentially reduced. Although equation (2.12) denotes the maximum allowable number of solutions n_i in each front i , in a population, there may not exist exactly n_i solutions in front i . To resolve this problem, the procedure starts from the first front and the number of individuals in the first front are counted. Suppose that there are n'_1 solutions. If $n'_1 > n_1$, we would only choose n_1 solutions using the *least-crowded selection algorithm* described above. On the other hand, if $n'_1 < n_1$, we would choose all n'_1 solutions and note the difference $r_1 = n_1 - n'_1$. The maximum allowed number of solutions in the second front is now increased to $n_2 \rightarrow n_2 + r_1$. Thereafter, the actual number of solutions n'_2 present in the second front is counted and compared with n_2 as above. This procedure is continued until all N are selected.

2.4 Competing Approaches

There is a large body of literature in the data mining field dealing with the feature subset selection problem. In this section we focus our examination on competing work in the evolutionary area in combination with the wrapper approach; however, before we proceed we will put into context our reasons for opting for evolutionary method as the wrapper instead of other methods.

2.4.1 Classical Methods

As mentioned earlier, the feature subset selection problem can be cast as a MOOP. This implies that traditional mathematical programming techniques such as the weighted sum method, the ϵ -

Constraint method, and goal programming methods can be applied. In this thesis, we refer to these methods as classical methods to distinguish them from evolutionary methods.

Examination of the most popular classical methods reveals that they have a number of difficulties in solving MOOPs, especially if the goal is to find multiple Pareto-optimal solutions. The following difficulties characteristic of classical methods are [Deb, 2001]:

1. Only one Pareto-optimal solution is found in one simulation run of a classical algorithm.
To get N solutions, the algorithm has to be run N times with different parameter settings.
2. Some classical algorithms are limited in handling non-convex MOOPs; as a result not all Pareto-optimal solutions are found.
3. All classical algorithms require a certain amount of problem knowledge such as suitable weights or target values.

As a result of the difficulties associated with classical methods, a large number of research efforts from the machine learning community have surfaced to deal with MOOPs, and in particular, the feature subset selection problem. We will begin our discussion by examining the two approaches to the problem of feature subset selection that have been applied by the machine learning community: (i) the filter approach, and (ii) the wrapper approach.

2.4.2 The Filter and Wrapper Approach

The filter approach is a feature selection strategy that is performed without taking into account the specific induction algorithm that will be applied to the selected features. The aim of this approach is to select a subset of features that preserves as much relevant information found in the entire set of features as possible [Freitas, 1998]. See Figure 2.4.

Figure 2.4: The Filter Approach (From [Freitas, 1998])

Unfortunately, its decision to ignore the induction algorithm in the feature selection process is its major weakness and this is aptly illustrated by [Kohavi and John, 1997]. [Kohavi and John, 1997] shows that the result of a feature subset selected by a filter algorithm known as “Relief” and based on relevance measures devised specifically for linear regression, would not be appropriate for induction algorithms such as Naïve-Bayes because in some cases the performance of Naïve-Bayes improves with the removal of relevant features [Kohavi and John, 1997]. Consequently, a feature selection scheme should take into account the biases of the induction algorithm, as is done in the wrapper approach.

The wrapper-based approach to feature selection is closely coupled to the induction algorithm that will be applied to the selected features. In this approach, the aim is to search for a candidate feature subset that is *optimal* for a given inducer. The result of this close relationship is that the predictive accuracy for a given inducer is generally higher compared to that of the filter approach.

However, this higher predictive accuracy comes at the expense of requiring more computation since the inducer executes multiple times on many different feature subsets. . Another disadvantage of the wrapper approach is the loss of generality since the feature subsets selected are optimized for one induction algorithm. However, if emphasis is placed on maximizing predictive accuracy for a given inducer, which is the case in this thesis, then the wrapper approach has admirable characteristics which suggest its use.

Two well-known search strategies used with the wrapper-based approach are forward sequential selection (FSS) and backward sequential selection (BSS) [Freitas, 1998]. FSS starts with an empty candidate feature subset and iteratively adds to the subset one feature at a time until a stopping criterion (i.e., predictive accuracy) is reached. At each iteration, the feature added to the current feature subset is the feature whose inclusion maximizes the measure of performance. BSS takes an opposite strategy to that of FSS. It starts with a candidate feature subset that is the entire set of available features and iteratively removes one feature at a time from the subset as long as some measure of performance is improved. At each iteration, the feature added to the current feature subset is the feature whose removal maximizes the measure of performance.

FSS and BSS strategies are both greedy search methods that work with one feature at a time, and therefore have the drawback of being sensitive to feature interaction. Some examples of methods adopting the FSS and BSS strategies are:

- Neural networks through pruning techniques that work by removing nodes from the network in such a way as to minimize the squared error after training for a number of epochs [Onnia et al., 2001];
- Fuzzy classification systems for pattern recognition through the generation and pruning of fuzzy if-then rules [Nürnberg et al., 2000]; and
- Decision trees that implicitly provide feature selection capability by partitioning the training data based on a single feature at each tree node, and eliminating a particular

feature from classification if it is not tested at any node of the decision tree [Cardie, 1993].

Another weakness inherent in these methods is the fact that they are only capable of finding one Pareto-optimal solution in one simulation run.

Due to the weaknesses in greedy search methods, more robust methods have been proposed by machine learning optimization researchers for MOOPs. As with most techniques designed for MOOPs, these algorithms have their history in single-objective optimization problems (SOOPs). Three popular approaches are: (i) tabu search (TS), (ii) simulated annealing (SA), and (iii) evolutionary algorithms (EAs). We will briefly examine each approach and justify our design decisions from the analysis.

2.4.3 Multi-Objective Tabu Search

Tabu search [Houck et al., 1996a] is a meta-heuristic optimization procedure that utilizes memory as a guide to repeatedly search for better solutions. At each iteration, new solutions are constructed by examining a neighbourhood and these new solutions are compared with a data structure called a tabu list which acts to prevent cycling. The best new solution which is not tabu is selected and the system moves to that new solution. This process continues until a stopping criterion is met.

The Multi-Objective Tabu Search (MOTS) [Hansen, 1996] was adapted to handle MOOPs by providing a population of solutions that are optimized towards the non-dominated front since the traditional TS is a non-population method. Since the MOTS also utilize the non-domination concept, it can handle both convex and non-convex problem since MOTS allows the Mean Square Error to increase. However, modification of the MOTS to find a Pareto-optimal front did not eliminate its inability to efficiently escape from a local optima when an MOTS individual (solution) becomes trapped. To escape a local optima requires a large number of iterations, which

is computationally intensive for a population of individuals. Another problem with the MOTS is the need to correctly set the number of current solutions to use. If too few current solutions are present, the non-dominated frontier will not be reasonably covered. On the other hand, if too many current solutions are present, the solutions will overlap resulting in unacceptable interference among solutions and an uncontrolled oscillation of the points. The oscillation of the points (i.e. solutions) is the result of the fact that the MOTS update its current solutions based on the proximity of other solutions for a given objective. That is, the closer a weight vector element of a current solution to another non-dominated solution s_i in the neighbourhood, the more the weight is increased to s_i , assuming that s_i is the best solution in the neighbourhood. The result is that each current solution converges to the centre of its own region. Though a mechanism has been proposed to dynamically determine the number of current solutions to have at any given iteration in the algorithm, it requires a user-defined constant that helps to maintain a uniform distribution of solutions along the non-domination front [Hansen, 1996]. Of course, the correct setting of this constant is difficult since it requires some knowledge of the Pareto-optimal front. Unless a correct number of current solutions can be set, the update strategy just described for a current solution may cause difficulty in dealing with a large number of feature interactions [Hansen, 1996].

2.4.4 Multi-Objective Simulated Annealing

Simulated annealing is a stochastic search method which was designed using a spin glass model by [Kirkpatrick et al., 1983]. A SA optimization starts with a Metropolis Monte Carlo simulation at a high temperature. A Metropolis Monte Carlo simulation is a sampling procedure proposed by [Metropolis et al., 1953] which incorporates a *temperature* into a system and [Kirkpatrick et al., 1983] used it to find the lowest energy (most stable) orientation of a system. [Kirkpatrick et al., 1983]’s method was based upon the procedure used to make the strongest possible glass. The

procedure heats the glass to a high temperature so that the glass is a liquid and the atoms can move relatively freely. At high temperature, this means that a relatively large percentage of the random steps that result in an increase in the energy will be accepted. After a sufficient number of Monte Carlo steps, or attempts, the temperature is decreased. The Metropolis Monte Carlo simulation is then continued. This process is repeated until the final temperature is reached.

The adaptation of SA to handle MOOPs continues to retain its well known ability to perform local fine-tuning towards the end of the search to give finer results due to the gradual temperature reducing technique [Nam and Park, 2000]. The modification of the SA algorithm to a Multi-Objective Simulated Annealing (MOSA) algorithm can also handle both convex and non-convex problems [Nam and Park, 2000]. However, the modification did not result in a population based method as was the case for the tabu search algorithm. As a result, the MOSA algorithm has to be run multiple times with different parameter settings as in the classical methods discussed earlier. Moreover, SA is well known to have a long annealing time and this is also present in MOSA. Other difficulties associated with the MOSA are its inability to effectively handle large problem size and large feature interactions [Nam and Park, 2000].

2.4.5 Multi-Objective Genetic Algorithm

Examination of the methods above and noting some of the major drawbacks suggests that there is merit in using a MOGA as a wrapper for the feature subset selection problem due to the following reasons. The first reason for using a MOGA is that prior knowledge of the problem is not required. For example, MOTS requires too much prior knowledge of the problem. A second benefit is that a MOGA can handle large problem size and large epistatic effects (i.e. a large number of feature interactions) [Freitas, 1998]. Experiments have shown MOSA to have difficulty with large problem size and epistasis. A third reason is that MOGAs do not assume monotonicity like in many non-evolutionary methods. The monotonicity assumption – adding

features to a feature subset is guaranteed not to reduce predictive accuracy – is not very realistic in the wrapper approach due to feature interactions, noisy features, etc. A major criticism of using a GA is its high time complexity. However, as will be shown in Chapter 3 and Chapter 4, the time complexity of our MOGA is not infeasible in solving real-world datasets of non-trivial size.

2.4.6 Competing Single-Objective Evolutionary Approaches

In the sections that follow, we will be examining in detail some competing evolutionary approaches pertaining to the feature subset selection problem. In this section we cover the single-objective evolutionary algorithms (SOEAs) because they are not only predecessors to the multi-objective evolutionary algorithms (MOEAs), but because novel work has been proposed that pertains specifically to the feature subset selection problem. Later sections will cover previous work done for the feature subset selection problem in the MOEAs area.

2.4.6.1 Feature Selection Via Constrained Genetic Algorithm

In their investigation of the feature selection problem, [Siedlecki and Sklansky, 1989] proposed a constrained optimization GA that wrapped around a *KNN* (*K*-Nearest Neighbours) classifier. The method’s objectives were to minimize the number of selected features while maintaining the classifier’s performance at a specified level in this study, ensuring the error rate does not exceed a given threshold served as the constraint variable. The following describes the essential aspects of [Siedlecki and Sklansky, 1989]’s GA.

A standard individual encoding for feature selection was used for an individual $a = (\alpha_1, \alpha_2, \dots, \alpha_d)$ in the GA population that consisted of a d -element binary-valued vector, where α_i represents the i -th bit (gene) and d is the number of original features. An element in the binary-valued

vector assumes a value of 0 if the i -th feature is excluded from the subset and a 1 if it is present in the subset. Associated with each individual a is a level function value $l(a)$, which is simply the total number of features selected out of the d available features.

The fitness function to maximize is given by:

$$f(a_i) = (1 + \varepsilon) \max_{a_j \in \Pi} J(a_j) - J(a_i), \quad (2.13)$$

where $\Pi = \{a_1, a_2, \dots, a_n\}$ denotes a population of feature selection vector; and ε is a small positive constant that assures that even the least fit solution is given the opportunity to reproduce (i.e., $\min f(a_i) > 0$). The term $J(a_i)$ is to be minimized and is defined as:

$$J(a) = l(a) + p(e(a)), \quad (2.14)$$

where a is a bit string representing a feature subset, and $l(a)$ is defined as above and represents approximately the cost of extracting features. The $l(a)$ plays the obvious role of favouring smaller feature subsets being selected. The $p(e(a))$ term is a penalty function that is dependent on an error rate e which is in turn dependent on a . $p(e(a))$ is defined as:

$$p(e) = \frac{\exp((e - t)/m) - 1}{\exp(1) - 1}, \quad (2.15)$$

where e is the error rate, t is the feasibility threshold which is used to measure the classifier's performance (i.e., a subset is defined as feasible if its error rate is below t), and m is a scale factor (referred to as the 'tolerance margin'). Notice several properties arising from equation (2.15):

- Solutions with error rate $e < t$ receives a negative penalty;
- Solutions with the same level function value $l(a)$ are validated according to their error rate, and the ones with the lower error rates are better adapted (adaptation is measured by equation (2.14));
- Solutions with error rate $e > t$ but $e < (t + m)$ receive a small penalty (between 0 and 1), which makes them approximately equivalent to subsets at one level higher; and

- Solutions with error rate $e > (t + m)$ receive a relatively high penalty (over 1) and cannot even compete with subsets at the next higher level.

The error rate e in equation (2.15) is defined in the k -degree algebraic form as:

$$e = e_0 + \sum_{i_1, \dots, i_k} w_{i_1, \dots, i_k} \alpha_{i_1} \dots \alpha_{i_k} + \varepsilon + \sum_{j=1} v_j \alpha_{i_1} \dots \alpha_{i_k}, \quad (2.16)$$

where:

- e_0 is a constant;
- $\dot{\alpha}_i$ denotes $1 - \alpha_i$;
- α_i denotes a binary value as stated above;
- $w_{i_1, \dots, i_k}$ are constants;
- ε is a variance on the error rate with a normal distribution $N(0, \varepsilon_0)$ to simulate training defects; and
- v_j represents the absolute value of local defects.

In the above formula, the constants $w_{i_1, \dots, i_k}$ are called penalty factors that represent the penalty for dropping the information contained in features, and could take on nonnegative or negative values. Nonnegative values arise due to Bayes theory that the error rate function should increase when features are dropped (monotonicity property). However, because data contain various anomalies that can cause the error rate function to decrease, $w_{i_1, \dots, i_k}$ can take on negative values. The variance ε used to represent the training defects covers situations when the training procedure does not produce the best classifier for the data. Finally, specific subsets of features can contain local defects that are significantly larger than those generated by an error rate model whose k -degree is equal to 2. This additional number of defects s is each represented by their absolute value v_j , and the combinations of features for which they occur.

The GA suggested by [Siedlecki and Sklansky, 1989] effectively took into consideration the fact that real-world data sets have anomalies that contradicts the monotonicity assumption by allowing negative penalty values. Moreover, training defects and local defects were factored into

the error rate function to provide a better-informed fitness function. However, the penalty function requires the setting of two additional parameters: the feasibility threshold t and the tolerance margin m . Finding the proper values for these two parameters tend to difficult because it implies that some knowledge of the problem is known *a priori*.

2.4.6.2 Commonality-Based Crossover Operator for Feature Selection

In [Siedlecki and Sklansky, 1989]’s GA, standard genetic operators proposed in [Goldberg, 1989] were used with a standard individual encoding for feature selection. [Chen et al., 1999] challenged the effectiveness of such a partnership by proposing a non-standard crossover operator for a GA that works as a wrapper around a Euclidean Decision Tables (EDT) classifier. The fitness function minimizes the number of classification errors first and then minimizes the number of selected features.

The developed Common Features/Random Sample Climbing (CF/RSC) crossover operator for feature subset selection follows the Commonality-Based Crossover Framework which defines crossover as a two-step process: 1) preserve the maximal common schema of two parents, and 2) complete the solution with a construction heuristic.

Following the standard individual encoding adopted by [Siedlecki and Sklansky, 1989], an offspring resulting from an application of the CF/RSC operator inherits all features that are indicated by common 1’s from their parents. This is in contrast to an offspring of standard crossover operators that inherits all of the common 1’s and common 0’s from their parents. The reason for the CF/RSC’s preservation of common 1’s is that the investigators believe that in feature selection, the “1”s (potentially relevant features) of an individual may be more informative than its “0”s. This is implied from the commonality hypothesis [Chen and Smith, 1999] that common 1’s should identify relevant features and the common 0’s should identify irrelevant features. However, it may be inappropriate to hypothesize that a feature is entirely

irrelevant due to the non-occurrence of a feature in both parents (indicated by 0's from both parents), since the feature can be “weakly relevant”.

To find weakly relevant features, the investigators used a heuristic to estimate which features are strongly relevant and start the search from this subset [Kohavi, 1995]. From above, strongly relevant features can be identified by using the commonality hypothesis, and the search can be (re)started from the common 1's. Thus, the Common Features partial solutions consisting of common 1's is used to initialize RSC (construction heuristic defined by the Commonality-Based Crossover Framework), which is used to probabilistically search (based on multiple bit mutations) for additional (weakly relevant) features.

In experiments using the CF/RSC, the investigators found the proposed operator to outperform RSC and CHC [Guerra-Salcedo et al., 1999] when run alone in terms of time and accuracy. The reasons CF/RSC performs better are because commonality-based restarts are near-optimal regions and smaller subsets are discovered. For the latter reason, although CF/RSC tend to do worse in training, performance was better during final testing. This is in contrast to large feature subsets that are found in CHC which may lead to over-fitting of the training data.

2.4.6.3 Feature Weighting Representation

Inspired by [Siedlecki and Sklansky, 1989]'s work, [Punch et al., 1993] proposed a similar GA that also utilized the KNN algorithm. However, the authors modified and extended the original approach in a number of important ways.

The individual representation of the GA population, like [Siedlecki and Sklansky, 1989]'s GA, used a fixed length chromosome where each gene corresponds to each available feature in the original data set. However, while [Siedlecki and Sklansky, 1989] used one bit that can take on either a value of 0 or 1 to represent the weight of each feature, [Punch et al., 1993] allocated a fixed number of Gray coded bits so that each feature's weight can be represented in a broader

scale, such as 0 to 10. Hence, each gene in [Punch et al., 1993]’s GA consists of multiple bits, and the total length of an individual is the product of the number of bits used per feature weight by the number of features.

By allowing each feature to be weighted in a scaled range, the KNN space in [Punch et al., 1993]’s implementation is in effect being warped or modified. Weights that move towards 0 indicate that their corresponding features are not important for the classification process and are essentially removed from consideration in the KNN space by condensing that feature’s KNN dimension. In contrast, a weight that moves towards the maximum weight indicates that the discrimination task is sensitive to small changes in that feature. The elongation of that feature’s KNN dimension probably reflects that an increased separation between the classes would give better discrimination capability. Note that the feature weights are rarely equal to 0; they only approach 0. Hence, if a feature weight falls below a small pre-specified threshold value then the feature weight is effectively set to 0. The following fitness function was proposed by the authors:

$$\text{Fitness} = \gamma * (\text{TotPred} - \text{CorrectPred}) / \text{TotPred} + \delta * (nmin / k) / \text{TotPred}, \quad (2.17)$$

where, *TotPred* is the total number of predictions, *CorrectPred* is the number of correct predictions, *k* is the number of nearest neighbours used for classifying a data instance, and *nmin* is the number of neighbours in the set of *k* nearest neighbours that have a minority class (i.e., the number of neighbours that were not used in the resulting classification). λ and δ are user-defined weights used to tune the processing of the algorithm. Note that the minimization of this fitness function favours candidate solutions that simultaneously minimize the number of incorrect predictions and the value *nmin*.

Running their GA on rhizosphere data sets, [Punch et al., 1993] found that their proposed algorithm produced lower classification error rates than that compared to [Siedlecki and Sklansky, 1989]’s approach. However, as in [Siedlecki and Sklansky, 1989]’s approach, [Punch et al., 1993]’s GA require setting two additional parameters: the user-defined weights λ and δ . As

with [Siedlecki and Sklansky, 1989]’s approach, finding the correct values for the user-defined weights is difficult.

2.4.6.4 Combining Feature Selection and Feature Weighting

Yet another GA that utilized the KNN algorithm was proposed by [Raymer et al., 2000] for the feature subset selection problem. However, unlike the previous methods, this approach simultaneously performs feature selection and feature weighting. The essential ideas of the algorithm are described in the following paragraphs.

The individual representation of the population consists of three substrings. The first substring is a weight vector comprising of a single real value for each feature and this is followed by the masking vector (second substring) consisting of one or more binary mask bits for each feature. A feature is considered to be omitted from classification if it is a 0, while a 1 indicated that the feature should be included under a single masking bit implementation. However, in the experiments performed by [Raymer et al., 2000], multiple bits were used to represent each feature and a feature would be included in classification only if the majority of the masking bits associated with that feature were set to one. Their motivation for using multiple bits is because single bits can cause discontinuities in the objective function, making optimization more difficult for the GA. Finally, the third substring encodes some classifier-specific information to be optimized along with the feature weights. Since the investigators used KNN classification, the substring encoding classifier-specific information specifies the value of k , where k is the number of nearest neighbours used for classification.

The objective function being minimized by the GA consisted of the following error function:

$$\begin{aligned} \text{Fitness} = & C_{pred} * (\# \text{ incorrect predictions}) + \\ & C_{mask} * (\# \text{ unmasked features}) + \\ & C_{vote} * (\# \text{ incorrect votes}) + \end{aligned}$$

$$C_{bal} * (\text{difference in accuracy between classes}), \quad (2.18)$$

where:

- *# incorrect predictions* is the number of classification errors in the training set;
- *# unmasked features* is the number of selected features;
- *# incorrect votes* is the number of incorrect votes during the KNN classification;
- *difference in accuracy between classes* is self-explanatory; and
- C_{pred} , C_{mask} , C_{vote} , and C_{bal} are user-defined weights.

In the above formulation, the prediction and voting terms are used to drive the search towards weight sets that achieve better prediction results. Moreover, the voting term is included to smooth the fitness function. Specifically, when only the number of classification of incorrect predictions was considered, the investigators found the fitness landscape to have a discrete stepwise character. The reason this occurs is because classification tends to ignore the vote of a minority of nearest neighbours, while using the class of the majority of the nearest neighbours. Therefore, including the number of incorrect votes permits the votes of all nearest neighbours to be considered, which leads to a more robust fitness function that is better aware of its local regions. The *difference in accuracy between classes* term serves to provide a balance in the number of correct predictions of all classes, avoiding maximization of the number of correct predictions in some class at the expense of reducing the number of correct predictions in other classes. Finally, the relative importance of each objective in the formula is controlled by the user-defined weights: C_{pred} , C_{mask} , C_{vote} , and C_{bal} .

Experiments on a number of different data sets by [Raymer et al., 2000] demonstrated that simultaneous optimization of weight vectors, masking vectors, and classifier-specific information outperformed the sequential floating forward selection algorithm (SFFS) of [Pudil et al., 1994] when used in conjunction with KNN. However, the proposed GA requires additional parameters C_{pred} , C_{mask} , C_{vote} , and C_{bal} to be fixed. Given that *a priori* knowledge of the underlying search

space is usually not known, setting the correct values for the user-defined weights is not a trivial task.

2.4.6.5 Feature Selection using a Feature Index-Based Individual Encoding

The standard individual encoding for feature selection as in [Siedlecki and Sklansky, 1989]’s is by far the most commonly used representation. However, in their work on feature selection using a GA as a wrapper around a C4.5 algorithm, [Cherkauer and Shavlik, 1996] proposed an alternative form of individual encoding that could offer some advantages beyond the standard individual encoding. In this study, an individual (feature subset) is represented by a fixed-length genome containing N genes, where the value of each gene may take on either a feature or be empty. The value of N is a user-specified parameter that is independent of the total number of initial features. A feature may occur multiple times and in any position. The fact that a particular feature f_i occurs twice compared to another feature f_j is irrelevant for the purpose of decoding the individual into a subset of selected features.

This unconventional individual encoding is motivated by the following two reasons: (i) the fact that multiple occurrences of the same features can exist in the genome helps to slow the loss of genetic diversity that tends to occur during genetic search and allows better features to propagate, and (ii) because the length of an individual for this encoding is independent of the number of original features, this approach is more scalable to data sets with a very large number of features.

[Cherkauer and Shavlik, 1996]’s GA utilized a variant of the standard crossover and mutation operators in addition to a new genetic operator called *Delete Feature* to support the proposed unconventional individual encoding. The *Delete Feature* accepts as input one parent and produces as output one offspring that is identical except that all occurrences of one (equally

probable chosen) feature is removed from the child. Hence, this operator directly biases the selection of smaller feature subsets.

Although the proposed individual encoding is unconventional, nevertheless, it provides a direct mechanism for controlling the comprehensibility of the classification rule produced by the algorithm through the parameter N . Through N , the user can tune a solution's comprehensibility with respect to predictive accuracy.

2.4.6.6 Incorporating the Baldwin Effect for Feature Selection

All of the proposed techniques for feature selection discussed thus far have been considered to be predicated on a wrapper approach even though they include a certain flavour of the filter approach from using the number of selected features as a metric. As mentioned earlier, the wrapper approach should produce better classification performance than the filter approach, but adds considerable overhead. [Bala et al., 1997] argues that this overhead is usually avoided by restricting the number of alternative feature subsets to be evaluated, but this may also produce suboptimal results. To address this issue, [Bala et al., 1997] recommended a hybrid “filter + wrapper” approach that uses a GA to search the space of possible feature subsets, and a fitness function that evaluates the candidate solutions based on performance, information content, and penalty. In particular, the fitness function to maximize takes the form:

$$\text{Fitness}(x) = \text{Info}(x) - \text{Penalty}(x) + \text{Accuracy}(x), \quad (2.19)$$

where x is the candidate feature subset that uses a fixed, standard individual encoding as in [Siedlecki and Sklansky, 1989]; $\text{Accuracy}(x)$ is the performance evaluation in terms of classification accuracy rate that is measured on unseen test data when applied to C4.5 based on x ; $\text{Penalty}(x)$ is the number of features in x ; and $\text{Info}(x)$ is an information theoretic measure which is an estimate of the discriminatory power of the features in x . Note that $\text{Info}(x)$ and $\text{Penalty}(x)$ are filter-oriented metrics since they are independent of the classification algorithm used. The

investigators found that using all of the above metrics yielded the best performance in terms of error rates and the number of features used by C4.5.

[Bala et al., 1997] also investigated the Baldwin Effect in their implementation. They discovered that by using the entropy measure as an information measure during tree derivation, the actual number of features in the classifier were generally significantly less than that specified by the genome. Three effects were observed by the investigators as a result of the Baldwin Effect: (i) the effect of learning significantly accelerated convergence to feature sets displaying higher fitness, (ii) the features actually used by the high performance classifiers are those which dominate the genomes in later generations even though no back-coding of this information occurred, and (iii) observation of a decrease in the number of features not used by C4.5 corresponded with a decrease in the size of the feature subsets being evaluated by the GA. However, a problem associated with the use of Baldwinian learning is that it requires solving multiple mapping issues [Houck et al., 1996b].

2.4.6.7 Incorporating the Lamarckian for Feature Selection

Unlike the Baldwinian learning strategy adopted by [Bala et al., 1997], [Oh et al., 2002] proposed a wrapper-based GA that incorporates a Lamarckian learning strategy that finds a feature subset size that satisfies the constraint d . A candidate solution of size n that is found to violate the constraint (i.e., $n > d$) is given a penalty, *penalty*. In particular, the fitness of a chromosome C is based on a standard individual encoding as in [Siedlecki and Sklansky, 1989]'s and the corresponding feature subset F_C is defined by:

$$Fitness(C) = Acc(F_C) - penalty(F_C), \quad (2.20)$$

where $Acc(F_C)$ is the accuracy rate returned by a classifier algorithm (in this case, a 1-NN); and the penalty term is given by:

$$penalty(F_C) = w * (|F_C| - d), \quad (2.21)$$

where w is a weight.

A distinguishing characteristic of [Oh et al., 2002]’s proposal is the local search (improvement) method that is applied on the offspring (feature subset) before including it in the population. The local improvement method is accomplished by first putting all 1’s in C into X , and putting all 0’s in C into Y . Thereafter, one of two local search operations is executed depending the size of X . The first local search operation is *ripple_rem(r)* that firstly removes the least significant features r times from X to Y , and then adds to X the most significant features $r-1$ times selected from Y ; the effect is removing one feature from X . Meanwhile, *ripple_add(r)* adds to X the most significant features r times selected from Y , and removes the least significant features $r-1$ times from X to Y ; the effect is an increase in one feature in C . The investigators call the parameter $r \geq 1$ the ripple factor. The ripple factor influences directly the actual number of removal and addition operations, which in turn influences the strength of the local improvement. A larger ripple factor results in a stronger local improvement.

The local improvement procedure has two desirable effects. The first effect is that the final performance is improved through local improvement of chromosomes. The second one is related to controlling the subset size. By applying a sequence of a proper number of removal and addition operations, the feature subset size can be made to satisfy the constraint d . However, a problem associated with Lamarckian evolution is that changes in the genetic material of the chromosomes may result in the loss of some potentially good inherited schemas [Houck et al., 1996b].

2.4.6.8 Feature Selection Via Search for Partially Defined Instances

All the algorithms discussed thus far applied all data instances with the same selected features to a given classifier during an evaluation of a single candidate solution. [Llora and Garrel, 2001] suggested a different approach where different data instances end up containing different selected

attributes during classification for a given candidate solution. In essence, each individual of a GA represents a set of partially-defined instances. Instances are considered partially-defined if at least one of their predictor features has a known value. An original, fully-defined instance can be transformed into a partially-defined instance by simply discarding some of the features. [Llora and Garrel, 2001] used this approach to select partially-defined instances for the nearest neighbour algorithm.

2.4.6.9 Joint Feature Selection and Instance Selection

Another novel approach was introduced by [Ishibuchi and Nakashima, 2000] that not only selected a set of features to be applied to a data mining task, but also a set of data instances, out of all original features and data instances. The obvious benefit of selecting a subset of data instances is the reduction of the number of data instances that have to be considered during execution of the data mining task. However, the major weakness inherent in this strategy is that it is not scalable to large datasets.

In [Ishibuchi and Nakashima, 2000], the GA is used as a wrapper around a nearest neighbour algorithm. The main difference between this algorithm and the previously described algorithms is that the GA individual in this implementation consists of selected features and selected instances, rather than just selected features. Thus, both features and data instances selected by a candidate solution are given as input to a classifier, and the performance of that algorithm based on a fitness function is given back to the GA.

A candidate solution in [Ishibuchi and Nakashima, 2000] is encoded using an extension of the method proposed by [Siedlecki and Sklansky, 1989]. More precisely, an individual consists of a binary string of length $m + n$, where m is the number of original features and n is the number of original data instances. Selected features and selected data instances are marked by bits set to 1.

Though this individual encoding scheme is simple to implement, it is not scalable to large data sets.

[Ishibuchi and Nakashima, 2000] also proposed a biased mutation operator that is applied to the binary substring encoding the selected data instances. This mutation operator favours fewer data instances being selected by having a higher probability of mutating a 1 to a 0 than from a 0 to a 1. Though the investigators only applied this mutation operator to the binary substring encoding the selected data instances, it could just as easily be applied to the binary substring encoding the selected features.

The fitness function proposed by [Ishibuchi and Nakashima, 2000] favours individuals that have high predictive accuracy, and few selected features and data instances. These criteria are combined into a weighted sum as follows:

$$\text{Fitness} = w_{acc} * Acc - w_{feature} * |F| - w_{instance} * |I|, \quad (2.22)$$

where Acc is the number of correctly classified data instances of the classification algorithm, run with only the selected features and data instances associated with the candidate solution; $|F|$ is the number of features selected by the candidate solution; $|I|$ is the number of data instances selected by the candidate solution; and w_{acc} , $w_{feature}$, and $w_{instance}$ are user-defined weights.

2.4.7 Competing Multi-Objective Evolutionary Approaches

In this section, we continue our discussion of competing evolutionary methods by examining approaches that actually cast the feature subset selection problem into a MOOP.

One such approach is an elitist multi-objective GA for feature selection that was proposed by [Pappa et al., 2002], and worked as a wrapper around the well-known tree classifier, C4.5. The proposal's basic ideas are described in the following paragraphs.

An individual of the GA population is represented using the standard encoding proposed by [Siedlecki and Sklansky, 1989]. Evaluation of a candidate solution consists of minimizing two

objectives: the error rate of C4.5 and the size of the decision tree built by C4.5. Elitism is enforced by allowing non-dominated candidate solutions with respect to the two objectives to survive unaltered to the next generation. Letting N and N_{elit} be the total population size and the number of individuals of reproduced by elitism, respectively, the remaining $N - N_{elit}$ individuals to reproduce are chosen by performing a tournament selection procedure $N - N_{elit}$ times, as follows. First, k individuals are randomly selected from the current population, where k is the tournament size. In the experiments performed by [Pappa et al., 2002], k was set at 2.

After selecting two individuals, the GA determines which one is the winner based on the concept of Pareto dominance, taking into account the two objectives to be minimized. Two possible situations arise in a tournament involving individuals I_1 and I_2 . The first one is that one of the individual dominates the other and that individual is selected as the winner of the tournament.

The second situation is that none of the individuals dominates the other. In this case, a tie-breaking criterion was used that requires selecting a winner based on the two objectives while simultaneously avoiding emphasis on any of the objectives. [Pappa et al., 2002] proposed the following tie-breaking criterion. For each individual I_i , $i = 1, 2$, the GA computes the number of individuals in the current population that are dominated by I_i , call it X_i , and the number of individuals in the current population that dominates I_i , call it Y_i . Thereafter, the winner of the tournament is the individual with the largest $X_i - Y_i$ value. In cases where both individuals have the same $X_i - Y_i$, the winner is randomly chosen.

The advantage of [Pappa et al., 2002]’s GA is that no explicit fitness assignment procedure is required. However, the implementation does not have an explicit niching mechanism for selecting an individual in the tournament in a non-dominance situation. In the later stages of the simulation, most solutions tend to be non-dominating and using a random selection scheme may result in a set of Pareto-optimal solutions that is not well-distributed. Moreover, lateral diversity is not emphasized. In [Pappa et al., 2002]’s GA, lateral diversity is quickly eliminated by the

elitism mechanism and by the tournament selection procedure. The result would be that the search for a diversified set of Pareto-optimal solutions would take much longer [Deb and Goel, 2001].

2.4.7.1 Non-Dominated Sorting Genetic Algorithm for Feature Selection

[Oliveira et al., 2003] also proposed an MOEA for the feature selection problem based on a wrapper approach. However, to find the Pareto-optimal front, they used [Srinivas and Deb, 1994]'s Non-Dominated Sorting Genetic Algorithm that works as a wrapper around a back propagation neural network that is used as the classifier. Like [Pappa et al., 2002's] MOEA, the candidate solution is encoded using the standard individual encoding as in [Siedlecki and Sklansky, 1989]. Moreover, the fitness function minimizes two objectives: the number of features and the error rate of the classifier. The number of features in a candidate solution is simple to calculate since it only requires accumulating the number of 1's in the portion of the chromosome that encodes for corresponding features in the data set. However, computing the error rate from the classifier is not trivial. In the context of the wrapper approach, in each generation, each chromosome (a feature subset) has to be evaluated by training the corresponding neural network and computing its accuracy. For a large training set, the required learning time is not feasible. Consequently, to reduce the huge learning time, [Oliveira et al., 2003] adopted the strategy proposed by [Moody and Utans, 1991], who uses the sensitivity of the network to estimate the relationship between the input features and the network performance.

The sensitivity of the network model to variable i is defined as:

$$S_i = \frac{1}{N} \sum_j S_{ij} \quad (2.23)$$

$$S_{ij} = SE(\bar{x}_i) - SE(x_{ij}) \quad (2.24)$$

$$\bar{x}_i = \frac{1}{N} \sum_{j=1}^N x_{ij} \quad (2.25)$$

S_i measures the effect on the training squared error (SE) of replacing the i th input x_i by the its average $\bar{x}_i$ for all N exemplars. For variables with small sensitivity values (S_i) with respect to the network outputs, [Moody and Utans, 1991] showed that removing these variables does not influence the final classification. Consequently, evaluating a given feature subset only requires the replacement of all unselected inputs x_i of the test set by their averages $\bar{x}_i$ computed by the training set. The result is that training the neural network is avoided and the wrapper approach becomes a feasible alternative.

Though [Oliveira et al., 2003] claim that by adopting [Moody and Utans, 1991]'s strategy will decrease the require number of computations, the savings may be minimal due to the fact that the neural network has to be retrained by the remaining input variables after the input variable with the lowest sensitivity value is removed. Moreover, sensitivity based pruning neglects possible correlations between inputs. An improvement of the method would be to first orthogonalize the inputs and perform the sensitivity analysis using the transformed inputs. Another drawback with respect to [Oliveira et al., 2003]'s implementation is associated with NSGA. NSGA require the sharing parameter σ_{share} to be set, which has a large effect on the set of Pareto-optimal solutions returned by the system. If σ_{share} is too small, the GA tends to form more niches than is actually present in the search. On the other hand, if σ_{share} is too large, not all optima may be found. Albeit some theoretical work has been done [Deb, 2001] to guide the selection of values for σ_{share} , choosing a value is still largely by trial and error.

2.4.7.2 MOEA and Constructive Neural Network for Feature Selection

A similar strategy to [Oliveira et al., 2003] was proposed by [Yang and Honavar, 1998] in their investigation into the feature subset selection problem. However, in [Yang and Honavar, 1998]'s work, emphasis was placed not so much on the MOEA used, but on the constructive neural network learning algorithms used to eliminate the need for ad hoc network architectures. Specifically, a constructive neural network known as DistAl [Yang et al., 1997] was used to build the neural network architecture. DistAl works by adding hidden neurons one at a time based on a greedy strategy which ensures that the hidden neuron correctly classifies a maximal subset of training patterns belonging to a single class. Examples that are correctly classified are eliminated from consideration. This process continues until the training set is empty, that is, when the network correctly classifies the entire training set at which time the training set is said to be linearly separable in the transformed space defined by the hidden neurons. Since DistAl is based on a greedy strategy, it is sensitive to input correlations as in [Oliveira et al., 2003].

2.4.7.3 Elitist Niche Pareto Genetic Algorithm for Feature Selection

[Emmanouilidis et al., 1999] and [Emmanouilidis et al., 2000] also introduced MOEAs to tackle the feature selection problem for the supervised learning task. Both [Emmanouilidis et al., 1999] and [Emmanouilidis et al., 2000] proposed a modified elitist Niche Pareto Genetic Algorithm (NPGA) that worked as a wrapper around a classifier. While [Emmanouilidis et al., 1999] used the GA as a wrapper around a neurofuzzy classifier, [Emmanouilidis et al., 2000] used a neural network. Since both works are not significantly different, only [Emmanouilidis et al., 2000] will be addressed. The main characteristics of the algorithms are as follows.

Based on the Niche Genetic Algorithm (NPGA) proposed by [Horn et al., 1994], this approach searches for a set of Pareto-optimal solutions by minimizing two objectives: (i) the number of features selected, and (ii) the estimated misclassification rate returned by the neural network. As with most GA based feature selection algorithms, candidate solutions are represented using [Siedlecki and Sklansky, 1989]'s standard individual encoding scheme. However, a number of modifications were made [Horn et al., 1994]'s original NPGA to improve its convergent property. The first modification made was the increase of the offspring population size to twice the size of the parent population. By doubling the offspring population, every individual in the parent population gets an opportunity to mate, which in turn helps to increase the diversity along the Pareto front. [Emmanouilidis et al., 2000] also introduced elitism to their NPGA that automatically includes non-dominated solutions in the next generation parent population while the rest of the individuals for the parent population are selected by a random sampling tournament selection. In this implementation, three individuals compete in each tournament and the tournament sampling set size is based on a heuristic that is set to a tenth of the parent population size. In some cases, tournament selection may result in a tie (i.e. none of the individuals in tournament sampling set dominates any other competitors). Here, fitness sharing is employed in which the individual with the smallest niche count is selected as the winner. [Emmanouilidis et al., 2000] calculated the niche count for each individual based on the Hamming space and the objective space.

Notice that this approach of selecting individuals for the next generation parent population is only viable if the number of elite solutions does not exceed the fixed parent population size. In the case when the number of elite solutions exceed that of the required number of the parent population size, [Emmanouilidis et al., 2000] utilized the clustering algorithm proposed by [Zitzler and Thiele, 1998] to select the least crowded individuals in the population to be included in the next generation parent population.

In addition to introducing a modified NPGA, [Emmanouilidis et al., 2000] proposed a novel genetic operator for the feature selection problem called the Subset Size-Oriented Common Features crossover operator (SSOCF). In the context of feature selection, a standard crossover operator applied on two feature subset of size m and n tends to yield an offspring of complexity approximately $(m + n) / 2$. The implication of this is that the EA tends to explore mostly medium-sized subsets while the edges of the non-dominated front are less well explored. SSOCF endeavours to circumvent this by ensuring that each offspring retains the feature subset size close to that of each mating parent. Specifically, each offspring preserves the common features of both parents. Non-shared features of the i -th parent are inherited by the offspring with probability $(n_i - n_c) / n_u$, where n_i is the subset size of the i -th parent, n_c is the number of commonly selected features across both mating partners, and n_u is the number of non-shared selected features. Those non-shared features which are not inherited by the first offspring are inherited by the second offspring.

As with [Pappa et al., 2002]’s GA, [Emmanouilidis et al., 2000]’s modified NPGA does not explicitly maintain lateral diversity. Moreover, two additional parameters require setting: the sharing parameter σ_{share} and the size of the tournament sampling set. The importance of setting a correct value for σ_{share} has already been discussed above. The GA is also very sensitive to the chosen size of the tournament sampling set. If it is too small, the non-domination check would be noisy, which may not adequately emphasize the true non-dominated solutions in the population. On the other hand, if the chosen size is too large, the true non-dominated solutions would be emphasized, but the computational complexity would be greater. Despite the above shortcomings of [Emmanouilidis et al., 2000]’s NPGA, the proposed SSOCF has some theoretical set properties that warrant further investigation.

2.4.7.4 Feature Selection Via MOGA for Clustering

Unlike previously discussed algorithms, this next evolutionary algorithm proposed by [Morita et al., 2003] for the feature selection problem concentrates on unsupervised learning (clustering task). The objective in unsupervised feature selection is to search for a subset of features that best uncovers “natural” groupings (clusters) from data according to some criterion. In [Morita et al., 2003]’s approach, a standard K -means clustering algorithm [k-means clustering, 2004] is used for the purposes of clustering. As with most evolutionary algorithms used for the feature subset selection problem, this method utilizes the wrapper approach. The specific MOGA used in this technique is the NSGA proposed by [Srinivas and Deb, 1994] and attempts to minimize two objectives: (i) the number of selected features, and (ii) the Davies-Bouldin (DB) index Davies et al. [Davies and Bouldin, 1979]. The DB index is a validity index that measures the quality of clusters by combining intra-cluster scatter and inter-cluster separation metrics. However, the DB index in its original form as proposed by [Davies and Bouldin, 1979] is not suitable for the task of feature selection since it is predicated on geometric distance metrics and is therefore subject to be biased by the dimensionality of the space, which is variable in such problems. That is, the higher the number of features selected, the higher the DB index. To circumvent this problem, [Morita et al., 2003] normalized the DB index by the number of selected features.

The DB index is a function of the ratio of the sum of intra-cluster scatter to inter-cluster separation, and is defined as:

$$I_{DB} = \frac{1}{D} \frac{1}{K} \sum_{i=1}^K R_i, \quad (2.26)$$

where K is the number of selected clusters, D is the number of selected features, and R_i is defined as:

$$R_i = \max_{j, j \neq i} \left\{ \frac{S_i + S_j}{d_{ij}} \right\}, \quad (2.27)$$

The denominator d_{ij} corresponds to the Euclidean distance between the centroids Z_i and Z_j of the clusters C_i and C_j , respectively, and is defined as:

$$d_{ij} = \{\|Z_i - Z_j\|\}, \quad (2.28)$$

The numerator S_i is the intra-cluster scatter of the i -th cluster and is computed as:

$$S_i = \frac{1}{|C_i|} \sum_{X \in C_i} \{\|X - Z_i\|\}. \quad (2.29)$$

The individual representation used for the candidate solutions of the population consists of two parts: the first part encodes the selected features and the second part encodes the number of selected clusters. The selected features are encoded following [Siedlecki and Sklansky, 1989]’s method of encoding selected features. The number of selected clusters is represented by the number of bits set to 1 in the second part of the chromosome. Note that the DB index is computed only after performing clustering using a standard K -means algorithm which is based on the number of selected clusters represented by the second part of the chromosome.

As in [Oliveira et al., 2003]’s implementation, [Morita et al., 2003]’s proposal requires setting the sharing parameter σ_{share} . Moreover, use of the standard K -means algorithm for the clustering task does not guarantee that the clusters found will be optimal since K -means is sensitive to centroid initialization. Therefore, the DB index calculated may not reflect the true quality of the candidate solution. However, this problem can be remedied by using a genetic-based clustering algorithm instead of K -means.

2.5 Conclusion

In this chapter we discussed the appropriateness of transforming the feature subset selection problem into a multi-objective optimization problem. Furthermore, we introduced our approach for tackling the feature subset selection problem by proposing a wrapper approach that uses a multi-objective genetic as the wrapper around a back-propagation neural network. The discussion

of our approach was facilitated by examining in detail the previous research efforts our approach built upon and also introduced our *least-crowded tournament selection operator* and our *least-crowding selection algorithm* for selecting the least crowded solution in a search space. We also discussed other methods such as classical mathematical programming techniques, greedy methods (i.e. neural networks and decision trees), multi-objective tabu search, and multi-objective simulated annealing to put into context our reasons for choosing a wrapper approach and a multi-objective genetic algorithm for our implementation. Finally, we presented in detail a number of competing evolutionary methods applied to the feature subset selection problem.

In the next chapter we continue to look at the feature subset selection problem and provide a more precise of what we mean when we say a feature subset is *optimal* or *relevant*. Moreover, we provide a more detailed analysis and comparison of our approach to that of the back-propagation neural network for feature detection and the single-objective genetic algorithm for feature subset selection.

Chapter 3

Method Analysis and Comparison

3.1 Introduction

In chapters 1 and 2, we made reference to the set of Pareto-optimal solutions of a learning algorithm for the feature subset selection problem as being *optimal* or *relevant*. In this chapter, we clarify this qualification. Moreover, we provide an analysis of the three competing algorithms performed in our experiments: (i) the back-propagation (BP) algorithm run alone, (ii) the single-objective genetic algorithm (SOGA), and (iii) our multi-objective genetic algorithm (MOGA). Our motivation for comparing (i), (ii), and (iii) comes from the fact that the BP algorithm is capable of *feature detection*, and algorithms (i) and (ii) are capable of feature subset selection. Our analysis of the algorithms will show that our MOGA is the only one that can simultaneously fulfill the three goals of maximizing accuracy, minimizing model complexity (i.e. smaller feature subset), and maximizing selection for the user (i.e. a set of Pareto-optimal solutions that the user can select from based on his/her criteria) in one simulation run.

This chapter is divided into a few sections. In Section 3.2 we provide a definition of what *relevant* and *optimal* means with respect to feature selection and present the problem facing us. In Section 3.3 we analyze and compare the BP algorithm for feature detection, the SOGA for

feature subset selection, and our MOGA for feature subset selection. We show that our approach is feasible in handling the problem before us. In particular, we focus on the computational complexity of the three competing approaches because one of the drawbacks in using the GA is the high computational requirement and the amount of time it takes to find an “optimal” solution (or a set of Pareto-optimal solutions) should still be reasonable. We will also examine the mechanisms used to maintain population diversity and other issues related to optimization in a GA framework. Section 3.4 will give a brief summary of the chapter and give a short outline of the research focus in the following chapter.

3.2 Problem Statement

Practical machine learning algorithms such as the decision tree, the instanced-based algorithm, and neural networks are known to degrade in performance (prediction accuracy) when confronted with many features that are unnecessary for predicting the desired output [Kohavi and John, 1997]. This implies that the goal of the feature subset selection problem is to find a subset from the entire set of features of a dataset such that an induction algorithm that is applied to the data with only these features generates a classifier with the highest possible accuracy. In the literature, a distinction is made between whether a selected feature in the subset is optimal, relevant, or irrelevant [Kohavi and John, 1997]. In [Kohavi and John, 1997], an optimal subset is defined in terms of an optimal Bayes classifier as follows:

Definition 3.1 (*optimal subset*). Given an induction algorithm I , and a dataset D with features $X_1, X_2, \dots, X_n$, from an unknown underlying distribution γ over the labeled instance space, an *optimal feature subset*, X_O , is a subset of features such that the accuracy of the induced classifier $C = I(D)$ is maximal.

Relevant features are also defined in terms of an optimal Bayes classifier, and are separated into two categories of strong relevance or weak relevance. Before we give [Kohavi and John, 1997]’s definition of strong relevance and weak relevance, we require the following notation. Let $S_i = \{$

$X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_m\}$, the set of all features except X_i . Also, let s_i be a value assigned to the set S_i . We are now in a position to define whether a feature is considered strongly relevant, weakly relevant, or irrelevant [Kohavi and John, 1997]:

Definition 3.2 (strong relevance). A feature X_i is *strongly relevant* if and only if there exists some x_i, y , and s_i with $p(X_i = x_i, S_i = s_i) > 0$ such that

$$p(Y = y \mid X_i = x_i, S_i = s_i) \neq p(Y = y, S_i = s_i).$$

Definition 3.3 (weak relevance). A feature X_i is *weakly relevant* if and only if it is not strongly relevant, and there exists a subset of features S'_i of S_i for which there exists some x_i, y , and s'_i with $p(X_i = x_i, S'_i = s'_i) > 0$ such that

$$p(Y = y \mid X_i = x_i, S'_i = s'_i) \neq p(Y = y, S'_i = s'_i).$$

Definition 3.4 (irrelevance). Irrelevant features are neither, strongly relevant, or weakly relevant.

Though the above definitions permit greater comprehensibility of the issue of the feature subset selection problem, for our purposes the following definition will be used:

Definition 3.5 (approximately optimal subset). Given an induction algorithm I , and a dataset D with features $X_1, X_2, \dots, X_n$, from an unknown underlying distribution γ over the labeled instance space, an *approximately optimal feature subset*, X_{AO} , is a subset of features such that the accuracy of the induced classifier $C = I(D)$ is relatively close to the local maximal and may contain optimal, relevant, and/or irrelevant features.

Though Definition 3.5 is still defined in terms of the optimal Bayes classifier, it is modified to reflect the practical scenarios that the selected feature subset is only approximately optimal (i.e. feasibly suboptimal) that maximizes the prediction accuracy for the specified learning algorithm. This is a consequence of two problems. The first problem arises in practical learning scenarios wherein the underlying distributions are not known to the learning algorithms. This problem is closely related to the bias-variance dilemma in which bias reduction must be compromised to reduce variance for a training sample of fixed size [Haykin, 1999]. The second problem comes from the fact that a given learning algorithm is searching for a hypothesis that is known to be NP-hard. That is, the size of the search space of the feature subset selection problem is 2^M , and increases exponentially with the number of features, where M is the number of features. Note

also that our definition only promises to find a solution with an accuracy rate that is only relatively close to the local maximal. In a multi-modal search space it is not guaranteed that a global optima will be obtained for a given induction algorithm due to the unknown underlying distribution over the labeled instance space; however, discovery of an approximately optimal feature subset by a given learning algorithm will imply that at the very least a solution relatively close to the locally maximal accuracy rate in a given region can be obtained since our approach does not utilize any hill-climbing techniques.

Definition 3.5 also indicates that an approximately optimal feature subset may contain optimal, relevant, and/or irrelevant features. This result comes from the fact that practical induction algorithms that generate classifiers may benefit from omitting certain features, including strongly relevant features [Kohavi and John, 1997]. Moreover, relevance of a feature does not imply that it should be in the optimal feature subset, and that irrelevance does not imply that it should not be in the optimal feature subset [Kohavi and John, 1997]. However, in practical learning scenarios irrelevant features are generally removed.

With Definition 3.5, we can now define the feature subset selection problem more succinctly. The feature subset selection problem consists of finding an *approximately optimal feature subset* from the entire set of features of a dataset such that an induction algorithm that is applied to the data with only these features generates a classifier with the highest possible accuracy. Or in our case, the feature subset selection problem consists of finding a Pareto-optimal front that is *approximately optimal* and consists of Pareto-optimal feature subsets that are *approximately optimal* such that an induction algorithm that is applied to the data with only these features generates a classifier with the highest possible accuracy.

3.3 Algorithm Analysis and Comparison

The previous section examined the characteristics of the feature subset selection problem and the large space that has to be searched for an approximately optimal feature subset. From hereon, when we speak of *optimal*, we mean a local optimum as stated in Definition 3.5 unless stated otherwise. In any data mining endeavour, the ability to achieve a given task with a simpler less complex method is preferred if performance is not compromised. For the feature subset selection problem, there is always the argument that the back-propagation algorithm alone can perform feature detection without the need for additional overhead such as the genetic algorithm. Moreover, even if a genetic algorithm is to be used, there should be a valid reason for the extra complexity associated with the use of a multi-objective evolutionary approach. In this section we analyze the back-propagation algorithm, the single-objective genetic algorithm, and our multi-objective genetic algorithm for the feature subset selection problem in the context of the three goals of maximizing accuracy, minimizing model complexity, and providing greater selection for the user.

3.3.1 The Back-propagation Algorithm

The back-propagation algorithm is well known for its ability to classify non-linear separable datasets due to its network structure that includes one or more layer of hidden neurons that performs the necessary non-linear transformations. These non-linear transformations performed by the hidden neurons for classification implicitly act as feature detectors. That is, as the simulation progresses, the salient features that characterize the training data are gradually “discovered” by the hidden neurons [Haykin, 1999]. This is essentially accomplished by subjecting the input data to non-linear transformations that map it into a new space called the

hidden space, or *feature space*. In this new space, the classes in a pattern classification task may be more easily separated from each other than in the original input space [Haykin, 1999].

The detection of features by the BP algorithm through mappings from an input space to a feature space implies that the BP needs to somehow represent every part of its input space in order to know how that part of the space should be correctly mapped [Neural Network FAQ, 2004]. Covering the input space requires training data, and the amount of training data required is proportional to the hypervolume of the input space. In “high dimensions” and for a given feature space that is represented by an arbitrarily complex function, the BP would need dense sample (data) points to learn it well. In particular, the sampling density is proportional to $N^{1/m}$, where N denotes the number of training exemplars and m denotes the dimension of the input vector [Haykin, 1999]. For example, if $N_1 = 100$ represents a dense sample for a single input problem, then $N_{10} = 100^{10}$ is the sample size required for the same sampling density with 10 inputs. Unfortunately, dense samples are difficult to find in high dimensions and the result is a phenomenon known as the curse of dimensionality. That is, the exponential growth in complexity owing to the high dimensionality causes uniformly randomly distributed points in higher-dimension spaces to lose its space-filling properties, thus making it more difficult to discern complexities in the search space [Haykin, 1999]. In the context of the feature selection problem, the curse of dimensionality may cause networks with lots of irrelevant inputs to behave relatively badly since the network may use almost all of its weights to represent irrelevant portions of the feature space. Even if the network algorithm is able to focus on important portions of the input space, the higher the dimensionality of the input space, the more data may be needed to find out what is important and what is not.

One way to overcome the curse of dimensionality is to incorporate *a priori* information about the function representing the feature space that is beyond the knowledge given by the training data. Another way to overcome the curse of dimensionality is through feature selection of the input data. Assuming that the dataset in question is not under sampled and that many attributes

exist (many of which are irrelevant), the BP algorithm's use of all the input attributes implies that it may have trouble handling the curse of dimensionality problem as mentioned in the above paragraph. Moreover, its use of all the input attributes in the dataset also means that it fails to satisfy our goals of minimizing model complexity and providing greater solution selection to the user. However, the use of the BP algorithm for feature detection alone has its advantages in that fewer computations are required in comparison to when an additional optimization method such as the single-objective genetic algorithm is used. In particular, the BP algorithm has a computational complexity that is linear with respect to W , where W is the total number of synaptic weights (including the biases) [Haykin, 1999]. This can readily be verified by examining the computations required in performing the forward and backward passes of the BP algorithm. In the forward pass, the only computations involving the synaptic weights are those that pertain to the activation potentials of the various neurons in the network; and these computations are all linear [Haykin, 1999]. In the backward pass, the only computations involving the synaptic weights are those that pertain to (i) the local gradients of the hidden nodes, and (ii) the calculations required to update the synaptic weights itself. These calculations are also linear [Haykin, 1999]. Consequently, the computational complexity of the BP algorithm is $O(W)$.

3.3.2 The Single-Objective Genetic Algorithm

The Simple Genetic Algorithm (SGA) found in the literature today comes from the original work of John Holland back in the mid-1970s. Most single-objective optimization problems (SOOP) associated with the genetic algorithm invariably refer to the SGA or a variant of it. Though commended for its non-greedy approach, one of the known criticisms of the genetic algorithm, including the SGA, is the lengthy simulation run time required since many generations must be executed to converge to an *optimal* solution. This criticism is not lost with respect to the feature subset selection problem because in our case the BP algorithm is the fitness function for the SGA

and is therefore executed many times (from hereon we refer to this approach to feature subset selection as the *SGA-BP-Wrapper* approach). In particular, the *SGA-BP-Wrapper* approach has a time complexity that is directly proportional to the number of generations, the size of the population, and the time complexity of the evaluation algorithm (in our case, it is the BP algorithm). Since the time complexity of the genetic operators is not significantly high, its exclusion from the total time complexity of the *SGA-BP-Wrapper* approach does not greatly change the final figure. Hence, the result is a time complexity of $O(mNW)$, where m is the number of generations executed, N is the population size, and W is the time complexity of the BP algorithm. Even though combining the wrapper approach, the SGA, and the BP for the feature subset selection problem is computationally intensive, it does offer a number of admirable advantages that merit its use.

One advantage is that the wrapper approach tends to give a higher predictive accuracy rate for a given induction algorithm as compared to a method such as the filter approach [Kohavi and John, 1997]. A second advantage is that the genetic algorithm is a global search technique that can handle a large number of feature interactions well due to its population-base approach and its use of recombination operators that permits it to search a large portion of the search space. As noted earlier, the wrapper approach used with greedy search strategies such as the forward sequential selection and backward sequential selection are still limited in dealing with many feature interactions.

A third advantage in combining the wrapper approach with a GA has to do with the fact that a population of solutions (individuals) is maintained. Recall that the BP is known to get trapped in a local optima and that to get a solution that is reasonably close to the global optima requires repeating the simulation. With a population of solutions, even if some solutions converge prematurely, there is still a high probability that its schemas will survive to the next generation because other solutions relatively similar to it may happen to converge to their “true” *optimal* value. That is, if the best individual in a cluster of individuals gets trapped in a local optimum,

other individuals with schemas that are relatively similar to the best individual may have found *optimal* values that truly reflect their input values. This is especially true during the later stages of the simulation wherein the population contains clusters of good solutions near many optima and competitions among clusters of different optima drives the population to converge to a single optimum. These clusters of good solutions increase the probability that the schema of a good solution trapped in a local optima will not be lost. Moreover, even if good schemas are lost because there are none or few similar solutions in the population to provide the redundancy (this can happen during the early stages of the simulation wherein the solutions are more randomly distributed in the search space), the SGA can still re-discover it through the use of the mutation operator. However, because the mutation operator has a constructive as well as a destructive effect, this may cause the GA simulation to oscillate and will require additional computations before it converges.

A final advantage for using the GA as a wrapper is that it can help overcome the curse of dimensionality. As mentioned previously, the curse of dimensionality requires *dense* sample points in order to effectively learn the mapping of an arbitrarily complex function. Since the amount of training data required in higher dimensionality is prohibitively difficult to obtain, the ability of the SOGA to converge to an acceptable *optimal* solution by using the BP algorithm to evaluate subsets of the entire feature set helps to overcome this problem. However, the ability of the GA to help overcome the curse of dimensionality assumes that the dataset we are dealing with is not under sampled. That is, if our dataset is under sampled, the model constructed by the backpropagation algorithm will be misleading since any number of models can be incorrectly fitted to the data sample due to the high degree of freedom relative to the size of the sample.

3.3.3 The Multi-Objective Genetic Algorithm

Although using the SGA as a wrapper can satisfy the maximization of predictive accuracy, it still does not explicitly and simultaneously satisfy our other two goals of minimizing model complexity and maximizing selection for the user. To fulfill all three goals, we introduce a multi-objective genetic algorithm that acts as a wrapper around the BP algorithm. That is, our MOGA attempts to find a set of Pareto-optimal solutions as defined in Definition 3.5, taking all objectives (i.e. minimize model complexity and maximize accuracy) into account, without assigning priority to one objective or the other. The ultimate choice about which solution should be used in practice is left to the user who can apply his/her background knowledge and experience to choose the “best” solution for his/her particular needs. In addition to the three goals mentioned earlier, our design decisions also address the issues of (i) speeding up the convergent property of the GA, and (ii) being able to solve a problem without requiring any specific knowledge *a priori*.

As discussed earlier, with the exception of the high time complexity, the *SGA-BP-Wrapper* approach has many admirable characteristics that warrant its use. As such, our MOGA for the feature subset selection problem also utilizes the wrapper approach and therefore also shares the same advantages and disadvantages with that of the *SGA-BP-Wrapper* approach. Moreover, modification of the *SGA-BP-Wrapper* approach to handle MOOPs is feasible to implement, since the only major additions are mechanisms to maintain diversity in the GA population. From hereon we interchangeably refer to our approach as the *MOGA-BP-Wrapper* approach.

A recurring theme in GA literature is how to maintain an *optimal* balance between exploitation versus exploration of the search space which inevitably affects the convergence property of the GA [Deb and Goel, 2001]. Recall that the conventional recombination operators (i.e. two-point crossover and mutation operators) are responsible for creating new solutions in the population in order to explore new regions of the search space. Unfortunately, the creation of new individuals in the population due to these operators is at the expense of individuals in the

previous generation that may represent good schemas. In order to offset this destructive effect, we propose a number of mechanisms in our design that not only emphasize exploitation of the search space, but at the same time, force population diversity.

One mechanism to offset the destructive behaviour of the recombination operators is that a population of size $2N$ instead of size N (following the notation from Chapter 1, Figure 1.1) competes for a chance to survive into the next generation. The reason for doing this is that by allowing the solutions from both the parent and offspring populations to compete for survival, fewer good schemas are lost and have to be rediscovered; and therefore the probability of finding a set of Pareto-optimal solution in fewer generations increases.

A second mechanism is our *least-crowded tournament selection operator*. As mentioned earlier, our *least-crowded tournament selection operator* selects as the winner the dominating solution in the tournament and thus helps to preserve better solutions. Besides providing a mechanism to exploit the search space, our *least-crowded tournament selection operator* has a number of other commendable characteristics. One characteristic is that our *least-crowded tournament selection operator* helps maintain population diversity in non-dominating scenarios by selecting the solution that is the least crowded in the next generation mating pool. This explicit mechanism to maintain diversity is extremely important for finding a set of Pareto-optimal solutions that is as uniformly distributed as possible because solutions subjected to the standard SGA tend to cluster to a single optima late in the simulation [Deb, 2001]. Furthermore, because our *least-crowded tournament selection operator* utilizes our novel least crowded method for selecting the “best” solution in a non-dominating scenario, it provides the missing component in ranking a solution in all scenarios for a given MOOP. That is, even though the concept of domination can classify solutions into different non-dominating fronts, it does not provide an algorithm for selecting the “best” solution in a set of non-dominating solutions. The least crowded concept used by our *least-crowded tournament selection operator* is also important in another way; it allows us to continue to use the concept of domination. This is significant

because the concept of domination allows us to rank individuals without having to assign weights to each criterion to reflect their importance. Recall that methods that do this essentially convert a MOOP into a SOOP and are making the assumption that the problem structure is convex. In real-world problems, the underlying structure of a problem is generally not known *a priori*. Another strength that our *least-crowded tournament selection operator* has is that a user-defined parameter is not required in ranking solutions. For example, in the NSGA [Srinivas and Deb, 1994] algorithm a user-defined parameter is required for the sharing function. Based on this parameter, the sharing function determines the fitness level of each solution by redistributing fitness values predicated on the crowding level of a given solution. Determining a correct value for this parameter is not trivial and directly affects how well solutions in a Pareto-optimal front are distributed. However, because our *least-crowded tournament selection operator* is based on our *least-crowded selection algorithm*, its time complexity is also higher than other proposed tournament selection operators such as the one proposed by [Pappa et al., 2002]. Specifically, our *least-crowded tournament selection operator* has a time complexity of $O(k\tau^2)$, where k is the number of competitors in each tournament play and τ is the number of tournaments played. In practice, the time complexity is usually lower than $O(k\tau^2)$ because determination of which competitor is the least-crowded in the existing mating pool is only required in non-dominating scenarios. Hence, the time complexity is generally even lower in practice. A disadvantage of our *least-crowded tournament selection operator* is that it is not feasible when higher values of k are used. In all experiments that we perform, k is equal to two.

Another mechanism is the maintenance of a separate elite set E_i , thereby emphasizing exploitation of the search space. The reason for maintaining a separate elite set is that schemas leading to good solutions should be preserved. The problem with maintaining a separate elite set that is capped at size N is that later in the simulation, most solutions in the population are non-dominating in the first front and are therefore have the potential to be included in the elite set; the result is that the elite set size may exceed N . In order to truncate an elite set with size exceeding

N , we use the *least-crowded selection algorithm* proposed earlier (although we use the elite set to discuss the *least-crowded selection algorithm*, we can truncate any set to a desired size). The *least-crowded selection algorithm* offers a number of worthy characteristics. Firstly, because it increases selection pressure (i.e. higher probability of being selected) on solutions in less crowded regions of the search space, it helps maintain diversity in the population. Secondly, because the *least-crowded selection algorithm* utilizes the same least crowded concept as used by the *least-crowded tournament selection operator*, it allows us to use the concept of domination and thus permit our approach to handle both convex and non-convex problems. In addition, as with the *least-crowded tournament selection operator*, the *least-crowded selection algorithm* does not have any user-defined parameters to set. Although the *least-crowded selection algorithm* helps to overcome a number of problems, its time complexity of $O(N^3)$ is quite high, where N is the number of solutions in E_i . Fortunately, with simple bookkeeping, the time complexity can be reduced to $O(N^2)$.

The break-down of the time complexity of the simple bookkeeping strategy to reduce the time complexity of our *least-crowded selection algorithm* from $O(N^3)$ to $O(N^2)$ is summarized below. First, the time complexity required to find the two furthest solutions in the set E is $O(N^2)$. After the first two solutions are removed from E and added to a set E' , finding the third solution to remove from E and to include in E' requires an additional time complexity of $O(N)$ since a distance calculation from each remaining solution in E is performed for each solution in E' . In the process of finding the third solution, the shortest distance for each solution in E from solutions in E' is noted. To find the fourth solution to remove from E and to include in E' , only the distance of the third solution in E' has to be calculated from each of the remaining solutions in E , and this requires a time complexity of $O(N)$. Again, as in the process of finding the third solution, the shortest distance for each solution in E from solutions in E' is noted and updated. This process is repeated until E' contains N solutions. Therefore, the total time complexity for our *least-crowded selection algorithm* is $O(N^2)$.

Thus far in the analysis, we have almost exclusively concentrated on mechanisms designed to exploit the search space. In MOGA, too much exploitation in the form of elitism may cause the true Pareto-optimal front to be missed if the GA preserves non-dominated solutions that lie away from the Pareto-optimal front. To avoid this, a mechanism that maintains lateral diversity is needed. Lateral diversity is the maintenance of diversity across all fronts, not just along the current non-dominated front. It is important because it helps the search algorithm to avoid convergence to a local Pareto-optimal front. For example, in the later stages of the search most or all of the solutions will belong to the first non-dominated front because the high selection pressure of elitist solutions will cause the rapid deletion of solutions belonging to non-elitist fronts (i.e. second front, the third front, and so forth). If these elite solutions lie in a front that is away from the true Pareto-optimal front, there will not be enough new solutions accepted into the population to provide the diversity needed to escape this local Pareto-optimal front. The reason why maintaining solutions from non-elitist fronts can avoid premature convergence is that they can provide certain decision variables that may not be present in elite solutions, but are needed in order to push the search towards better regions of optimality. Note that this definition of local Pareto-optimal front does not follow Definition 3.5 but instead has the same connotation as when we speak of a backpropagation neural network converging prematurely.

To maintain lateral diversity, our approach incorporates an explicit controlled mechanism that forces the selection of solutions from non-elitist fronts [Deb and Goel, 2001]. Within each front, the *least-crowded selection algorithm* is used to select n solutions from n_k solutions in the k -th front, where $n < n_k$. Hence, this contributes to a time complexity of $O(n_k^2)$ for each front k .

As in most multi-objective evolutionary approaches, our approach requires sorting solutions in the population into non-dominated sets as defined by Definitions 2.6, 2.7, and 2.9. Our proposed *non-dominated sorting algorithm* accomplishes this and requires a time complexity of $O(fMN^2)$, where f is the number of fronts present in the population, M is the number of objectives,

and N is the population size. In the worse case scenario where every solution occupies its own front, the time complexity increases to $O(MN^3)$; however these scenarios are rare. In practice, the time complexity of our *non-dominated sorting algorithm* is lower than that of $O(fMN^2)$ because solutions that are removed and placed in their corresponding non-dominated front after each iteration of the procedure never get visited again. Though our *non-dominated sorting algorithm* is computationally higher than other non-dominated sorting algorithms, it does not require additional storage requirements and it is simple to implement. For example, although the non-dominated sorting algorithm proposed by [Srinivas and Deb, 1994] has a lower time complexity of $O(MN^2)$, it requires additional storage space of $O(N^2)$ and its implementation is more complicated.

Combining the time complexities of each module, our approach has a total time complexity of $O(mN^2W)$. Recall that m is the number of generations and W is the linear time complexity of the back-propagation algorithm.

From the analysis of our approach, it can be seen that our approach has the capability to simultaneously satisfy all three goals of maximizing accuracy, minimizing model complexity, and providing greater selection for the user than when the BP algorithm is executed alone or when the *SGA-BP-Wrapper* approach is used. Though the time complexity of our approach is higher than the time complexity of the BP algorithm and the *SGA-BP-Wrapper* approach, it is still within a manageable bound. That is, our approach has a high probability of finding a set of Pareto-optimal solutions within polynomial time on a subset of real-world dataset that is of non-trivial size and complexity.

3.4 Conclusion

In this chapter we started out by defining what an *optimal* feature subset means for our purposes. Due to a number of unknown parameters in a given problem, we have defined the results of a

learning algorithm to be weakly or *approximately optimal* to reflect this realistic uncertainty. With the appropriate definitions, we went on to define the complexity of the feature subset selection problem and analyzed in detail three approaches with respect to our three goals of maximizing accuracy, minimizing model complexity, and increasing selection for the user. The analysis of the three algorithms shows our approach to be most feasible in obtaining the required objectives. In the next chapter, we validate our theoretical results from this chapter. We also provide a detailed outline of the experiments we performed and examine in detail the results of the algorithms on three non-trivial real-world datasets.

Chapter 4

Empirical Validation and Analysis

We now validate the theoretical results from the previous chapter by providing empirical evidence that our multi-objective genetic algorithm does offer an effective means to finding a set of Pareto-optimal feature subsets that simultaneously maximize predictive accuracy rate and minimize the feature subset size.

4.1 Introduction

To validate our theoretical results from the previous chapter, four experiments will be performed for two real-world datasets: (i) the ionosphere dataset, and (ii) the hepatitis dataset. In the first experiment, we run a regular backpropagation (BP) algorithm by itself for feature detection and make note of its predictive accuracy rate. We also apply the *SGA-BP-Wrapper* approach (see Chapter 3) and our approach to the two datasets to determine their performance level (i.e. predictive accuracy rates and the number of features selected). Thereafter the performances of all three approaches are then compared based on the foregoing two metrics.

In the second experiment we investigate the convergence property of our approach and the *SGA-BP-Wrapper-CV* approach. We compare the performance of the two approaches by applying a Wilcoxon Rank-Sum test (See Appendix A). In the third and fourth experiment, we

investigate the effects lateral and vertical population diversity has on the performance of a Genetic Algorithm (GA), respectively.

This chapter is divided into a number of sections. Section 4.2 discusses the 10-fold cross-validation procedure. In Section 4.3, a description of the datasets is given. Section 4.4 discusses in detail the procedure used to prepare the data so that it is in a suitable format for the algorithms used in this thesis. Sections 4.5, 4.6, 4.7, and 4.8 examines in detail the four experiments mentioned above. Section 4.9 summarizes our empirical results.

4.2 10-Fold Cross-Validation

The procedure used to perform most of the experiments in this thesis is the 10-fold cross-validation procedure [Pappa et al., 2002]. The 10-fold cross-validation procedure starts off by dividing the entire dataset into a test set and a training set. The test set contains the exemplars used to test the performance level of the network after training with the exemplars in the training set is complete. After the test set and the training set are determined, the training set is further divided into ten equal size sets p_i , where $i = 1, 2, \dots, 10$. Thereafter, the following procedure is performed k times, where $k = 1, 2, \dots, 10$ (we refer to each iteration k as a *fold* and a specific iteration k using a specific training set and validation set as *fold k*). First of all, a partition p_j^k is selected as the validation set for iteration k , where $j = 1, 2, \dots, 10$. The validation set contains exemplars that are not used to directly train the network but are instead used to test the predictive accuracy level of the network during specific intervals in which the training of the network is temporarily stopped in order to determine whether or not to continue the training session (see section 4.5.1).

After the validation set is selected, the remaining partitioned sets $p_{i \neq j}^k$ are combined and used as the training set for *fold k*. When training of the network is complete, its performance level (i.e.

accuracy rate) is determined by using the resulting weights of the network to predict the class label of the exemplars in the test set. After all instances of a given algorithm are executed, the procedure is repeated with $k=k+1$ and $j=j+1$ (i.e. the validation set selected for iteration $k+1$ is p_{j+1}^{k+1}). Again, the remaining partitions, not including p_{j+1}^{k+1} , are combined to form the new training set which is then used to train the network. Figure 4.1 shows the general flow of the 10-fold cross-validation procedure.

The reason for using the 10-fold cross-validation procedure is because the performance of different executions should not be significantly different for different subsets of data used to train and validate the network. Thus, to ensure that the performance of an algorithm is not higher than it should be due to some bias distribution present in some training sets, validation sets, or the test set, we expose the algorithm to different subsets of the data and measure the performance over all execution instances.

Figure 4.1: 10-Fold Cross-Validation Procedure

4.3 Data Collection and Description

4.3.1 Ionosphere Database

The first dataset used to attain the empirical results is known as the “Ionosphere Database” from the Space Physics Group at Johns Hopkins University [Machine Learning Repository, 2004]. Collected by a system consisting of a phased array of 16 high-frequency antennas with a total transmission power on the order of 6.4 kilowatts in Goose Bay, Labrador, each of the 351 radar data instances in the dataset is comprised of 34 predictor attributes (i.e. features) and one target

attribute. The 34 predictor attributes are continuous and correspond to the 17 pulse numbers for the Goose Bay system, where each pulse number is described by two attributes resulting from the complex values returned by the function due to the electromagnetic signal. The function is an autocorrelation function that processes the received radar signal by taking two arguments (i) the time of a pulse, and (ii) the pulse number. The target values in this dataset correspond to free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere, while "Bad" returns are those that do not (i.e. their signals pass through the ionosphere). There are a total of 225 "Good" instances and 126 "Bad" instances in the dataset. Moreover, there are no missing attribute values.

We selected the ionosphere dataset because it has been shown by [Sigillito et al., 1989] to have empirically enough attributes to do feature subset selection in addition to having enough data points for training the network. Also, because there are no missing attribute values, it also serves as a simple dataset for testing the algorithms in this thesis without having to address the issue of assigning correct values to data points with missing attribute values.

4.3.2 Hepatitis Database

The second dataset used for the experiments is a medical database containing hepatitis patient information. This dataset was originally donated by Bojan Cestnik from the Jozef Stefan Institute and was subsequently released by Gail Gong from Carnegie-Mellon University [Machine Learning Repository, 2004]. Containing a total of 155 data instances, each data instance in the dataset consists of 19 predictor attributes and one target attribute. Table 4.1 provides a summary of the predictor attributes and the target attribute.

Table 4.1: Dataset Attributes: Hepatitis

Attribute Name	Attribute Type	Bin Values
AGE	Predictor, Continuous	10, 20, 30, 40, 50, 60, 70, 80
SEX	Predictor, Categorical	male, female
STEROID	Predictor, Categorical	no, yes
ANTIVIRALS	Predictor, Categorical	no, yes
FATIGUE	Predictor, Categorical	no, yes
MALAISE	Predictor, Categorical	no, yes
ANOREXIA	Predictor, Categorical	no, yes
LIVER BIG	Predictor, Categorical	no, yes
LIVER FIRM	Predictor, Categorical	no, yes
SPLEEN PALPABLE	Predictor, Categorical	no, yes
SPIDERS	Predictor, Categorical	no, yes
ASCITES	Predictor, Categorical	no, yes
VARICES	Predictor, Categorical	No, yes
BILIRUBIN	Predictor, Continuous	0.39, 0.80, 1.20, 2.00, 3.00, 4.00
ALK PHOSPHATE	Predictor, Continuous	33, 80, 120, 160, 200, 250
SGOT	Predictor, Continuous	13, 100, 200, 300, 400, 500
ALBUMIN	Predictor, Continuous	2.1, 3.0, 3.8, 4.5, 5.0, 6.0
PROTIME	Predictor, Continuous	10, 20, 30, 40, 50, 60, 70, 80, 90
HISTOLOGY	Predictor, Categorical	No, yes
Class	Target, Categorical	DIE, LIVE

In Table 4.1 the third column (i.e. “Bin Values”) contains the discretized bin values for each attribute for the purposes of data analysis as released by the original author of the dataset. With respect to categorical attributes, the bin values are just the categories for those attributes.

There are several things to note from Table 4.1. The first thing to note is the “BILIRUBIN” attribute. The donator indicates that “BILIRUBIN” is a continuous attribute and that the six real values associated with the attribute are so called “boundary” values; according to these “boundary” values the attribute can be discretized for experimental purposes. For our purposes, we treat “BILIRUBIN” as a continuous attribute.

Though the donator only noted the “BILIRUBIN” attribute as continuous, examining the data instances reveal that the “ALBUMIN” attribute is also continuous. Moreover, the “AGE”, “ALK PHOSPHATE”, and “PROTIME” attributes in the dataset also show that these are non-categorical integer attributes. The last attribute in Table 4.1 is the target attribute indicating the

status of the patient. There are a total of 32 “DIE” instances and a total of 123 “LIVE” instances in the dataset.

Unlike the ionosphere dataset, this dataset contains missing attribute values. Table 4.2 provides a summary of the attributes and the number of missing values associated with it.

Table 4.2: Missing Attribute Values: Hepatitis

Attribute Name	Number of Missing Values
AGE	0
SEX	0
STEROID	1
ANTIVIRALS	0
FATIGUE	1
MALAISE	1
ANOREXIA	1
LIVER BIG	10
LIVER FIRM	11
SPLEEN PALPABLE	5
SPIDERS	5
ASCITES	5
VARICES	5
BILIRUBIN	6
ALK PHOSPHATE	29
SGOT	4
ALBUMIN	16
PROTIME	67
HISTOLOGY	0
Class	0

We selected the hepatitis dataset as our second dataset because of the fact that some of its attributes contain missing attribute values. That is, we could utilize our approach to test the feasibility of the strategy used to assign values to missing attribute values. If consistently selected, it may be that the strategy used to assign values to missing attribute values may in some ways reflect the actual values taken on by these attributes; this can lead to further investigation of the selected attributes and the missing attribute value assignment strategy. In our experiments, we use a technique that simply assigns random values to missing attribute values based on a range

that is determined by the dataset itself. This technique was used just because it is simple to implement.

4.4 Data Format

Before the ionosphere and the hepatitis datasets can be used to train and test the network, each dataset must be divided into training sets and test sets. Each class label in the training and testing set should be fairly represented with respect to the class distribution. Because the 10-fold cross-validation procedure is used, partitioning of the training set into 10 equal size subsets should have a similar proportion of class labels. Another data preparation step required is that data instance attributes have to be encoded into a numerical format that is suitable for the backpropagation neural network. After the required encoding of the attribute values are performed, mathematical transformation can be applied that may enhance the learning capability of the induction algorithm.

4.4.1 Ionosphere Dataset

To divide the entire dataset into their respective training set and test set, we adopted the scheme used by [Sigillito et al., 1989] since they investigated this dataset using a backpropagation algorithm and obtained impressive results. Following [Sigillito et al., 1989]'s division scheme, our training set used the first 200 instances for training, which were carefully split almost 50 percent "Good" instances and 50 percent "Bad" instances. The remaining 151 test instances consisted of 123 "Good" instances and only 25 "Bad" instances. As mentioned earlier, data instances in the training set are then divided into 10 equal partitions with class label proportions that are as similar as possible for each class.

After the training and test set are determined, the data instances with target values “Bad” and “Good” are encoded into their numerical representation of “0.0” and “1.0”, respectively. Since the predictor attributes are already in a continuous format, it is not necessary to perform any changes on them and so are kept in their raw numerical format.

After the non-numeric target values are replaced with their numerical representation, the predictor attributes are standardized to obtain “standard normal” random attribute variables with mean 0 and standard deviation 1 [Neural Network FAQ, 2004]. The following formulas are used to standardize the input attribute X [Neural Network FAQ, 2004]:

$$\bar{X} = \frac{\sum_{i=1}^N x_i}{N}, \quad (4.1)$$

$$\sigma_X = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{X})^2}{N-1}}, \quad (4.2)$$

$$s_i = \frac{x_i - \bar{X}}{\sigma_X}. \quad (4.3)$$

In the above formulations, x_i represents the raw predictor attribute value x for the i th training exemplar; N is the number of training exemplars; $\bar{X}$ is the mean value of input attribute X ; σ_X is the standard deviation of input attribute X ; and s_i is the standardized value corresponding to x_i . The resulting standardized value s_i corresponding to x_i for each raw predictor attribute x are used as the network inputs. The numerical target values of 0 and 1 are not standardized because we use a logistic function as the output activation function, which has a range of 0 to 1. Literature on why inputs into the network should first be standardized can be found in [Haykin, 1999; Neural Network FAQ, 2004].

4.4.2 Hepatitis Dataset

In all our experiments using the hepatitis dataset, the training set contained a total of 110 exemplars; 88 belonging to the “LIVE” class label while the remaining 22 belonging to the “DIE” class label. The test set contained 45 exemplars; 35 belonging to the “LIVE” class label while the remaining 10 belonging to the “DIE” class label. The proportion of class labels within the training and test sets were purposely chosen to reflect approximately the proportion of each class label in the entire dataset. As with the ionosphere dataset, data instances in the training set were then divided into 10 equal partitions with class label proportions that are as similar as possible for each class.

Like the ionosphere dataset, the hepatitis dataset is a binary classification system and so the categorical target variable values “DIE” and “LIVE” are encoded into their numerical representation of “0.0” and “1.0”, respectively, as in the ionosphere dataset. However, unlike the ionosphere dataset, the hepatitis dataset contains a mix of continuous and non-categorical predictor attributes. In this dataset, all the categorical predictor attributes are binary categorical attributes and so one dummy variable is created to represent each attribute; one class is arbitrarily chosen to be 0.0 while the other is 1.0. The continuous attributes are left unchanged.

In the hepatitis dataset, a number of predictor attributes do not have values (i.e. they have missing values). For the attributes containing missing values, the values are randomly chosen in the permitted range. The reason for doing this is to see whether the GA will actually select the attributes containing a high number of randomly chosen values. For example, the attribute “PROTIME” has 67 missing values and so this attribute should have a low selection pressure (i.e. a high probability of not being selected).

As mentioned in the foregoing paragraph, missing attribute values are randomly assigned value in a permitted range. For the categorical attributes, the missing values are randomly chosen as being either 0.0 or 1.0. For the continuous variables, a real value is randomly selected in a range appropriate for the variable. The ranges for the different predictor variables are:

- “BILIRUBIN”: 0.30 to 5.0;
- “ALK PHOSPHATE”: 26.0 to 295.0;
- “SGOT”: 13.0 to 500.0;
- “ALBUMIN”: 2.1 to 6.0; and
- “PROTIME”: 10.0 to 100.0.

The range for each continuous attribute listed above are chosen by examining the bin values in Table 4.1 under the column “Bin Values” and at the actual values taken on by that attribute in the dataset. Specifically, the lower bound is the lesser of the lowest value in the dataset or the lower bound indicated in Table 4.1 under the column “Bin Values” for the given attribute; and the upper bound is the greater of the highest value in the dataset or the upper bound indicated in Table 4.1 under the column “Bin Values” for the given attribute. Determination of the range is done this way for the continuous attribute in order to accommodate as many unforeseen values as possible as indicated by the range in Table 4.1 and the actual dataset.

After the encoding and the missing values are determined, the predictor attributes are standardized using Equations 4.1, 4.2, and 4.3 as in the ionosphere dataset. As with the ionosphere dataset, the standardized values are used as network inputs.

4.5 Accuracy Rate and Model Complexity Comparison

In this section we partially validate some of the theoretical results discussed in Chapter 4. Specifically, we compare the performance (i.e. accuracy rate and model complexity) of the BP

algorithm alone, the *SGA-BP-Wrapper* approach (see Chapter 3), and our approach (i.e. the *MOGA-BP-Wrapper*) when applied on the ionosphere and hepatitis datasets. The following three subsections describe in detail the network architectures used to generate the results required for the comparison.

4.5.1 The BP-CV Approach

To generate the results (i.e. accuracy rate) for the BP run alone on the selected datasets, we make use of the 10-fold cross-validation procedure described earlier. In particular, we replace algorithm x (dotted box) in Figure 4.1 with the algorithm listed in Figure 4.2. We refer to this resulting algorithm as the *BP-CV* approach (*cv* is an abbreviation for cross-validation).

1. Let N be the population size.
Let n be the initial number of hidden nodes.
Let m be a fixed multiple constant.
Let *fold* $k = 1$.
2. (i) Initialize a counter variable $i = 1$.
(ii) Execute the backpropagation algorithm with $n + (i - 1)m$ hidden nodes for j iterations.
(iii) Increment i by 1; if $i \leq 10$ go back to step 2(ii), otherwise go to step 3.
3. Increment k by 1; if $k \leq 10$ go back to step 2, otherwise go to step 4.
4. Done.

Figure 4.2: Module in BP-CV Approach

Notice in step 2(ii) of Figure 4.2 that the BP is executed multiple times j using the same number of hidden nodes. The reason for doing this is to provide redundancy because the backpropagation algorithm is known for converging prematurely. Moreover, the reason for varying the number of hidden nodes while keeping the remaining parameters fixed is because it directly determines the feature search space of a given problem [Haykin, 1999], and variations in the number of inputs

(i.e. features selected) seem to have some correlation with the number of hidden nodes [Yang and Honavar, 1998].

In all the experiments that we perform for this approach and the other two approaches, the logistic function is used as the output activation function because both our problems are binary classification problems. Since the output from the logistic function can approach 0.0 or 1.0, but never attain these values, limiting values are required in order to classify whether a prediction is in one class or the other. For this experiment and all subsequent experiments, we use ± 0.21 as the limiting values. Thus, any prediction value less than $0.0+0.21$ (i.e. 0.21) is considered to be in the class with target value 0.0, while any prediction value greater than $1.0-0.21$ (i.e. 0.79) is considered to be in the class with target value 1.0. These limiting values were chosen because their proportion to the entire output range of (0.0, 1.0) of the logistic function output correspond to the same proportion that the limiting values of the hyperbolic tangent function $\varphi(v) = a \tanh(bv)$ possess with output range (-1.7159, 1.7159), where $a = 1.7159$ and $b = \frac{2}{3}$ [Haykin, 1999].

Like the logistic activation function, all experiments use the hyperbolic tangent function as the hidden layer sigmoid activation function (i.e. hyperbolic tangent). The hyperbolic tangent function is used because it tends to yield faster training due to the fact that it produces both positive and negative value (i.e. better numerical conditioning); see [Neural Network FAQ, 2004].

To help enhance the learning capability of the network, we randomize (i.e. shuffle) the order in which the training exemplars are presented to the network from one epoch to the next [Haykin, 1999]. Moreover, because the backpropagation algorithm is known to have overtraining problems, we try to avoid this by using a stopping criterion that is based on how well the network performs in predicting exemplars in the validation set. Figure 4.3 illustrates the typical behaviour

of a network from experimental experience available in neural network literature and show how it is used to stop training [Hecht-Nielsen, 1991].

Figure 4.3: Method to Avoid Overtraining (From [Hecht-Nielsen, 1991])

In Figure 4.3, the horizontal axis measures the number of presentations of the training set to the network while the levels of performance are measured along the vertical axis (i.e. error rate). At periodic intervals in this process (i.e. 30 epochs in all our experiments), the network training is stopped, the weights are temporarily frozen, and the network's error performance is tested using, in our case, mean squared error (MSE) on the validation set. If the MSE on the validation set for the current test period is lower than or equal to the previous test period, training is continued. On the other hand, if the MSE on the validation set for the current test period is higher than the previous test period, training is stopped; and the resulting weights are then applied to the test set to test the performance of the network.

Notice in Figure 4.3 that the error curve of training set continues to decrease, while the validation set error typically decreases for a while but then begins to increase again (often it eventually seems to level out, but not always) [Hecht-Nielsen, 1991]. It is at the point where the validation set error starts to increase that the network would be stopped.

4.5.2 The SGA-BP-Wrapper-Cross-Validation Approach

As mentioned at the beginning of this section, the second algorithm used in this is the *SGA-BP-Wrapper* approach. Because we are combining the *SGA-BP-Wrapper* approach with the 10-fold cross-validation procedure in this experiment, from hereon we refer to it as the *SGA-BP-Wrapper-CV* approach. Shown in Figure 4.4 is the *SGA-BP-Wrapper* algorithm that is executed in place of algorithm x in the dotted box in Figure 4.1 to give us the *SGA-BP-Wrapper-CV* approach.

1. Create initial random population $P_{i=0}$ of size N .
2. Evaluate each solution in P_i by using the induction algorithm I , where P_i is the population for the i th generation.
3. Perform selection, crossover, and mutation on P_i to get P_{i+1} .
4. If the stopping criterion is satisfied, go to step 5, otherwise go back to step 2.
5. Done.

Figure 4.4: SGA-BP-Wrapper Algorithm

A close examination of the algorithm listed in Figure 4.4 will reveal that it is the same algorithm found in [Goldberg, 1989] but modified for our feature subset selection problem. The algorithm starts off by initializing a population $P_{i=0}$ consisting of N candidate GA individuals (i.e. solutions) that are randomly generated, where i is the generation executed. We require that at least one feature is selected from the random process. As mentioned earlier, we use a binary encoding scheme to represent whether or not a feature is selected for testing with our neural network. A “1” means the feature is selected while a “0” means it is not. After the initial population is generated, each candidate solution n with its respective chosen features is trained and tested using the induction algorithm I (in our case, the backpropagation algorithm) for the selected training and test sets; the predictive accuracy level of the trained network then becomes the fitness level

associated with solution n (step 2). Having determined each candidate solution's fitness value, a new population p_{i+1} containing N individuals for the next generation are selected by subjecting the solutions in p_i to selection, crossover, and mutation based on its fitness level (step 3). Since we are maximizing the predictive accuracy rate, solutions with a higher fitness level have a higher selection pressure. Thereafter, the fitness level of each solution in p_{i+1} is determined as before by applying the solution to the neural network (go back to step 2). This procedure repeats until some stopping criterion is satisfied. In our experiment, the stopping criterion in step 4 is the maximum number of generations, which is set by the user.

In applying the *SGA-BP-Wrapper-CV* approach to the selected datasets, we decided to use the tournament selection operator because, as mentioned in Chapter 2, much GA literature have shown good results associated with its use. Moreover, we decided to use the two-point crossover operator for binary encoding because we believe that it provides the right balance in terms of exploring the search space; that is, it is not as disruptive as uniform crossover and it offers more exploratory power than the one-point crossover operator. The mutation operator is just the standard mutation operator used with binary encoding [Goldberg, 1989].

In *BP-CV* approach, we noted that the number of hidden nodes to use is a significant factor for feature detection and we proposed a deterministic strategy for assigning a variety of different numbers of hidden nodes used to execute each instance of the BP algorithm. With the *SGA-BP-Wrapper-CV* approach wherein the number of inputs (i.e. features) varies due to the genetic operators of the SGA, using a deterministic scheme may hinder performance [Yang and Honavar, 1998]. As a result, we introduce a stochastic strategy that will search for the *optimal* number of hidden nodes. That is, the number of hidden nodes is another attribute possessed by the GA individual and is subject to evolution during the mating process. However, because the objective of the search is to only maximize predictive accuracy, the number of hidden nodes is only used to ensure that solutions with a higher fitness level selected for subsequent generations will execute with a similar number of hidden nodes that contributed to their high performance in the first

place. Moreover, evolution of the number of hidden nodes is subjected only to crossover and not mutation because we want to avoid destructive changes made to good solutions.

The crossover operator used to evolve the number of hidden nodes is a real-parameter crossover operator introduced by [Eshelman and Schaffer, 1993] called a blend crossover (BLX- α). For two parent solutions h_i^1 and h_i^2 in the i -th generation (assuming $h_i^1 < h_i^2$), the BLX- α randomly picks a solution in the range $[h_i^1 - \alpha(h_i^2 - h_i^1), h_i^2 + \alpha(h_i^2 - h_i^1)]$. The α parameter is a user defined constant in the range $[0.0, 1.0]$. Notice that the greater the value for α , the larger the range from which a solution can be selected.

Two more user-defined parameters are associated with the use of the BLX- α crossover: (i) the smallest number of hidden nodes allowed $h^{(L)}$, and (ii) the largest number of hidden nodes allowed $h^{(U)}$. Thus, part of initializing the GA population is to randomly select the number of hidden nodes in the range $[h^{(L)}, h^{(U)}]$.

4.5.3 The MOGA-BP-Wrapper-Cross-Validation Approach

The network architecture used to generate the results for our approach is very similar to the *SGA-BP-Wrapper-CV* approach; that is, instead of using the SGA, we use our *MOGA-BP-Wrapper* algorithm. Since this network architecture also makes use of the 10-fold cross-validation procedure, we call it the *MOGA-BP-Wrapper-CV* approach. Specifically, we run our algorithm listed in Figure 1.1 in place of algorithm x (in dotted box) in Figure 4.1.

Since our approach deals with multiple objectives that are simultaneously optimized without assigning importance to any one objective, scaling of the objectives are required. Scaling is performed on the objective space vector (i.e. the number of features selected or the accuracy rate) because we decided to work with the objective space in our MOGA. In all the experiments using

our MOGA, we standardized the objectives by multiplying the accuracy rate of a GA individual by the total number of features in the dataset.

4.5.4 Ionosphere Dataset: Parameters Settings and Results

Table 4.3 summarizes the parameters settings used for the ionosphere dataset for each approach in this experiment.

There are several things to note about Table 4.3:

1. In the row labeled “Number of Inputs”, the value “Varies” indicate that this value is to be determined by the simulation.
2. The learning and the momentum rates were determined by trial and error by running the backpropagation algorithm with all the features in the dataset; the rates that gave the highest accuracy rate for different numbers of hidden nodes were chosen for all three approaches.
3. The mutation rate for the GA was determined by dividing 1 by the length of the chromosome [Goldberg, 1989].
4. The crossover rate was set at 0.6 to ensure that enough new solutions are created to explore the search space while not allowing its destructive effect to destroy too much previously discovered schemas.
5. The notation $[\gamma_L, \gamma_U]$ corresponding to the parameter “Number of Hidden Nodes” for the “SGA-BP-Wrapper-CV” and the “MOGA-BP-Wrapper-CV” approach mean that the number of hidden nodes is assigned a value between γ_L and γ_U , inclusive.
6. We use a value of 0.20 for the BLX- α operator constant since we do not want drastic changes.

7. We use a lateral diversity reduction rate of 0.50 because it offers a balance between emphasizing solutions from the first fronts and other fronts.
8. NA means “Not Applicable”.

Table 4.3: Network Parameters: Ionosphere

Parameter	<i>BP-CV</i>	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Number of Inputs	34	Varies	Varies
Number of Hidden Nodes	Initial number of hidden nodes $n = 1$; Fixed multiple constant $m = 1$	[3, 68]	[3, 68]
Number of Output Nodes	1	1	1
Learning Rate	0.2	0.2	0.2
Momentum Rate	0.65	0.65	0.65
Intervals Before Running validation set	30 Epochs	30 Epochs	30 Epochs
Hidden Layer Activation Function	Hyperbolic Tangent	Hyperbolic Tangent	Hyperbolic Tangent
Output Activation Function	Logistic	Logistic	Logistic
Maximum Number of Epochs	5000	5000	5000
Number of Iterations j	10	NA	NA
Population Size	150	50	50
Elitism	NA	Yes	Yes
Selection Operator	NA	2-Player Tournament	2-Player Least-Crowded Tournament
Crossover Operator	NA	2-point	2-point
Length of Chromosome	NA	34	34
Crossover Probability	NA	0.6	0.6
Mutation Probability	NA	0.029	0.029
Maximum Number of Generations	NA	50	50
BLX- α Operator Constant	NA	0.20	0.20
Lateral Diversity	NA	NA	0.50
Reduction Rate			

We now compare the three approaches described above by examining and comparing their performance (i.e. predictive accuracy rate and the number of features selected) for the ionosphere dataset. Though it does not seem correct or even fair to compare the *BP-CV* approach with the other two approaches in terms of the number of features selected, we still do it because we are interested in knowing whether the *SGA-BP-Wrapper-CV* approach and our approach are capable of selecting significantly fewer features to justify the additional complexity that they bring. Table 4.4 summarizes the results found in this experiment. The values are rounded to the nearest 1000 after the decimal point.

In Table 4.4, the statistics for each approach were derived by selecting the solution with the highest accuracy rate from each *fold*, and then using that value to calculate the arithmetic mean and the standard deviation. The statistics for the number of features selected were also calculated by using the same solutions used to calculate the statistics for the accuracy rate. Additionally, the “Confidence Interval” for the accuracy rate and number of features selected statistics were determined by subtracting and adding the standard deviation from and to the arithmetic mean to get the low and high interval values, respectively.

Table 4.4: Accuracy Rate and Model Complexity Results: Ionosphere

Statistical Parameter	<i>BP-CV</i>	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Accuracy Rate: Arithmetic Mean	87.616%	93.709%	94.106%
Accuracy Rate: Standard Deviation	2.209%	0.715%	0.852%
Accuracy Rate: Confidence Interval	(85.407%, 89.825%)	(92.993%, 94.424%)	(93.254%, 94.958%)
Number of Features: Arithmetic Mean	34	15.7	9.1
Number of Features: Standard Deviation	0	5.519	3.035
Number of Features: Confidence Interval	(34, 34)	(10.181, 21.219)	(6.065, 12.135)

Comparing the results of the *BP-CV* approach to that of the *SGA-BP-Wrapper-CV* approach for this dataset, we see that the results obtained by the latter approach are significantly better. With respect to the predictive accuracy rate, the *SGA-BP-Wrapper-CV* approach on average outperforms the *BP-CV* approach by 6.093 percent. Also, notice that the “best” solutions (i.e. solutions with the highest accuracy rate for each *fold*) discovered by the *BP-CV* approach has a large variation in terms of accuracy rate, ranging from 85.407 percent to 89.825 percent, while the solutions found by the *SGA-BP-Wrapper-CV* approach are fairly consistent, ranging from 92.993 percent to 94.424 percent. Examination of the number of features selected indicates that the *SGA-BP-Wrapper-CV* approach used on average 18.3 fewer features (i.e. 34 versus 15.7 features used), which is a significant difference.

Next, we compare the *BP-CV* approach to that of our approach (i.e. the *MOGA-BP-Wrapper-CV* approach). The results show a very similar outcome to the one discussed above. In terms of accuracy rate, our approach outperforms the *BP-CV* approach on average by 6.490 percent. Similarly to the *SGA-BP-Wrapper-CV* approach, the solutions discovered by our approach are fairly consistent, ranging from 93.254 percent to 94.958 percent. With respect to the number of features selected, our approach used on average 24.9 fewer features than the *BP-CV* approach, an even wider margin when compared to the difference between the *BP-CV* and the *SGA-BP-Wrapper-CV* approach.

Finally, we compare our approach to that of the *SGA-BP-Wrapper-CV* approach. For this dataset, the results show that on average our approach performs equally as well as the *SGA-BP-Wrapper-CV* approach even though our approach had to search for a set of solutions with multiple conflicting objectives (i.e. a difference of 0.397 percent in favour of our approach). Furthermore, comparing the number of features selected indicates that on average our approach selected 6.6 less features than the *SGA-BP-Wrapper-CV* approach (i.e. 9.1 versus 15.7). Additionally, the low and high values for the Confidence Interval for the number of features selected for our approach (i.e. (6.065, 12.135)) is almost half the number of features used by the *SGA-BP-Wrapper-CV* approach (i.e. (10.181, 21.219)).

Since we utilize the same data partitioning scheme used by [Sigillito et al., 1989] to determine our training and test sets, it is not infeasible for us to assume the accuracy on "Good" instances to be much higher than for "Bad" instances as was the case for [Sigillito et al., 1989]. However, the results obtained by [Sigillito et al., 1989] showed a higher accuracy rate than the *BP-CV* approach and our approach. Specifically, [Sigillito et al., 1989] obtained an average of over 96 percent accuracy on the remaining 150 test instances, which is about 8 and 1.5 percent higher than the *BP-CV* approach and our approach, respectively. However, unlike our experiments which exclusively used the regular backpropagation algorithm, the results of [Sigillito et al., 1989] were obtained by running different variants of the backpropagation algorithm after a periodic number of epochs for a number of different numbers of hidden units (in $[0, 15]$).

Even though higher accuracy rates were obtained by [Sigillito et al., 1989] compared to that of our approach, the former approach does not fulfill our third metric (i.e. providing greater selection for the user) because an explicit mechanism is not used to search for a set of Pareto-optimal solutions; a later experiment will show this to be also true for the *SGA-BP-Wrapper-CV* approach. Moreover, besides providing more selection for the user, the feature subsets returned by our approach has a fairly high intersection rate, implying economic saving since emphasis has to be placed on fewer features during data collection and analysis. Table 4.5 illustrates the Pareto-optimal feature subsets discovered by our approach for *fold 1*.

Table 4.5: Features Selected for Fold 1: Ionosphere

Solution	Features Selected
1	9
2	1, 4
3	4, 5, 8
4	1, 4, 5, 6
5	1, 4, 5, 6, 31, 32

Notice that in Table 4.5, except for features 8 and 9, the intersection of the remaining features selected is fairly high amongst the solutions in the Pareto-optimal front. Moreover, this pattern is quite similar in the Pareto-optimal front in the other *folds*. However, the intersection of features amongst solutions in the Pareto-optimal front across different *folds* is significantly lower (i.e. only feature 1 was selected for all *folds*).

4.5.5 Hepatitis Dataset: Parameters Settings and Results

Table 4.5 provides a summary of the parameters settings used for the ionosphere dataset for each approach in this experiment. The parameters in this table are the same parameters present in Table 4.3.

Table 4.6: Network Parameters: Hepatitis

Parameter	<i>BP-CV</i>	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Number of Inputs	19	Varies	Varies
Number of Hidden Nodes	Initial number of hidden nodes $n = 1$; Fixed multiple constant $m = 1$	[2, 38]	[2, 38]
Number of Output Nodes	1	1	1
Learning Rate	0.2	0.2	0.2
Momentum Rate	0.7	0.7	0.7
Intervals Before Running validation set	30 Epochs	30 Epochs	30 Epochs
Hidden Layer Activation Function	Hyperbolic Tangent	Hyperbolic Tangent	Hyperbolic Tangent
Output Activation Function	Logistic	Logistic	Logistic
Maximum Number of Epochs	5000	5000	5000
Number of Iterations j	10	NA	NA
Population Size	300	Yes	Yes
Elitism	NA	2 Player Tournament	<i>Least-Crowded Selection Tournament</i>
Selection Operator	NA	2-point	2-point
Crossover Operator	NA	50	50
Length of Chromosome	NA	19	19
Crossover Probability	NA	0.6	0.6
Mutation Probability	NA	0.052	0.052
Maximum Number of Generations	NA	50	50
BLX- α Operator Constant	NA	0.20	0.20
Lateral Diversity Reduction Rate	NA	NA	0.50

We now examine and compare the predictive accuracy rate and the number of features selected for the three approaches using this dataset. Table 4.7 summarizes the results found in this experiment. The results for this dataset are quite similar to the results performed on the ionosphere dataset. Comparing the results of the *BP-CV* approach to that of the *SGA-BP-Wrapper-CV* approach for this dataset, we see that the results obtained by the latter approach are significantly better. With respect to the predictive accuracy rate, the *SGA-BP-Wrapper-CV* approach on average outperforms the *BP-CV* approach by 6.889 percent, which is slightly higher than it was for the ionosphere dataset (6.093 percent). However, for this dataset, the “best” solutions (i.e. solutions with the highest accuracy rate) discovered by the *BP-CV* approach have a smaller variation in terms of accuracy rate than it did for the ionosphere dataset, ranging from 80.996 percent to 86.559 percent. The solutions found by the *SGA-BP-Wrapper-CV* approach are fairly consistent as in the case of the ionosphere dataset, ranging from 89.770 percent to 91.604 percent. Examination of the number of features selected indicates that the *SGA-BP-Wrapper-CV* approach used on average 8.8 fewer features (i.e. 19 versus 10.2 features used) than the *BP-CV* approach; a significant difference.

Table 4.7: Accuracy Rate and Model Complexity Results: Hepatitis

Statistical Parameter	<i>BP-CV</i>	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Accuracy Rate: Arithmetic Mean	83.778%	90.667%	90.444%
Accuracy Rate: Standard Deviation	2.781%	0.937%	1.073%
Accuracy Rate: Confidence Interval	(80.996%, 86.559%)	(89.770%, 91.604%)	(89.471%, 91.518%)
Number of Features: Arithmetic Mean	19	10.2	9.6
Number of Features: Standard Deviation	0	1.989	1.897
Number of Features: Confidence Interval	(19, 19)	(8.211, 12.189)	(7.703, 11.497)

Comparing the *BP-CV* approach to that of our approach (i.e. the *MOGA-BP-Wrapper-CV* approach), we find a similar outcome to the one above. In terms of accuracy rate, our approach outperforms the *BP-CV* approach on average by 6.666 percent, which is slightly higher than it was for the ionosphere dataset (6.490 percent). Similarly to the *SGA-BP-Wrapper-CV* approach, the solutions discovered by our approach are fairly consistent, ranging from 89.471 percent to 91.518 percent. With respect to the number of features selected, our approach used on average 9.4 fewer features than the *BP-CV* approach; again, a significant difference.

A comparison of the accuracy rate with our approach to that of the *SGA-BP-Wrapper-CV* approach for this dataset shows that on average our approach performs equally as well as the *SGA-BP-Wrapper-CV* approach (i.e. a difference of 0.223 percent in favour of the *SGA-BP-Wrapper-CV* approach). Unlike the ionosphere dataset, a comparison of the number of features selected revealed similar results with our approach choosing on average 9.6 features versus 10.2 features chosen by the *SGA-BP-Wrapper-CV* approach. Moreover, the low and high values for the Confidence Interval for the number of features selected are quite similar (see Table 4.7).

It should be noted that the accuracy on “LIVE” instances was much higher than for “DIE” instances in Table 4.7. However, the average accuracy rate for the *BP-CV* approach is very similar to the result obtained by [Cestnik et al., 1987] whom reported an 83 percent accuracy rate.

As with the ionosphere dataset, there is a fairly high commonality in the features selected amongst the Pareto-optimal solutions within each *fold* for this dataset. Table 4.8 illustrates the Pareto-optimal feature subsets discovered by our approach for *fold* 1.

Table 4.8: Features Selected for Fold 1: Hepatitis

Solution	Features Selected
1	18
2	3, 18
3	2, 6, 11, 18
4	6, 7, 8, 14, 16, 18
5	1, 2, 6, 7, 8, 9, 10, 11, 15, 16, 18

In Table 4.8, feature 1 corresponds to the “AGE” attribute Table 4.1, feature 2 corresponds to the “SEX” attribute Table 4.1, and so on. Notice that in Table 4.8, except for features 1, 9, 10, and 15, the remaining features are selected in at least two solutions. Moreover, although not shown in Table 4.8, the intersection of features selected amongst Pareto-optimal solutions within each *fold* for the remaining nine *folds* show an even higher intersection rate than *fold* 1. However, as with the case of the ionosphere dataset, the intersection of features amongst solutions in the Pareto-optimal front across different *folds* is significantly lower. Specifically, features 17 and 18 (i.e. “ALBUMIN” and “PROTIME”, respectively) were most commonly selected, appearing in nine out of the ten *folds*. It is interesting to note that “PROTIME” has the highest number of missing attribute value (i.e. 67) while “ALBUMIN” has the third highest number of missing attribute values (i.e. 16). These findings indicate that these attributes warrants further investigation.

Though the results obtained in this experiment show a progressive improvement (i.e. in accuracy rate and the model complexity) in favour of our approach, the execution time required by our approach is also much higher. Shown below in Table 4.9 is the execution time for each approach for one *fold*.

Table 4.9: Execution Time

Dataset	<i>BP-CV</i>	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Ionosphere	18 minutes	3 hrs	10 hrs, 30 minutes
Hepatitis	24 minutes	2 hrs, 18 minutes	9 hrs, 9 minutes

The results in Table 4.9 are based on running our code on the Java(TM) 2 Runtime Environment, Standard Edition Version 1.4.2 with a Linux Red Hat 7.2 installation. The hardware used to perform the experiments was modest, consisting of an Asus A7V266-E DDR motherboard, a 1.61 GHz AMD Athlon Processor, and 256 MB of RAM.

Examination of Table 4.9 clearly indicates the increasing execution time required as the complexity of the approach increases. Moreover, because the ionosphere dataset contains more features than the hepatitis dataset (34 versus 19), its execution time is higher; however, the

difference in execution time required does not appear to be proportional to the difference in the total number of features for the datasets.

4.5.6 Summary of Experiment

Comparison of the performance (i.e. accuracy rate and model complexity) of the *BP-CV* approach, the *SGA-BP-Wrapper-CV* approach, and our approach when applied to the ionosphere and hepatitis datasets have clearly shown that executing the backpropagation alone results in lower performance. However, comparison of our approach to that of the *SGA-BP-Wrapper-CV* approach has shown the results to be quite similar although our approach on average selected fewer features for both datasets. In the next section, we perform experiments to compare the convergence property of our approach to that of the *SGA-BP-Wrapper-CV* approach. That is, we are interested in finding out how many generations it will “normally” take before an *optimal* (Definition 3.5) or globally optimal solution (defined in the next section) is found using our approach and the *SGA-BP-Wrapper-CV* approach.

4.6 Comparison of Convergence

As was mentioned in the end of the previous section, in this experiment we are interested in knowing how fast our approach converges to a *local optimal* (Definition 3.5) or *globally optimal* solution compared to that of the *SGA-BP-Wrapper-CV* approach. In particular, we record the number of generations an approach takes before the *globally optimal* solution is found (or the *best* solution if the *globally optimal* is unknown), and from these data points determine statistically whether there is a difference in the speed of converging to the *best* or *global optimal* solution. Here, *globally optimal* refers to the best solution that can be found in a theoretically infinite set of solutions while the *best* qualifier refers to our definition of *approximately optimal* in Definition

3.5. Since our approach returns a set of non-dominating solutions, the generation that contains the *best* solution is the generation with the largest elite population (i.e. this implies having the greatest diversity) and contains the *best* or *globally optimal* solution.

To generate the needed data to perform the test of significance, we arbitrarily select a *fold* k (in our case $k = 1$) and execute it 30 times (with 50 generations per execution) using the *SGA-BP-Wrapper-CV* approach and our approach. We will refer to each execution of the arbitrarily selected *fold* k as a *sample experiment*. Thereafter, we take note of the number of generations it took to find the *best* solution from each *sample experiment*; the number of generations observed is a statistical random variable that is also known as the response variable in this experiment. Thus, for each approach we perform 30 *sample experiments* to obtain 30 data points required for the statistical significance test. The parameters settings done in this experiment are the same as in the previous experiment (see section 4.5) for both the ionosphere dataset (see Table 4.3) and the hepatitis dataset (see Table 4.6).

As mentioned earlier, the GA may not always find the global optimal solution. Experimental experiences have shown that such behaviour results in data points that are never normally distributed, and so we use the Wilcoxon’s Rank-Sum test to do our statistical analysis [Conover, 1998]. Appendix A describes in detail the Wilcoxon’s Rank-Sum test used in our experiments. For both the ionosphere and hepatitis dataset, a test of significance at the 95% Confidence Interval (CI) assumes the null hypothesis to mean no difference in the speed in which the two approaches converge to the *best* solution, while the alternative hypothesis means that there is a difference (i.e. a two-tailed distribution test).

4.6.1 Ionosphere Dataset: Results of Convergence Property

The results based on the Wilcoxon’s Rank-Sum test at a statistical significance level of 0.05 for the ionosphere dataset are shown below in Table 4.10.

Table 4.10: Test of Significance: Ionosphere

Statistical Parameter	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Number of Runs Count	30	30
Degrees of Freedom	58	
Rank-sum W	728	1102
Arithmetic Mean	915	915
Standard Deviation	67.6387463	67.6387463
P - value	0.00762648	0.00762648
α (significance level)	0.05	
<i>Binomial Value b</i>	10	
Lower Bound	14	21
Median	19	30.5
Upper Bound	24	35

The results of Table 4.10 show that there is strong evidence against the null hypothesis of no difference based on the two-tailed Student t-distribution. As can be seen in Table 4.10, the P -value $2P(T \geq |2.76468755|)$ is equal to 0.00762648, indicating that the chances of observing such a result based on the assumption of no difference is extremely small and thus is statistically significant at the 0.05 significance level. The difference in the two approaches in terms of the number of generations it took to find the *best* solution can readily be seen by referring to the Confidence Interval in Table 4.10. For the *SGA-BP-Wrapper-CV* approach, the interval is [14, 24] with a median value of 19, while for our approach it is significantly higher at [21, 35], with a median value of 30.5. The reason that our approach performed worse against that of the *SGA-BP-Wrapper-CV* approach may be attributed to the fact that our algorithm explicitly maintains population diversity. That is, because our approach allocates solutions to more than one region of the search space, for a constant size population, there are less schemas representing any given region (including the region containing the solution with the highest accuracy rate); the result is that it may take the GA longer to converge to an *optimal* solution in a given region than if most solutions in the population are “pushed” towards one *optimal* region as in the case of the *SGA-BP-Wrapper-CV* approach.

Although the *SGA-BP-Wrapper-CV* approach is faster than our approach, it is still not capable of finding the Pareto-optimal front because it does not explicitly maintain population diversity. To show this, we use and graph the parent population from the last GA generation of *fold 1* from the previous experiment (see section 4.5) for our approach and for the *SGA-BP-Wrapper-CV* approach (see Figures 4.5 and 4.6, respectively); *fold 1* was arbitrarily chosen since the results from the other *folds* are not significantly different from one another. In Figures 4.5 and 4.6, the accuracy rates have been converted into error rates in order to show the possible problem structure (i.e. a convex structure). Also, the titles are shortened for the purposes of readability. With respect to the legend in the graphs, “front 0” corresponds to the front containing the *best* non-dominating solutions (i.e. the Pareto-optimal solutions).

Parent Population (MOGA Wrapper) Ionosphere Dataset

Figure 4.5: Parent Population (MOGA-BP-Wrapper-CV): Ionosphere

Figure 4.6: Parent Population (SGA-BP-Wrapper-CV): Ionosphere

As can be seen by observing Figure 4.5 and Figure 4.6, our approach shows a more diverse parent population while for the *SGA-BP-Wrapper-CV* approach, the parent population tends to be clustered in a single region. It is this higher diversity population maintained by our approach that gives our method the ability to find a set of Pareto-optimal solutions that is as uniformly distributed as possible. See Figure 4.7.

The overall diversity shown in Figure 4.5 is due to two factors: (i) a vertical diversity mechanism (i.e. the *least-crowded selection algorithm*) which in this case seems to favour lower fronts (solutions clustered near the bottom-left of the graph), and (ii) a lateral diversity mechanism used to maintain solutions from other fronts to preserve lateral diversity across different fronts. These two phenomena will be analyzed in more detail in later experiments. However, as mentioned earlier, maintenance of a more diverse population to gain greater selection for the user (i.e. to find a set of well distributed solutions in the Pareto-optimal front) inevitably require more generations to be executed before a Pareto-optimal front can be found.

Figure 4.7: Pareto-Optimal Front (MOGA-BP-Wrapper-CV): Ionosphere

4.6.2 Hepatitis Dataset: Results of Convergence Property

We now analyze the convergence property of our approach to that of the *SGA-BP-Wrapper-CV* approach utilizing the hepatitis dataset. The results of the statistical significance test are given in Table 4.11.

Table 4.11: Test of Significance: Hepatitis

Statistical Parameter	<i>SGA-BP-Wrapper-CV</i>	<i>MOGA-BP-Wrapper-CV</i>
Number of Runs Count	30	30
Degrees of Freedom	58	
Rank-sum W	762	1068
Arithmetic Mean	915	915
Standard Deviation	67.6387463	67.6387463
P -value	0.02745946	0.02745946
α (significance level)	0.05	
Binomial Value b	10	
Lower Bound	16	29
Median	25.5	32
Upper Bound	31	37

Similar to the ionosphere dataset, the test of significance shows evidence that there is a difference in the rate at which the two approaches converge to the *best* solution(s) (i.e. we reject the null hypothesis). However, the P -value $2P(T \geq |2.26201709|)$ equal to 0.02745946 shows that it is only slightly less than the 0.05 significance level.

As with the ionosphere dataset, the difference in the two approaches in terms of the number of generations it took to find the *best* solution is noticeable; though for this dataset the difference is not as great. For this dataset, the interval is [16, 31] with a median value of 25.5 for the *SGA-BP-Wrapper-CV* approach, while for our approach it is slightly higher at [29, 37], with a median value of 32. Again, the difference may be due to the fact that our approach has to allocate a certain number of solutions to a search region while in the SGA all the solutions cluster around a single region.

In the previous subsection (4.6.1), we examined our approach's ability to find the Pareto-optimal front by graphing the elite population from the last GA generation (i.e. the 50-th generation) and discovered that the solutions lying approximately along the convex curve were quite diverse (see Figure 4.7). Moreover, we graphed the parent population from the last GA generation for both approaches and found that later into the simulation the SGA was clearly not capable of finding a set of Pareto-optimal solutions since its solutions tended to cluster in a single region (See Figures 4.5 and 4.6). Here, we follow the same procedure to see whether a similar pattern holds. Below are the graphs of the solutions from the parent population for the last GA generation parent population discovered by our approach and by the *SGA-BP-Wrapper-CV* approach (see Figures 4.8 and 4.9, respectively). The graph of the solutions in the elite population representing the Pareto-optimal front from our approach is shown in Figure 4.10.

Parent Population (MOGA Wrapper) Hepatitis Dataset

Figure 4.8: Parent Population (MOGA-BP-Wrapper-CV): Hepatitis

Figure 4.9: Parent Population (SGA-BP-Wrapper-CV): Hepatitis

Figure 4.10: Pareto-Optimal Front (MOGA-BP-Wrapper-CV): Hepatitis

Examining Figures 4.8 and 4.9, we definitely see a pattern that is quite similar to the results obtained from the ionosphere dataset in the previous subsection. Moreover, analysis of Figure 4.10 indicate that without an explicit diversity mechanism, solutions with 1, 2, and 4 features

selected would most likely be missed since the region where the solutions from the *SGA-BP-Wrapper-CV* approach is clustered around is quite distance from these three solutions.

4.6.3 Summary of Experiment

In this experiment, a test of significance for each dataset was performed to compare the convergence property of our approach and the *SGA-BP-Wrapper-CV* approach. The results for both datasets suggested that the *SGA-BP-Wrapper-CV* approach outperformed our approach and we surmise that it is because the latter approach concentrates all of its efforts to finding a single *optimal* solution. However, the gain in speed for the *SGA-BP-Wrapper-CV* approach also makes it most likely incapable of finding a set of Pareto-optimal solutions since it does not attempt to maintain diversity in the population; this was shown graphically when compared to our approach.

Up to this point we have shown that our approach compared to the other two approach is the only one capable of simultaneously satisfying all three metrics of maximizing predictive accuracy rate, minimizing model complexity, and maximizing selection for the user. However, there is still the question of whether our approach is capable of finding the “true” Pareto-optimal front. Since the “true” Pareto-front is not known in most cases, we fall back on our definition of *approximately optimal* from Definition 3.5. In the next section we show that without explicit mechanisms to push the search towards the “true” Pareto-optimal front, the GA will never converge to the “true” Pareto-front; and that the use of explicit mechanisms are required to direct the search towards the desired region.

4.7 The Effects of Lateral Diversity

As noted at the end of the previous section, we are interested in finding out whether our approach is capable of finding, or at the very least show signs of moving towards the “true” Pareto-optimal

front. That is, we want to compare the level of selection for the user of our approach with and without explicitly maintaining lateral diversity. To show this, we run for each dataset, our approach with the same parameters settings as before (Table 4.3 for the ionosphere dataset; Table 4.6 for the hepatitis dataset), except now we set the lateral diversity reduction rate to 0.0 (i.e. no lateral diversity), and then examine the solutions in the resulting Pareto-optimal front. After the execution of our approach run with no lateral diversity, we graphed the parent population and the elite population from the last GA generation of *fold 1*; *fold 1* is chosen since we can compare the results of this approach with the results in Figures 4.5 and 4.7 (for the ionosphere dataset), and Figures 4.8 and 4.10 (for the hepatitis dataset).

4.7.1 Ionosphere Dataset: No Lateral Diversity

Shown below are the graphs of the parent population (Figure 4.11) and the elite population (i.e. Pareto-optimal front; Figure 4.12) from the last GA generation for the ionosphere dataset run without lateral diversity.

Figure 4.11: Parent Population (MOGA-BP-Wrapper-CV: No Lateral Diversity): Ionosphere

Figure 4.12: Pareto-Optimal Front (MOGA-BP-Wrapper-CV: No Lateral Diversity): Ionosphere

Notice that both of the above graphs are same. This is not surprising since 48 of the solutions in the parent population in the last GA generation were comprised of solutions where only one feature was selected (i.e. the first feature of the dataset). Comparing the results of this experiment with that of running our approach with lateral diversity (see Figures 4.5 and 4.7), there is little doubt that the latter approach has greater overall population diversity, hence indicating, at least for this dataset, that an explicit lateral diversity mechanism is required to move closer to the “true” Pareto-optimal front.

To further reinforce the evidence presented above, we provide in Table 4.12, the number of Pareto-optimal solutions in the last GA generation for each *fold* for both approaches.

Table 4.12: Number of Pareto-Optimal Solutions: Ionosphere

<i>fold k</i>	<i>MOGA-BP-Wrapper-CV</i> (with lateral diversity)	<i>MOGA-BP-Wrapper-CV</i> (without lateral diversity)
1	5	3
2	6	3
3	5	2
4	7	4
5	5	3
6	6	4
7	5	2
8	6	2
9	7	3
10	6	4

Examining Table 4.12, it is clear that the number of Pareto-optimal solutions discovered by the *MOGA-BP-Wrapper-CV* approach run with lateral diversity is greater than the number of Pareto-optimal solutions discovered by the *MOGA-BP-Wrapper-CV* approach run without lateral diversity. In fact, using the foregoing values in Table 4.12, we calculated the arithmetic mean and the standard deviation, and used these values to determine the Confidence Interval (i.e. (arithmetic mean) $\pm$ (standard deviation)). The results are summarized in Table 4.13 and show a clear difference in favour of the *MOGA-BP-Wrapper-CV* approach run with lateral diversity.

Table 4.13: Statistics for the Number of Pareto-Optimal Solutions Discovered: Ionosphere

Statistical Parameter	<i>MOGA-BP-Wrapper-CV</i> (with lateral diversity)	<i>MOGA-BP-Wrapper-CV</i> (without lateral diversity)
Arithmetic Mean	5.8	3
Standard Deviation	0.788811	0.816497
Confidence Interval	(5.011, 6.589)	(2.184, 3.816)

4.7.2 Hepatitis Dataset: No Lateral Diversity

We now turn to see whether a similar pattern is found in the hepatitis dataset. Shown below in Figures 4.13 and 4.14 are the graphs for the hepatitis dataset run with no lateral diversity.

Parent Population (MOGA Wrapper: No Lateral Diversity) Hepatitis Dataset

Figure 4.13: Parent Population (MOGA-BP-Wrapper-CV: No Lateral Diversity): Hepatitis

Figure 4.14: Pareto-Optimal Front (MOGA-BP-Wrapper-CV: No Lateral Diversity): Hepatitis

A cursory glance at Figure 4.13 seems to show that the parent population diversity is quite high. However, a closer examination will reveal that the lateral population diversity is quite low across fronts even though vertical diversity is high. This can easily be seen by comparing this graph with the graph in Figure 4.8 (i.e. our approach run with a lateral diversity mechanism). Recalling the graph of Figure 4.8, we can see that the solutions in the higher fronts (i.e. 8, 7, and to a lesser extent 6) lie further away from solutions in the lower fronts (i.e. 1 to 5) with the same corresponding horizontal axis value. However, in the graph of Figure 4.13, with the exception of the solutions with horizontal axis value of 1 (i.e. 1 feature selected), all the solutions with the same horizontal axis lie quite close to one another even though they are in different fronts from one another. The result is that less diversity exists in the mating pool for our approach run with no lateral diversity, thus leading to fewer new solutions being discovered for future populations (including the elite population). Now, comparing the graph in Figure 4.14 to that of the graph in Figure 4.10 (i.e. the Pareto-optimal front found by running our approach with lateral diversity), we observe a significant difference in the number of solutions found for the Pareto-optimal front (five solutions found with lateral diversity versus two solutions found without lateral diversity).

This result should come with no surprise since the diversity for the parent population for our approach run with no lateral diversity is low compared to that of the parent population run with lateral diversity.

The close proximity of solutions across different fronts in Figure 4.13 can be explained by the fact that as the simulation progresses, dominated solutions in higher fronts carrying schemas that cover other regions of the search space are eliminated due to low selection pressure. Although the resulting solutions that replace the foregoing dominated solutions are better, the problem is that these solutions are extremely similar to the solutions already present in the lower fronts. Hence, though these new solutions are better (i.e. lie in a lower front), they add very little lateral diversity needed in the population.

To further reinforce the evidence presented above, we provide in Table 4.14, the number of Pareto-optimal solutions in the last GA generation for each *fold* for both approaches.

Table 4.14: Number of Pareto-Optimal Solutions: Hepatitis

<i>fold k</i>	<i>MOGA-BP-Wrapper-CV</i> (with lateral diversity)	<i>MOGA-BP-Wrapper-CV</i> (without lateral diversity)
1	5	2
2	5	2
3	4	2
4	6	3
5	5	3
6	5	2
7	5	3
8	6	2
9	5	3
10	5	2

As with the ionosphere dataset, Table 4.14 indicates a significant difference in favour of the *MOGA-BP-Wrapper-CV* approach run with lateral diversity. As done earlier for the ionosphere dataset, we use the foregoing values in Table 4.14 to calculate the arithmetic mean and the standard deviation, and then use these values to determine the Confidence Interval (i.e.

(arithmetic mean) $\pm$ (standard deviation)). The results are summarized in Table 4.15 and again show a clear difference in favour of the *MOGA-BP-Wrapper-CV* approach run with lateral diversity.

Table 4.15: Statistics for the Number of Pareto-Optimal Solutions Discovered: Hepatitis

Statistical Parameter	<i>MOGA-BP-Wrapper-CV</i> (with lateral diversity)	<i>MOGA-BP-Wrapper-CV</i> (without lateral diversity)
Arithmetic Mean	5.1	2.4
Standard Deviation	0.567646	0.516398
Confidence Interval	(4.532, 5.668)	(1.884, 2.916)

The lack of diversity in the parent populations for the ionosphere and hepatitis datasets for our approach run with no lateral diversity gives strong evidence that the “true” Pareto-optimal front will surely not be found in the later stages of the simulation unless an explicit lateral diversity mechanism is used. Whether or not our approach can truly find the “true” Pareto-optimal front using a lateral diversity mechanism may never be unknown, however, because our approach does find more Pareto-optimal solutions it indicates that such as mechanism is a step in the right direction.

4.8 The Effects of Vertical Diversity

In the previous experiment, we investigated the effect of maintaining lateral diversity in the population. In this experiment we continue our investigation of population diversity, but now we turn to look at the effects vertical diversity has on the GA’s ability to find the “true” Pareto-optimal front.

As was the case for the previous experiment, we are interested in finding out whether our approach is capable of finding, or at the very least show signs of moving towards the “true” Pareto-optimal front. That is, we are interested in comparing the level of selection for the user of

our approach to itself, but this time we focus on the effects of vertical diversity instead of lateral diversity. To show this, we run for each dataset, our approach with the same parameters settings as before (Table 4.3 for the ionosphere dataset; Table 4.6 for the hepatitis dataset). However, to illustrate the effect that our *least-crowded selection algorithm* and *least-crowded tournament selection operator* have on the GA's ability to find the "true" Pareto-optimal front, we made the following changes to our algorithm (see Figure 1.1); recall that these algorithms provide the mechanisms for maintaining vertical diversity in the population:

1. For the *least-crowded tournament selection operator*, if a given tournament corresponds to a non-dominating scenario, the winner is selected randomly. A dominating scenario is left unchanged. See Chapter 2, section 2.3.4.
2. If the elite population size $|E_i|$ for generation i is greater than N , we randomly delete $|E_i| - N$ solutions from E_i instead of using our *least-crowded selection algorithm* to select N solutions for the new elite population (step 2 in Figure 1.1).
3. To select N solutions for the new parent population, $|x_n|$ solutions from each front F_n are selected randomly instead of using our *least-crowded selection algorithm*, where $|x_1| + |x_2| + \dots + |x_n|$ equals size N .

As with the previous experiment, the graphs in this experiment are based on the results of *fold*

1. Also, to compare the effects of not explicitly maintaining vertical population diversity, we again make use of the graphs in Figures 4.5 and 4.7 (for the ionosphere dataset), and Figures 4.8 and 4.10 (for the hepatitis dataset).

4.8.1 Ionosphere Dataset: No Vertical Diversity

Shown below are the graphs of the parent population (Figure 4.15) and the elite population (i.e. Pareto-optimal front; Figure 4.16) from the last GA generation for the ionosphere dataset run with no vertical diversity.

Figure 4.15: Parent Population (MOGA-BP-Wrapper-CV: No Vertical Diversity): Ionosphere

Figure 4.16: Optimal Front (MOGA-BP-Wrapper-CV: No Vertical Diversity): Ionosphere

A cursory glance at the graph in Figure 4.15 seems to give an impression that running our approach with no vertical diversity gives a fairly good distribution of solutions in the search space. However, a closer examination of the graph in Figure 4.15 (vertical diversity) reveals that it has significantly fewer solutions represented along the horizontal and vertical axes than the graph of Figure 4.5 (with vertical diversity). This suggests a lower population diversity for the former approach. Specifically, the number of features selected ranges from 1 to 13 for our approach run with no vertical diversity, while the number of features selected ranges from 1 to 34 for our approach run with vertical diversity; a significant difference. With respect to the error rate, the results from our approach run with no vertical diversity show a range from 5.961 percent

to 15.57 percent, while our approach run with vertical diversity show a range from 7.947 percent to 95.364 percent. These large differences in the range for the two objective parameters indicate that running our approach without vertical diversity would not adequately cover the search space and as a result may cause the GA to more likely preserve solutions that lie away from the “true” Pareto-optimal front.

Notice that the graph of Figure 4.5 show the solutions in front 6, 7, 8, and 9 to lie quite a bit away from the lower fronts (0 to 5), while all the solutions in the graph of Figure 4.15 lie fairly close to one another (especially along the vertical axis). The reason why this may be happening is that when no emphasis is placed on maintaining vertical diversity, the solutions in the higher fronts that are furthest away (i.e. solutions lying near the boundaries of the front which most likely contain schemas that are quite different from solutions in the lower fronts) are not preserved. The result is that in later generations, the GA population tends to contain solutions that are quite similar even though they are categorized in different fronts.

Comparison of the Pareto-optimal front found in this experiment (Figure 4.16) with that of our approach run with vertical diversity (Figure 4.7) provides strong evidence that vertical diversity is needed. Notice in Figure 4.16 that only three solutions are found, while in Figure 4.7 five solutions are found. To further reinforce the evidence presented above, we provide in Table 4.16, the number of Pareto-optimal solutions in the last GA generation for each *fold* for both approaches.

Table 4.16: Number of Pareto-Optimal Solutions: Ionosphere

<i>fold k</i>	<i>MOGA-BP-Wrapper-CV</i> (with vertical diversity)	<i>MOGA-BP-Wrapper-CV</i> (without vertical diversity)
1	5	3
2	6	4
3	5	4
4	7	5
5	5	3
6	6	3
7	5	2
8	6	4
9	7	4
10	6	3

Examining Table 4.16, it is clear that the number of Pareto-optimal solutions discovered by the *MOGA-BP-Wrapper-CV* approach run with vertical diversity is greater than the number of Pareto-optimal solutions discovered by the *MOGA-BP-Wrapper-CV* approach run without vertical diversity. In fact, using the foregoing values in Table 4.16, we calculated the arithmetic mean and the standard deviation, and used these values to determine the Confidence Interval (i.e. (arithmetic mean) $\pm$ (standard deviation)). The results are summarized in Table 4.17 and show a clear difference in favour of the *MOGA-BP-Wrapper-CV* approach run with vertical diversity.

Table 4.17: Statistics for the Number of Pareto-Optimal Solutions Discovered: Ionosphere

Statistical Parameter	<i>MOGA-BP-Wrapper-CV</i> (with vertical diversity)	<i>MOGA-BP-Wrapper-CV</i> (without vertical diversity)
Arithmetic Mean	5.8	3.5
Standard Deviation	0.788811	0.849837
Confidence Interval	(5.011, 6.589)	(2.650, 4.350)

Overall, the results of this experiment gives an indication, at least for this dataset, that an explicit vertical diversity mechanism is required to move the GA closer to the “true” Pareto-optimal front.

4.8.2 Hepatitis Dataset: No Vertical Diversity

We now turn to see whether a similar pattern is found in the hepatitis dataset. Shown below are the graphs of the parent population (Figure 4.17) and the elite population (i.e. Pareto-optimal front; Figure 4.18) from the last GA generation for the hepatitis dataset run without vertical diversity.

Figure 4.17: Parent Population (MOGA-BP-Wrapper-CV: No Vertical Diversity): Hepatitis

Figure 4.18: Pareto-Optimal Front (MOGA-BP-Wrapper-CV: No Vertical Diversity): Hepatitis

Comparison of the results for our approach run without vertical diversity (Figures 4.17 and 4.18) to that of our approach run with vertical diversity (Figures 4.8 and 4.10) suggests a similar situation to the ionosphere dataset. Again, as with the ionosphere dataset, the parent population for this dataset seems to show a fair amount of diversity in the population. However, when we compare the horizontal and vertical axes values associated with the solutions in the parent population in this experiment (Figure 4.17) to that of the graph in Figure 4.8, we find that our approach run with vertical diversity found solutions that are more widely dispersed in the search space: 1 to 19 features selected (with vertical diversity) versus 1 to 10 features selected (no vertical diversity); and an error rate that range from 11.112 percent to 73.334 percent (with vertical diversity) versus 11.111 percent to 33.333 percent error rate (no vertical diversity).

Unlike the case with the ionosphere dataset, only the solutions in the last two higher fronts (i.e. 7 and 8) lies quite a bit away from the lower fronts (i.e. 0 to 6) when vertical diversity is

emphasized (Figure 4.8). However, when vertical diversity is not emphasized, the parent population found by the GA for this dataset is similar to the parent population found for the ionosphere dataset (i.e. all the solutions lie fairly close to one another, especially along the vertical axis). Even though solutions in only fronts 7 and 8 lie quite away from the rest of the solutions in the population, higher diversity still exists in the population which can significantly affect the resulting Pareto-optimal front discovered by the GA. This conclusion can be seen by comparing the number of solutions in the Pareto-optimal front for the two approaches (i.e. four solutions when no emphasis is placed on vertical diversity versus five solutions for our approach run with vertical diversity). To further reinforce the evidence presented above, we provide in Table 4.18, the number of Pareto-optimal solutions in the last GA generation for each *fold* for both approaches.

Table 4.18: Number of Pareto-Optimal Solutions: Hepatitis

<i>fold k</i>	<i>MOGA-BP-Wrapper-CV</i> (with vertical diversity)	<i>MOGA-BP-Wrapper-CV</i> (without vertical diversity)
1	5	4
2	5	3
3	4	3
4	6	5
5	5	3
6	5	4
7	5	3
8	6	4
9	5	4
10	5	4

As with the ionosphere dataset, Table 4.18 indicates a difference in favour of the *MOGA-BP-Wrapper-CV* approach run with vertical diversity. As done earlier for the ionosphere dataset, we use the foregoing values in Table 4.18 to calculate the arithmetic mean and the standard deviation, and then use these values to determine the Confidence Interval (i.e. (arithmetic

mean) $\pm$ (standard deviation)). The results are summarized in Table 4.19 and again show a clear difference in favour of the *MOGA-BP-Wrapper-CV* approach run with vertical diversity.

Table 4.19: Statistics for the Number of Pareto-Optimal Solutions Discovered: Hepatitis

Statistical Parameter	<i>MOGA-BP-Wrapper-CV</i> (with vertical diversity)	<i>MOGA-BP-Wrapper-CV</i> (without vertical diversity)
Arithmetic Mean	5.1	3.7
Standard Deviation	0.567646	0.674949
Confidence Interval	(4.532, 5.668)	(3.025, 4.375)

4.9 Conclusion

In this chapter we discussed in detail the architectures of each algorithm used to perform each of the experiments. To validate our theoretical results from Chapter 3, we performed a number of experiments to show that our approach has admirable characteristics that warrants its use despite its higher computational complexity. The empirical results from our experiments showed that using the backpropagation algorithm alone (i.e. *BP-CV* approach) for feature detection performs poorly compared to that of the wrapper approach using the Simple Genetic Algorithm (i.e. *SGA-BP-Wrapper-CV* approach) in terms of both the predictive accuracy rate and the number of features selected. Thereafter, we compared the *SGA-BP-Wrapper-CV* approach to that of our approach (i.e. the *MOGA-BP-Wrapper-CV* approach). We discovered that both approaches performed equally well in terms of accuracy rate, but that the *SGA-BP-Wrapper-CV* approach fared better when it came to the speed in which the *best* solutions are found. However, because the *SGA-BP-Wrapper-CV* approach does not find a set of Pareto-optimal solutions, it offers a lower selection of solutions from which the user can choose from.

In our second experiment, we compared how fast our approach converges to an *optimal* solution to that of the *SGA-BP-Wrapper-CV* approach. A test of statistical significance at the

95% Confidence Interval found that there was statistical significance in favour of the *SGA-BP-Wrapper-CV* approach.

In our third experiment, we compared our approach with and without explicitly maintaining lateral diversity in the population. We discovered that when no lateral diversity mechanism was used on the datasets, lower selection for the user is most likely to occur because either the “true” Pareto-optimal front is missed or most of it is not discovered by the GA.

In our last experiment, we compared our approach with and without explicitly maintaining vertical diversity in the population. Although the results favouring vertical diversity were not as strong as in the case of lateral diversity, it was nevertheless an important mechanism that the GA needs if it were to move closer to the “true” Pareto-optimal front.

In the next and final chapter we summarize the accomplishments made in this research, give some concluding remarks pertaining to our work, and suggest some possible future work related to this research.

Chapter 5

Conclusion

5.1 A MOGA Wrapper Approach for Feature Selection

In this thesis we have examined the feature subset selection problem and proposed a Multi-Objective Genetic Algorithm (MOGA) based on the wrapper approach to tackle this NP-hard problem. The problems we have identified in this thesis have been addressed in theory and supported by experimental results.

The principal algorithm we have developed is a diversity mechanism that favours the least crowded solutions in a set containing non-dominating solutions; this algorithm called the *least-crowded selection algorithm* forms the basis for our other contribution: the *least-crowded tournament selection operator*. Combining the lateral diversity mechanism proposed by Deb and Goel (2001) with that of our algorithms, we developed a system that is capable of finding a set of Pareto-optimal feature subset.

5.2 Summary of Contribution

At the beginning of this research we identified the limitations of using the entire feature set of a real-world dataset for data mining. Specifically, typical real-world datasets tend to have features that may be irrelevant and may even impede the performance of the data mining task.

To test the foregoing claim, we utilized a regular backpropagation (BP) algorithm alone for detecting the features and examined its performance (i.e. the *BP-CV* approach) for two real-world datasets (i) the ionosphere dataset, and (ii) the hepatitis dataset. We then compared the results obtained by the *BP-CV* approach (i.e. the accuracy rate) to that of the results running the *SGA-BP-Wrapper-CV* approach on the same two datasets, which as the name implies uses the Simple Genetic Algorithm (SGA) as a wrapper around a BP algorithm for feature subset selection. We kept all the same parameters settings for the BP algorithm used with the *BP-CV* approach when running the *SGA-BP-Wrapper-CV* approach; except for the number of hidden nodes which is evolved during the Genetic Algorithm (GA) simulation. Comparing the performance of the *BP-CV* approach to that of the *SGA-BP-Wrapper-CV* approach showed that the latter approach outperformed the former approach by a fairly significant margin.

Next, we compared our approach to that of the *BP-CV* approach and the *SGA-BP-Wrapper-CV* approach. Our approach outperformed the *BP-CV* approach in the same manner that the *SGA-BP-Wrapper-CV* approach outperformed the *BP-CV* approach. However, a comparison of our approach with that of the *SGA-BP-Wrapper-CV* approach showed similar performance level, though our approach selected fewer features with respect to both datasets.

For our second experiment we compared the speed with which our approach and the *SGA-BP-Wrapper-CV* approach can find the solution with the highest accuracy rate. A test of significance at the 95% Confidence Interval showed that the *SGA-BP-Wrapper-CV* approach outperformed our approach. This result is not surprising since our approach needs to allocate a certain number

of solutions to each region (i.e. maintain diversity) and so fewer schemas are available to converge to a single *optimal* (see Definition 3.5) region.

For our third experiment, we analyze the effect of the lateral diversity mechanism by setting the reduction rate to 0.0 while keeping all the same parameters settings used in earlier experiments that ran with lateral diversity. We discovered that the parent populations in the final GA generation had significantly lower diversity and that the resulting Pareto-optimal front had fewer solutions, thus indicating that the “true” Pareto-optimal front is more than likely missed and that a lateral diversity mechanism is required to find stronger results.

In our final experiment, we analyzed the effect our *least-crowded selection algorithm* and *least-crowded tournament selection operator* has with respect to the GA finding the “true” Pareto-optimal front. As with the experiment on lateral diversity, we kept all the same parameters settings used in earlier experiments that ran with vertical diversity and found that the results does support the use of a vertical diversity mechanism.

5.3 Future Work and Conclusion

This research has developed and examined a new Multi-Objective Genetic Algorithm (MOGA) for the feature subset selection problem based on the wrapper approach. It has been validated on two real-world datasets. But, there is still further research to be done. The work we think should be considered includes the following:

1. Mitigate the computational complexity by a parallel implementation.
2. More extensive experiments using datasets with small sample size compared to the number of features in the dataset (i.e. protein molecule datasets used for drug discovery problems).

3. A parent population size that can change during simulation to adapt to the number of potential regions of solutions in the Pareto-optimal front varies since enough individuals should be made available to provide adequate representation of each region (i.e. this may speed up the convergence property of the GA).
4. Penalty functions for costly features or features that contain missing values.

In spite of the vast amount of further work required, this thesis has made a contribution to the fields of data mining and genetic algorithm by providing a number of novel algorithms which were empirically validated and showed promises.

Appendix A

The Wilcoxon Rank Sum test is a non-parametric method used in situations when the distributions of two sample sets do not have a Gaussian distribution. The steps to determine whether two non-Gaussian datasets are statistically significant works as follows:

1. The two datasets are sorted together in either descending or ascending order. The rank value of each data instance is noted and the dataset it belongs to is noted.
2. Split the two datasets again keeping the combined ranking with each data instance.
3. Create the statistic W_1 , which is the sum of the ranks of the first dataset.
4. Compute the mean μ_1 and variance σ^2 of W_1 as follows:

$$\mu_1 = \frac{n_1(n_1 + n_2 + 1)}{2}; \quad \sigma^2 = \frac{n_1 n_2 (n_1 + n_2 + 1)}{12}.$$

In the above formulas, n_1 and n_2 are the number of data instances in the first dataset second dataset, respectively.

5. Perform the Student's T test on the normalized W_1 . To do this, the statistic t is first calculated:

$$t = \frac{W_1 - \mu_1}{\sigma}.$$

Next, the degrees of freedom dof are calculated: $n_1 + n_2 - 2$. Thereafter, the statistic t and the degrees of freedom dof are used together to determine the P -value for a two-tail Student's T distribution. The P -value is $2P(T \geq |t|)$, where T has the $t(dof)$ distribution.

The above test of significance only gives a P -value to determine whether or not to accept or reject a null hypothesis for a given Confidence Interval (CI). To find the actual CI for each population, the median is used. The arithmetic mean is not used because the distributions of our datasets are not normally distributed and using it would give misleading results.

To find the Confidence Interval around the median, we must first find the smallest binomial value b for which the cumulative binomial distribution is greater than or equal to a criterion value $\alpha / 2$.

After b is resolved, the ceiling of the lower percentile $l = \frac{b}{n-1}$ and the ceiling of the upper

percentile $u = \frac{(n-1)-b}{n-1}$ are calculated to determine the lower and upper percentile of the sorted

dataset, where n is the number of data points in the dataset. In our experiment n is equal to 30 (i.e. the number of experiments performed using the same *fold*).

Bibliography

- [Bala et al., 1997] Bala, J., De Jong, K., Huang, J., Vafaie, H., and Wechsler, H. (1997). Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation* 4(3), pp. 297-312.
- [Bins, and Draper, 2001] Bins, J. and Draper, B. (2001). Feature selection from huge feature sets. *International Conference on Computer Vision (ICCV'01)* 2, pp. 159-165. IEEE.
- [Cardie, 1993] Cardie, C. (1993). Using decisions trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 25-32. Amherst, MA: Morgan Kaufmann.
- [Cestnik et al., 1987] Cestnik, G., Kononenko, I., and Bratko, I. (1987). Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In *I. Bratko & N. Lavrac (Eds.) Progress in Machine Learning*, pp. 31-45. Cited from [Machine Learning Repository, 2004].
- [Chen et al., 1999] Chen, S., Guerra-Salcedo, C., and Smith, S. (1999). Non-standard crossover for a standard representation – commonality-based feature subset selection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 129-134.
- [Chen and Smith, 1999] Chen, S. and Smith, S. (1999). Introducing a new advantage of crossover: Commonality-based selection. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 6-12.
- [Cherkauer and Shavlik, 1996] Cherkauer, K. and Shavlik, J. (1996). Growing simpler

- decision trees to facilitate knowledge discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 315—318. San Mateo, CA: AAAI Press.
- [Conover, 1998] Conover, W. (1998). *Practical Nonparametric Statistics*. 3rd Edition. New York: Wiley Text Books.
- [Davies, and Bouldin, 1979] Davies, D. and Bouldin, W. (1979). A cluster separation measure. *IEEE PAMI 1*, pp. 224-227. Cited from Morita et al. (2003).
- [Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. New York: John Wiley & Sons.
- [Deb et al., 2000] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated Sorting genetic algorithm for multi-objective optimization: NSGA-II. *IEEE Transaction on Evolutionary Computation 6*(2), pp.182-197.
- [Deb and Goel, 2001] Deb, K. and Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-2001)*, pp. 67-81.
- [De Jong, 1975] De Jong, K. A. (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. Ph.D. Thesis, Ann Arbor, MI: University of Michigan. Dissertation Abstracts International 36(10), 5140B (University Microfilms No. 76-9381). Cited from Deb (2001).
- [Emmanouilidis et al., 2000] Emmanouilidis, C., Hunter, A., and MacIntyre, J. (2000). A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC'2000)*, pp. 309-316. IEEE.
- [Emmanouilidis et al., 1999] Emmanouilidis, C., Hunter, A., MacIntyre, J., and Cox, C. (1999). Multiple-criteria genetic algorithms for feature selection in neurofuzzy modeling. In *1999 International Joint Conference on Neural Networks (IJCNN'99)*, pp. 387-392. IEEE.

- [Fausett, 1994] Fausett, L. V. (1994). *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*. Englewood Cliffs, NJ: Prentice Hall.
- [Freitas, 1998] Freitas, A. (1998). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. New York: Springer.
- [Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley Professional.
- [Goldberg and Deb, 1991] Goldberg, D. and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms 1(FOGA-1)*, pp. 69-93.
- [Guerra-Salcedo et al., 1999] Guerra-Salcedo, C., Chen, S., Whitley, D., and Smith, S. (1999). Fast and accurate feature selection using hybrid genetic strategies. In *Proceedings of the Congress on Evolutionary Computation (CEC99)*, pp. 177—184.
- [Hansen, 1996] Hansen, M. P. (1996). Tabu search for multiobjective optimization: MOTS. Technical Report, *13th International Conference on MCDM*.
- [Haykin, 1999] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. New Jersey: Prentice Hall.
- [Hecht-Nielsen, 1991] Hecht-Nielsen, R. (1991). *Neuro Computing*. U.S.A.: Addison-Wesley Publishing Company.
- [Horn et al., 1994] Horn, J., Nafploitis, N., and Goldberg, D. (1994). A niched Pareto genetic algorithm for multi-objective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 82-87.
- [Houck et al., 1996a] Houck, C., Joines, J., and Kay, M. (1996). Characterizing search spaces for tabu search. *European Journal of Operational Research*, pp. 1-21.
- [Houck et al., 1996b] Houck, C., Joines, J., and Kay, M. (1996). Utilizing Lamarckian evolution and the Baldwin Effect in hybrid genetic algorithms. NCSU-IE Technical Report 96-01, 1996.

- [Ishibuchi and Nakashima, 2000] Ishibuchi, H. and Nakashima, T. (2000). Multi-Computation Conference (GECCO '2000), pp. 1069-1076. Morgan Kaufmann. Cited from Freitas (1998).
- [Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science* 220 (4598), pp. 671-680.
- [k-means clustering, 2004] K-Means Clustering (2004).
<http://fconyx.ncifcrf.gov/~lukeb/kmeans.html>. Webpage.
- [Kohavi, 1995] Kohavi, R. (1995). *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. Ph.D. Thesis, Stanford University. Cited from Chen et al. (1999).
- [Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2), pp. 273-324.
- [Llora and Garrel, 2001] Llora, X. and Garrel, J. (2001). Inducing partially-defined instances with evolutionary algorithms. In *Proceedings of the 18th International Conference on Machine Learning (ICML '2001)*, pp. 337-344. Morgan Kaufmann. Cited from Freitas (1998).
- [Machine Learning Repository, 2004] University of California, Irvine: Machine Learning Repository (2004). <http://www.ics.uci.edu/~mllearn/MLRepository.html>. Webpage.
- [Metropolis et al., 1953] Metropolis, N., Rosenbluth, A., Rosenbluth, N., Teller, H., and Teller, J. (1953). Equation of state calculation by fast computing machines. *Journal Chemical Physics* 21, pp. 1087-1092.
- [Moody and Utans, 1991] Moody, J. and Utans, J. (1991). Principled architecture selection for neural networks: Application to corporate bond rating prediction. In *Advances in Neural Information Processing Systems 4*, pp. 683-690. Morgan Kaufmann.
- [Morita et al., 2003] Morita, M., Sabourin, R., Bortolozzi, F., and Suen, C. Y. (2003). Unsupervised feature selection using multi-objective genetic algorithms for handwritten

- word recognition. In *7th International Conference on Document Analysis and Recognition (ICDAR'03)*, pp. 666-670.
- [Namand Park, 2000] Nam, D. and Park, C. (2000). Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems* 2(2), pp. 87-97.
- [Neural Network FAQ, 2004] Neural Network FAQ (2004).
<ftp://ftp.sas.com/pub/neural/FAQ.html>. Webpage.
- [Nürnberger et al., 2000] Nürnberger, A., Klose, A., and Kruse, R. (2000). Effects of antecedent pruning in fuzzy classification systems. In *Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies (KES 2000)*, pp. 154-157.
- [Oh et al., 2002] Oh, I., Lee, J., and Moon, B. (2002). Local search-embedded genetic algorithms for feature selection. pp. 148-151.
- [Oliveira et al., 2003] Oliveira, L., Sabourin, R., Bortolozzi, F., and Suen, C. Y. (2003). A methodology for feature selection using multi-objective genetic algorithms for handwritten digit string recognition. *International Journal on Pattern Recognition and Artificial Intelligence* 17(6), pp. 903-929.
- [Onnia et al., 2001] Onnia, V., Tico, M., and Saarinen, J. (2001). Feature selection method using neural network. *IEEE International Conference on Image Processing*, pp. 513-516.
- [Pappa et al., 2002] Pappa, G. L., Freitas, A. A., and Kaestner, C. A. A. (2002). Attribute selection with a multiobjective genetic algorithm. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA-2002), (Lecture Notes in Artificial Intelligence 2507)*, pp. 280-290.
- [Pudil et al., 1994] Pudil, P., Novovicova, J., and Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recognition Letters* 15, pp. 1119-1125. Cited from Raymer et al. (2000).

- [Punch et al., 1993] Punch, W., Goodman, E., Pei, M., Chia-Shun, L., Hovland, P., and Enbody, R. (1993). Further research on feature selection and classification using genetic algorithms. In *Proceedings of the 5th International Conference Genetic Algorithms (ICGA '93)*, pp. 557-564.
- [Raymer et al., 2000] Raymer, M., Punch, W., Goodman, E., Kuhn, L., and Jain, A. (2000). Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation* 4(2), pp. 164-171.
- [Siedlecki and Sklansky, 1989] Siedlecki, W. and Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* 10(5), pp. 335-347.
- [Sigillito et al., 1989] Sigillito, V. G., Wing, S. P., Hutton, L. V., and Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest* 10, pp. 262-266.
- [Spears, 1998] Spears, W. M. (1998). *The Role of Mutation and Recombination in Evolutionary Algorithms*. Ph.D. Thesis, Fairfax, VA: George Mason University.
- [Srinivas, and Deb, 1994] Srinivas, N. and Deb, K. (1994). Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation Journal* 2(3), pp. 221-248.
- [Winston, 1994] Winston, W. L. (1994). *Operations Research: Applications and Algorithms*. Belmont, California: Duxbury Press.
- [Yang and Honavar, 1998] Yang, J. and Honavar, V. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems* 13(2), pp. 44-49.
- [Yang et al., 1997] Yang, J., Parekh, R., and Honavar, V. (1997). DistAl: An inter- pattern distance-based constructive learning algorithm. Technical Report ISU-CS-TR 97-05, Iowa State University. Cited by Yang and Honavar (1998).
- [Zitzler and Thiele, 1998] Zitzler, E. and Thiele, L. (1998). An evolutionary algorithm

for multiobjective optimization: The strength Pareto approach. Technical Report 43, Zürich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH).